Dipl.-Ing. Josef Steinbäck, BSc

An Environmental Perception Platform Enabling Low-Level Sensor Fusion

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Eugen Brenner

Institut für Technische Informatik

Advisor
Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Graz, March 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

Acknowledgments

This thesis is the final milestone of my doctoral studies at the Graz University of Technology. It was carried out at the Institute of Technical Informatics in cooperation with Infineon Technologies Austria in Graz. At this point, I would like to write some words of appreciation to all the people who supported me during the process of writing this thesis.

First and foremost, I would like to thank my supervisor Prof. Eugen Brenner and my advisor Dr. Christian Steger from the Institute of Technical Informatics for their continuous support, valuable feedback, and contributed knowledge during the process of this work. Further, I would like to express my sincere gratitude to Dr. Norbert Druml, my advisor at Infineon Technologies, who convinced me to enroll in a doctoral program after completing my Master's degree in 2016. As a mentor, he provided me with useful advice and insightful comments. Due to his positive attitude, he was able to prevent me from quitting in moments of great doubt.

Furthermore, I would like to thank my colleagues at the *Cooperative Research and Exploration* department of the Infineon Development Center Graz for the great working atmosphere. Special thanks go out to my PhD student colleagues Andreas, Armin, Catarina, Hannes, and Philipp. I highly appreciate all of your useful advice, comments, and interesting discussions. Additionally, I would like to thank the funding agencies involved in financing the research projects assigned to this thesis[1]. The Virtual Vehicle Research Center in Graz, one of the project partners, largely contributed to this thesis's research outcome by providing their research vehicles for multiple test drives. In particular, I would like to thank Markus Schratter for his efforts in order to make this happen.

Finally, I would like to express my very profound gratitude to my family and friends for their support and patience during my studies and during the work on this thesis.

Graz, March 2021 Josef Steinbäck

[1]The work of this thesis was accomplished as part of multiple research projects. ACTIVE (*Autonomous CarTo Infrastructure communication mastering adVerse Environments*) is an *Austrian Research Promotion Agency* (FFG)-supported project with the number 855010. AutoDrive and PRYSTINE (*Programmable Systems for Intelligence in Automobiles*) are *Electronic Components and Systems for European Leadership Joint Undertaking* (ECSEL JU)-supported projects with the grant agreement numbers 737469 and 783190.

Abstract

Environmental perception sensors are among the key enablers of complex autonomous systems such as automated vehicles or autonomous robots. However, state-of-the-art perception systems are incapable of providing the required level of quality and robustness in various conditions (e.g., harsh weather). No single-sensor solution is currently available to meet these requirements. Thus, multiple different perception sensors are integrated into today's perception systems. Research and the industry currently face a significant gap. The ideal sensor composition and the best possible combination of the corresponding sensor data still impose open research questions. Additionally, the research community lacks the availability of open perception platforms to perform research on novel sensor fusion concepts at sensor-level. Researchers either face the burden to build their own custom sensor platform or have to rely on publicly available high-level datasets.

This work provides the research community with a blueprint on how to build an environmental perception platform, targeting the exploration of novel sensor fusion concepts. For that purpose, the hardware and software components of a low-level capable environmental perception platform were designed and implemented from scratch. The platform deploys multiple state-of-the-art perception sensors (radar, vision, time-of-flight) and provides the measurement data streams at various abstraction levels. The *Robot Operating System* (ROS) was selected as the base framework to enable rapid development and to provide an open interface to existing modules. Crucial implementation challenges were addressed, including the temporal and spatial alignment of low-level sensor data and the sensors' dynamic reconfiguration. Multiple common perception tasks were designed and implemented as use cases for the environmental perception platform. These use cases utilize the low-level sensor data in order to demonstrate the capabilities of the proposed platform and to introduce novel approaches to solve the perception tasks.

The platform was mounted on different vehicles to obtain measurement data of real-world scenarios. This data was utilized to evaluate the platform's perception capabilities and the sensors' data quality. The obtained real-world data proves that the resulting perception quality can be increased if the perception system enables low-level access to the sensors. The evaluation of the implemented use cases clearly displays the potential of low-level sensor fusion for various perception applications. The proposed environmental perception platform is well-suited to be used as a base system to develop and evaluate novel sensor fusion concepts. Researchers can utilize this work's approach as a reference for the construction of similar low-level capable perception platforms. The fast and easy access to such systems advances the associated research and contributes to increase the quality and robustness of future perception systems.

Kurzfassung

Sensoren zur Umgebungswahrnehmung gehören zu den wichtigsten Voraussetzungen für komplexe autonome Systeme wie selbstfahrende Fahrzeuge oder autonome Roboter. Aktuelle Systeme sind jedoch nicht in der Lage, die erforderliche Qualität und Robustheit unter wechselnden Bedingungen zu liefern. Es gibt derzeit keine ausreichende Lösung mit nur einem Sensor, daher werden in den heutigen Systemen mehrere verschiedene Umgebungssensoren integriert. Die ideale Auswahl von Sensoren und die bestmögliche Kombination der entsprechenden Sensordaten stellen noch offene Forschungsfragen dar. Zusätzlich fehlt es der Forschungsgemeinschaft an frei-verfügbaren Systemen, um neuartige Sensorfusionskonzepte auf Sensorebene zu erforschen. Forscherinnen und Forscher müssen ihre eigene Plattform bauen, oder sich auf öffentlich-verfügbare High-Level-Datensätze verlassen.

Diese Arbeit liefert einen Entwurf für den Aufbau einer Sensorplattform, die auf die Erforschung neuartiger Sensorfusionskonzepte abzielt. Zu diesem Zweck wurden die Hardware- und Softwarekomponenten einer Low-Level-fähigen Sensorplattform von Grund auf entworfen und implementiert. Die Plattform setzt mehrere hochmoderne Umgebungssensoren (Radar, Vision, Time-of-Flight) ein und stellt die Messdaten auf verschiedenen Abstraktionsebenen bereit. Das *Robot Operating System* (ROS) wurde als Basisframework gewählt, um eine schnelle Entwicklung zu ermöglichen und eine offene Schnittstelle zu bestehenden Modulen zu bieten. Entscheidende Herausforderungen bei der Implementierung wurden adressiert, darunter der zeitliche und räumliche Abgleich der Low-Level-Sensordaten und die dynamische Rekonfiguration der Sensoren. Die vorgestellte Plattform wurde als Basissystem verwendet, um neuartige Sensorfusionskonzepte zu evaluieren. Mehrere typische Anwendungsfälle wurden entworfen und für die Sensorplattform implementiert. Diese Anwendungsfälle nutzen die Low-Level-Sensordaten, um die Stärken der Plattform zu demonstrieren und neue Lösungsansätze vorzustellen.

Die Plattform wurde auf verschiedenen Fahrzeugen montiert, um Messdaten von realen Szenarien zu erhalten. Diese Daten wurden verwendet, um die Eigenschaften der Plattform und die Datenqualität der Sensoren zu bewerten. Die gewonnenen Daten belegen, dass die Wahrnehmungsqualität gesteigert werden kann, wenn das System den Low-Level-Zugriff auf die Sensoren ermöglicht. Die Auswertung der implementierten Anwendungsfälle zeigt das Potenzial der Low-Level-Sensorfusion für verschiedene Anwendungsfälle. Die Plattform ist gut geeignet, um als Basissystem für die Entwicklung und Evaluierung neuartiger Sensorfusionskonzepte verwendet zu werden. Forscherinnen und Forscher können diese Arbeit als Referenz für die Konstruktion ähnlicher Low-Level-fähiger Plattformen nutzen. Der schnelle und einfache Zugang zu solchen Systemen bringt die zugehörige Forschung voran und trägt zur Verbesserung zukünftiger Wahrnehmungssysteme bei.

Contents

List of Figures

List of Tables

List of Abbreviations

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance System
ADS	Automated Driving System
API	Application Programming Interface
BO	Bayesian Optimization
CAN	Controller Area Network
CCD	Charge-Coupled Device
CFAR	Constant False Alarm Rate
CMOS	Complementary Metal–Oxide–Semiconductor
CPS	Cyber-Physical System
CNN	Convolutional Neuronal Networks
DNN	Deep Neuronal Networks
ECU	Electronic Control Unit
EKF	Extended Kalman Filter
FFT	Fast Fourier Transform
FMCW	Frequency-Modulated Continuous-Waveform
FPGA	Field-Programmable Gate Array
FPS	Frames per Second
GNSS	Global Navigation Satellite System
HOG	Histogram of Oriented Gradients
ICP	Iterative Closest Point
IF	Intermediate Frequency
IMU	Inertial Measurement Unit
KB	Knowledge Base
NIR	Near-Infrared
PCL	Point Cloud Library
PMD	Photonic Mixing Device
PTP	Precision Time Protocol
RANSAC	Random Sample Consensus
RF	Radio Frequency

ROS	Robot Operating System
SAE	Society of Automotive Engineers
SDK	Software Development Kit
SLAM	Simultaneous Localization and Mapping
SNR	Signal-to-Noise Ratio
SVD	Singular Value Decomposition
TCP	Transmission Control Protocol
ToF	Time-of-Flight
UWB	Ultra-wideband
V2I	Vehicle to Infrastructure
VCSEL	Vertical-Cavity Surface-Emitting Laser
XML	Extensible Markup Language

Glossary

Environmental Perception Platform
A physical platform deploying multiple perception sensors and processing modules to perform certain perception tasks. The platform consists of multiple subsystems, performing the associated processing tasks. Possible applications are the detailed analysis of individual sensor data and the evaluation of novel sensor fusion concepts for research purposes.

Base Perception Subsystem
The fundamental subsystem of the environmental perception platform, performing the low-level interaction with multiple perception sensors. The subsystem receives the measurement data and applies certain (pre-) processing steps. The base perception system outputs structured and well-aligned data streams at different abstraction levels.

Use-Case Subsystems
The environmental perception platform incorporates one or multiple use-case subsystems. Each use case utilizes a subset of the base perception system's provided data streams and performs application-specific processing (e.g., fusion, detection). Examples of use cases are the visualization of sensor data and obstacle detection.

Data Streams
Data streams describe the continuous messages communicated between the single modules and subsystems of the environmental perception system. The streams originate from sensor measurements or from processing modules. In addition to a message's main data section (e.g., raw data), a timestamp and a frame identifier are included.

Sensor Fusion
Sensor fusion describes the combination of sensor data from multiple perception sensors into a common representation. In general, the input data originates from a composition of multiple heterogeneous and homogeneous sensors. The fusion of multiple perception sensors enables joint processing in order to enhance a system's perception performance.

Spatial and Temporal Alignment
The spatial and temporal alignment of the sensor data has to be known in order to fuse the data from multiple sensors. Timestamps are a common way to assign the measurement time to the corresponding data streams. A data stream's spatial alignment is indicated with an identifier of the associated measurement frame.

Time-of-Flight Camera
These cameras utilize the speed of light to determine a distance image of the scene. A Time-of-Flight camera actively illuminates the scene with infrared light and utilizes a pixel array to detect the reflections. The delay between the transmitted and the received light is utilized to obtain a distance value for each pixel.

Automotive Radar Sensor
An automotive radar sensor transmits electromagnetic waves to the scene and detects the reflections. The utilized modulation principle allows the extraction of each reflecting object's range, velocity, and angle. Automotive radar sensors typically utilize wavelengths in the millimeter range, compromising frequencies between 30 GHz and 300 GHz.

Vision Camera
A vision camera can capture two-dimensional color or grayscale images of the scene. The camera projects the scene's reflections of the ambient light (e.g., from the sun) via a lens onto the image sensor. In order to control the image acquisition, vision cameras typically allow the adaption of multiple parameters (e.g., lens aperture, gain, exposure time).

Inertial Measurement Unit
An *inertial measurement unit* (IMU) is a device capable of measuring the applied specific force, angular rate, and orientation using accelerometers, gyroscopes, and magnetometers. Since measurement inaccuracies lead to a drift in the estimated position over time, inertial measurement units are typically combined with other sensors.

Robot Operating System
The *Robot Operating System* (ROS) is an open-source software framework for robotic applications. The popular framework provides fundamental software methods to enable the rapid prototyping of robotic systems. The active community contributes open-available modules, which offer solutions to common robotic tasks.

Chapter 1

Introduction

The first chapter of this doctoral thesis starts with a motivation, presenting why environmental perception for autonomous systems is still an open research field. Major limitations of current systems are addressed, and this work's approach of enhancing the performance of perception systems is introduced. The chapter lists this work's contributions to the scientific community and is closed by presenting this thesis's structure.

1.1 Motivation

Environmental perception sensors can be considered the key-enabling technology of smart devices and machines interacting with their environment. The deployment of these sensors has enabled the introduction of well-established applications, such as *adaptive cruise control* (ACC) for road vehicles. Nowadays, the technology market is widely shaped by *cyber-physical systems* (CPS), which can benefit from context information provided by perception sensors. Robots/vehicles/drones are equipped with multiple perception sensors in order to perform autonomous operations. The automotive industry, a major driver of new technology, is targeting to build fully automated vehicles in the near future, which utilize perception sensors to sense the environment.

Since no single sensor is capable of solving the perception task on its own, multiple perception sensors have to be used [1]. The combination of multiple sensor technologies shall compensate for the weaknesses and vulnerabilities of single sensors. This diversity and redundancy of sensor data are mandatory to provide robust perception data and to obtain a confident model of the environment. Safety-critical applications particularly require multiple independent inputs in order to ensure safe operation in any condition.

The first automated vehicles were presented to the public in 2005 at the DARPA Grand Challenge, a competition for automated vehicles initiated by the United States Department of Defense [2], [3]. During this event, five of the 23 participating teams were able to autonomously complete a predefined path in a desert located between California and Nevada. In 2007, at the follow-up event, the DARPA Urban Challenge, automated vehicles of multiple teams were already able to safely navigate through city streets [4], [5], [6]. The individual teams' vehicles were equipped with various sensor setups, including laser scanners, vision cameras, and radar sensors. Most competitors published the system design of their vehicle, providing research and industry with valuable references for the

development of automated vehicles and robots. However, the utilized hardware was not feasible for mass production, and the systems were not generic enough to work in any scenario.

With increasing processing power and communication bandwidth available on these systems (e.g., vehicles, smartphones, robots), new abilities to handle perception data arise. Traditional systems typically perform individual processing of data on sensor level and provide the high-level output to a centralized processing unit. This approach is only capable of performing a high-level fusion of the sensor data, limiting the system's overall perception performance. Alternative fusion strategies require the perception data from multiple sources to also be available at low-level. However, the majority of the sensor modules offered by the automotive suppliers do not provide the raw low-level data to the vehicle's processing system. One reason for this development-trend is the lack of established high-speed automotive data buses. In addition, the expertise regarding low-level sensor processing is still with the suppliers and is among their sensor modules' main selling points. Although ready-to-use, low-level capable sensor modules targeting the automotive/robotic market are making their way into the industry, they have not appeared on the open market yet.

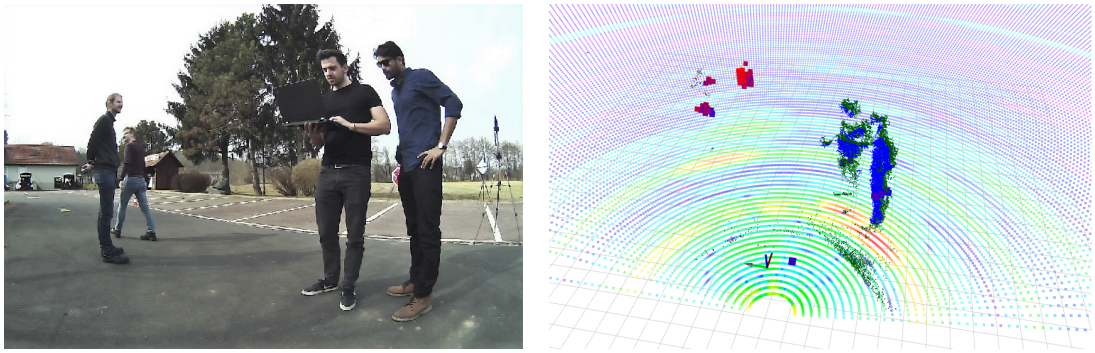
Fusing multiple sensor data streams at low-level allows the consideration of the input data's fine granularity-aspects, resulting in an increased perception performance compared to high-level fusion. If the data is aligned and properly combined, the output has increased expressiveness compared to the fused high-level data from single sensors. The fusion of heterogeneous low-level sensor data for environmental perception systems has not been sufficiently addressed by research. This thesis tackles this shortcoming by presenting an approach to make research on that topic more accessible and by providing novel approaches to perform low-level fusion.

1.2 Environmental Perception Platform

This section introduces this work's approach to building an environmental perception platform consisting of multiple heterogeneous perception sensors. The environmental perception platform enables research on the low-level aspects of multi-sensor perception systems, aiming to increase the performance of associated applications. In addition, the research questions and objectives of this work are elaborated, and an overview of the contributions is presented.

1.2.1 Overview

An environmental perception system describes a subcomponent of a system, able to sense its surroundings (e.g., smart device, vehicle). The system receives data from one or multiple perception sensors and performs the assigned (pre-) processing tasks before the data is transferred to succeeding modules for further, application-specific processing. Within the scope of this thesis, the subset of environmental perception systems targeting the deployment on vehicles and robots is explored. Since all available perception sensors have their individual shortcomings, the utilization of a single sensor type is not sufficient for safety-critical applications.



(a) Vision camera image. **(b)** Range data.

Figure 1.1: Visualization of heterogeneous sensor data, provided by this work's environmental perception system [7]. The left image depicts the output from a vision camera. The right image shows the visualization of combined range data from multiple perception sensors.

Multiple types of sensors are combined in order to build a robust perception system capable of handling the majority of situations. Various environmental perception systems have already been presented to the public. These systems utilize multiple heterogeneous sensors and fuse them in order to obtain an enhanced perception performance. Figure 1.1 shows the visualization of heterogeneous data obtained from this work's environmental perception system. The figure shows a vision camera's image and visualized data from multiple range sensors.

There is ongoing research on how to fuse the data from different perception sensors in order to provide the best-possible perception quality. Multiple research and industry institutions have built environmental perception platforms (e.g., in the form of research vehicles) based on a set of various perception sensors. A way to bypass the need of researchers to build their own hardware is to provide them with recorded datasets. Multiple research groups have published the sensor data from their perception platforms in order to encourage the development of new algorithms. Some of the available datasets became very popular (e.g., the KITTI dataset [8]) and contributed to the improvement of various perception algorithms. Since the same dataset can be used by various researchers, the results are well-suited to be compared to each other. However, since the data from the available platforms mostly consists of high-level sensor data, they only advance the development of a certain type of algorithms.

This work presents a research platform, which enables the fusion of low-level sensor data. The platform's implementation is based on the open-source *Robot Operating System* (ROS) framework [9] and can utilize data from *time-of-flight* (ToF), radar, and vision sensors. The system provides temporally and spatially aligned low-level sensor data streams, which can be utilized for further application-specific processing. The low-level interface to the sensors enables research on novel sensor fusion methods and holds the potential to improve the performance of state-of-the-art perception systems. The presented platform's capabilities are demonstrated by implementing multiple use cases (e.g., object/pedestrian detection). Researchers can utilize the proposed system architecture as a blueprint to build a perception system, which performs synchronized low-level fusion with market-available sensors.

1.2.2 Problem Statement

Researchers face a high burden when it comes to entering the field of research for low-level sensor fusion. There is only limited information available to the scientific community on how to design and implement a perception platform containing sensors with low-level interfaces. Most of the open-available automotive environmental perception sensors neither provide full access to the raw sensor data nor offer an interface to open-source frameworks like the ROS. A customized configuration of these sensor modules is also often not supported or only possible to a very limited extent. The low-level processing of the raw sensor data is typically handled by the sensor modules and isolated from the remaining system. The high-level output can then be communicated to the remaining system via a traditional automotive data bus, like the *Controller Area Network* (CAN) bus. Such sensor modules are not feasible for low-level sensor fusion approaches, which require access to the low-level data streams from a centralized module.

Due to the lack of ready-to-use sensor modules with full low-level access, research facilities are limited to available prototype versions of low-level capable sensors. Additionally, researchers face the burden of building their own platforms, consisting of multiple perception sensors, in order to perform research on low-level sensor fusion. Since building an environmental perception platform is a demanding task, low-level sensor fusion and the resulting capabilities have only been insufficiently covered by the research community. Due to the unavailability of capable perception sensors on the open market, the number of suitable datasets for developing low-level sensor fusion concepts and associated applications is strongly limited. The limited number of low-level capable datasets lack certain characteristics vital for a comprehensive study of low-level fusion strategies (e.g., time synchronization or low-level interfaces). Thus, there is the need for open platforms and open datasets capable of providing low-level sensor data to enable researchers to perform research on that topic.

The configuration and data processing of traditional perception sensors is performed by the sensor module, independent of the remaining subsystem. Thus, these systems are not aware of the robot's/vehicle's overall context and cannot adapt their configurations accordingly. Traditional perception sensors are often implemented statically and do not support any configuration adaption after the system left the factory. Some sensors perform automatic adaption of certain sensor parameters in the sensor module (e.g., automatic exposure for vision cameras). Others allow a limited adaption of the configuration parameters during runtime by enabling the adaption of certain module parameters. To sum up, traditional perception sensors do typically not provide full access to the configuration, especially not during runtime. Consequently, these sensors are not operated at their optimal configuration for a wide range of environmental context states. Perception sensors with full low-level access allow the implementation of novel concepts, which can improve the capabilities of a multi-sensor environmental perception system. The dynamic adaption of the sensor configuration, which allows the reaction to the system's current context, has not been sufficiently addressed yet.

The lack of methods to provide low-level data from multiple perception sensors to a system's centralized fusion module creates the need for additional research in this field. Novel system architectures and data-handling methods (pre-processing, timestamping) have to be developed in order to efficiently provide the low-level data to centralized modules. The

additional information contained in low-level data holds the potential to increase the performance of a wide range of perception tasks. The availability of low-level sensor data to the system's centralized processing modules enables the development of novel algorithms for various perception tasks. Existing perception algorithms have to be evaluated in order to determine whether they can benefit from the inclusion of low-level data.

1.2.3 Research Questions and Objectives

Considering the highlighted research gaps and the ability to access Infineon's state-of-the-art sensor technology, the following three research questions for this thesis were formulated:

R1 *What is a feasible design and implementation of an environmental perception platform capable of performing a low-level fusion of heterogeneous sensors?*

R2 *How can single components of a sensor fusion system be configured to obtain a satisfactory perception performance in changing environments?*

R3 *How can low-level fusion of radar and ToF data contribute to enhance the quality of state-of-the-art environmental perception systems for robotic/automotive applications?*

In order to answer these research questions, several objectives were introduced. The work is divided into three main objectives, compromising multiple tasks for each one. The achievement of these objectives is of key importance in order to determine answers to the research questions.

O1 Platform development
This objective includes the selection of feasible perception sensors and the design, implementation, and construction of a perception platform, enabling low-level sensor access. The system shall provide interfaces to allow a custom configuration of the sensors and the processing modules.

O2 Concept design
A feasible system architecture to enable the fusion of sensor data from multiple perception sensors shall be designed. This includes the development of concepts for sensor synchronization, spatial/temporal data alignment, and (pre-) processing of the measurement data. In addition, the design shall incorporate a method to dynamically adjust the system's parameters depending on the current context.

O3 Use case design and evaluation
Based on the low-level sensor data provided by the base system, multiple use cases shall be implemented. Each of these use cases solves a particular perception task by fusing and processing the available data streams. The implemented use cases are intended to highlight the benefits of including low-level data compared to existing high/mid-level approaches.

Figure 1.2 shows an overview of the conducted work throughout this thesis. The research questions are assigned to the major objectives, required to provide answers to the research questions. These objectives are then further assigned to the contributions of this thesis.

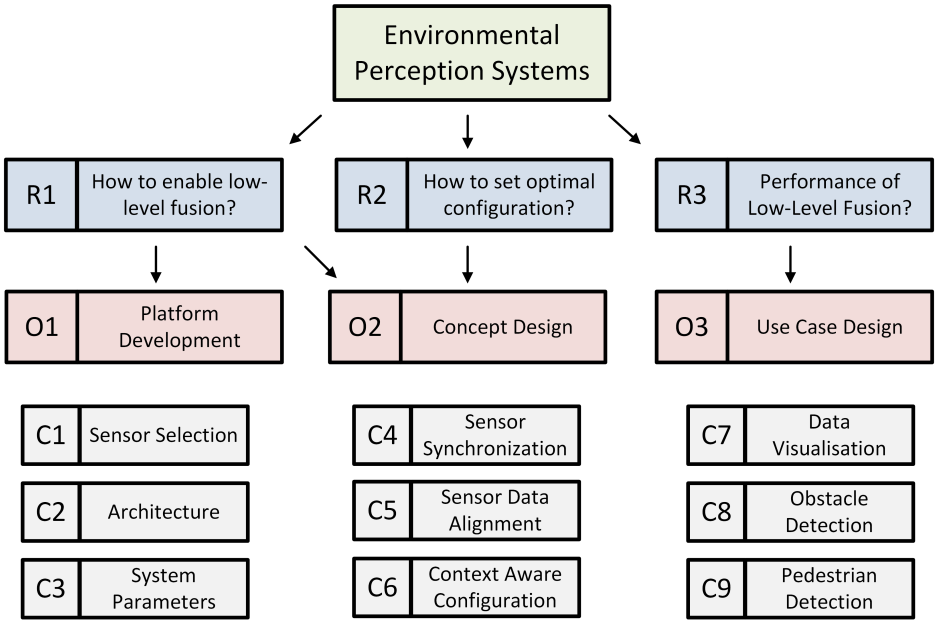


Figure 1.2: Subcategorization of the research questions, the objectives, and the assigned contributions of this thesis.

1.2.4 Contributions

The perception platform built as part of this work is similar to perception systems used in today's research vehicles and robots. The platform consists of open-available perception sensors, including ToF, radar, and vision sensors, which allow the perception of the environment. The ROS-based architecture utilizes the low-level access to these sensors and prepares the data streams for later fusion. This includes the spatial/temporal alignment, the (pre-) processing, the sensor synchronization, and the system partitioning. The presented work provides solutions to the arising challenges during the design and the construction of similar systems.

The presented platform's capabilities are demonstrated with the implementation and evaluation of several use cases. The designed environmental perception system implements low-level interfaces to multiple perception sensors, which allow dynamic reconfiguration during operation. This ability holds the potential to significantly improve the resulting perception quality compared to the mostly static systems deployed nowadays. The use cases utilize the structured data streams provided by the base perception system in order to perform application-specific perception tasks.

The conducted work to achieve the objectives and to answer the research questions resulted in a significant number of contributions to the scientific community. The main contributions generated during the work on this thesis are listed below.

C1 Sensor selection
The work presents the final selection of the utilized perception sensors (radar, ToF camera, vision camera), which are able to provide the required functionality. Additionally, a detailed description of the sensors' working principles and the raw data processing steps of radar and indirect ToF sensors are presented.

C2 Architecture
The ROS-based system architecture is capable of providing structured sensor data at multiple abstraction levels for further use-case-dependent fusion. The architecture includes low-level interfaces to the perception sensors and modules for raw data reception, spatial/temporal alignment, (pre-) processing, and reconfiguration.

C3 System parameters
The behavior of the system's modules can be customized to the expected environment via system parameters assigned to the system's modules at startup. Since these parameters control the sensor configuration and the (pre-) processing parameters, they have a high impact on the platform's performance.

C4 Sensor synchronization
The work introduces an approach capable of performing synchronized sensor measurement acquisition and accurate timestamping. An external microcontroller is used to generate a common trigger pulse for all perception sensors. Additionally, accurate measurement timestamps are estimated and assigned to the data streams when received by the processing system.

C5 Sensor data alignment
A method to obtain the transformations between the single perception sensors is presented, enabling the transformation of the heterogeneous sensor data into a common coordinate frame. This includes the estimation of the relative sensor alignments between the individual sensors' mounting positions.

C6 Context-aware sensor configuration
The system architecture implements the functionality to adapt the sensor configuration and the system parameters during runtime. This enables the dynamic adjustment of the system's perception capabilities, depending on the currently perceived context. Additional input sensors can be utilized to enhance the system's context awareness (e.g., rain, light, vehicle velocity, environment type).

C7 Distance data visualization
One implemented use case handles the visualization of sensor data from multiple perception sensors in a common coordinate system. Since the raw sensor data is often not directly visualizable, custom pre-processing is applied in order to obtain a visualizable format.

C8 Obstacle detection
Another use case employs the low-level data streams from multiple perception sensors and fuses them into a common occupancy grid. The occupancy grid is then used to obtain a better understanding of the system's surroundings and to identify obstacles in the environment.

C9 Pedestrian detection
The thesis presents a method to perform pedestrian detection, utilizing the structured data streams provided by the system's base perception system. The heterogeneous data streams are fused into a common representation and utilized to perform an enhanced detection of pedestrians in the sensors' common field-of-view.

1.2.5 Outline

This cumulative doctoral thesis is structured into the main part (Chapter 1-7) and a collection of the publications carried out as part of this thesis (Chapter 8). A state-of-the-art overview of environmental perception systems in general and an introduction of related work for selected applications are presented in Chapter 2. The fundamental working principles of radar sensors, ToF cameras, and vision cameras are described in Chapter 3 to provide the reader with the background knowledge required to understand the main part of this thesis. Chapter 4 outlines the requirements of the desired perception platform and introduces the system's architecture. The selected perception sensors are presented, and the hardware and software design of each subsystem is introduced. Implementation-specific details of the ROS-based software architecture and the hardware setup are described in Chapter 5. The results of the work, including real-world measurement data and performance metrics, are presented in Chapter 6. Chapter 7 provides a conclusion of the performed work and presents possible future directions based on the outcome of this thesis.

Chapter 2

Related Work

Since the fusion of data from multiple perception sensors is a common task in various fields (e.g., robotics, vehicles, smartphones), there already exists a vast amount of published work in this field. This chapter starts with the presentation of open-available environmental perception platforms (e.g., research vehicles/robots), open datasets, and their shortcomings. Existing approaches to perform spatial and temporal alignment of heterogeneous perception sensors are introduced as well as approaches to fuse the corresponding data. Finally, multiple perception applications are introduced, which could benefit from the utilization of low-level data streams from heterogeneous perception sensors.

2.1 Environmental Perception Platforms

The current research efforts, targeting the utilization and fusion of multiple environmental perception sensors, are mainly driven by the development of automated vehicles. A number of prominent universities (e.g., Stanford University, Carnegie Mellon University, Massachusetts Institute of Technology) and industrial companies (e.g., Google, Tesla) conduct research on automated vehicles [10]. Many of these institutions have built their own research vehicles, including the development of the environmental perception system.

This section starts with a short overview of the current path of automated vehicles and introduces some of the remaining challenges. Currently, research is typically performed using self-constructed hardware platforms (e.g., research vehicles) or utilizing open-available datasets. Thus, multiple research vehicles which publicly provide construction and implementation details are presented, as well as open datasets for automated driving research. The limitations of currently available open datasets and platforms are presented, showing the need for novel physical hardware platforms. Different approaches to solve the arising challenges when building such a system are introduced (e.g., calibration, synchronization).

2.1.1 Automated Driving Research

Since their introduction, the reliability and confidence of *automated driving systems* (ADS) have highly increased. Multiple institutions have built prototypes of automated vehicles, which successfully demonstrated their capabilities in various environments and traffic scenarios. However, as accidents with current research vehicles prove, the systems still lack reliability in harsh environmental conditions and fail in various edge cases [11].

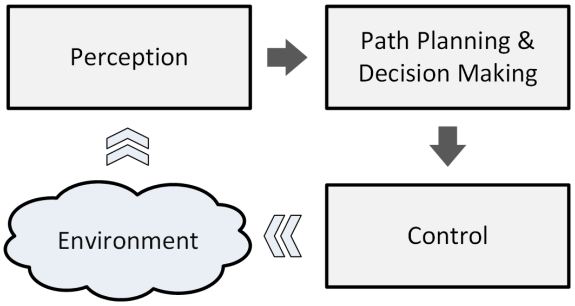


Figure 2.1: Major subsystems and processing flow of automated driving systems, adapted from [14].

There is no blueprint yet available defining the optimal set of market-available components (sensors/processing units) enabling fully automated vehicles. This is why research groups utilize different sets of perception sensors (e.g., radar, lidar, cameras), mounting strategies (e.g., front-facing, surround-view), processing units (e.g., GPU-based vs. CPU-based, centralized vs. distributed), and even different types of vehicles (e.g., golf carts, passenger vehicles, trucks). When considering the most successful approaches of research and industry up to the current date, certain trends can be observed.

The research vehicles utilize proprioceptive and exteroceptive sensors. Proprioceptive sensors provide the internal state of the vehicle (e.g., *Global Navigation Satellite System* [GNSS], wheel encoders), while exteroceptive sensors sense the vehicle's surroundings (e.g., camera, radar, lidar). While most groups' environmental perception systems consist of vision, radar, and lidar sensors, other groups try to solve the perception task solely using vision and radar sensors.

Some research platforms utilize centralized system architectures, while others are built upon distributed system architectures [12]. Within a centralized architecture, the majority of processing is done on a single processing unit, while the processing load is distributed onto several units in a distributed system. In general, high-performance computation units are used to execute computationally expensive tasks (e.g., sensor data processing, detection/tracking). Time-critical tasks (e.g., vehicle control) are performed on real-time capable *Electronic Computation Units* (ECUs) [13]. While earlier prototypes focused on using multiple distributed units for high-performance tasks, the current trend leads towards single-board supercomputers for that task (e.g., the Nvidia Drive AGX platforms[1]). Nvidia's platform includes a safety-focused microcontroller[2] responsible for the safety-critical tasks and running system diagnostics. These supercomputers are centralized all-in-one solutions, capable of executing multiple deep neuronal networks and real-time tasks in parallel. They provide direct interfaces to the sensors as well as to the vehicle's buses.

The same holds true for the software implementation of ADS. Since no flawless solution is yet available to the public, different institutions of research and the industry have come up with varying approaches. Most of the publicly available software architectures are structured as modular systems, which utilize multiple processing modules between the sensors and actuators. Figure 2.1 shows an illustration of the main processing blocks of

[1] Nvidia Drive: Autonomous Vehicle Development Platforms (`https://developer.nvidia.com/drive`).
[2] Infineon AURIX™(Automotive Realtime Integrated NeXt Generation Architecture).

ADS and their interaction with the environment. The modules can be roughly assigned to three main categories: perception, path-planning/decision-making, and control [15]. These three categories, including sort descriptions, are listed below.

- **Perception**
  The modules of this category sense information about the vehicle's surroundings and its current internal state. This includes the sensor data's reception, processing, and combination in order to get a semantic understanding of the environment.

- **Path-planning/decision-making**
  The vehicle plans its future actions and trajectories depending on the perceived environment and the overall navigation goal.

- **Control**
  Longitudinal (throttle/brake) and lateral (steering) control commands are sent to the vehicle's actuators in order to follow the planned trajectories and perform the requested actions.

In addition to modular systems, end-to-end architectures are a relatively new approach to implement the software system of automated vehicles without the need of intermediate modules. These systems fully rely on deep neuronal networks, which generate the actuator outputs directly from the sensor input [11]. However, although end-to-end systems are capable of performing well in known, pre-trained scenarios, they might lead to unsafe behavior in unconventional scenarios (not trained). Thus, safety-focused systems usually rely on neuronal networks for subtasks only (e.g., parameter estimation, classification) instead of using them as end-to-end systems.

One of the key requirements of ADS is the high quality and the robust perception of its surroundings. Since this work's main focus is on perception systems, the perception subsystem of ADS is further analyzed in this section. The main responsibilities of the perception subsystem include the completion of the following tasks:

- **Localization**
  Self-localization is performed using the vehicle's proprioceptive sensors (e.g., GNSS, wheel odometry, *inertial measurement unit* [IMU]) for coarse positioning, as well as its exteroceptive sensors for localization within an a-priori map (e.g., via point cloud matching).

- **Object detection/tracking**
  This task includes the detection of static and dynamic objects of interest (including road users, traffic lights, or road signs). The detection is commonly performed with a lidar sensor, while the vision camera is utilized for the classification task. The movement of dynamic objects shall also be tracked over time.

- **Drivable regions detection**
  This task includes the detection and localization of the road, the lanes, and the drivable space. Additionally to the detection, a semantic understanding of the road lanes and their connections is required (e.g., intersection lanes).

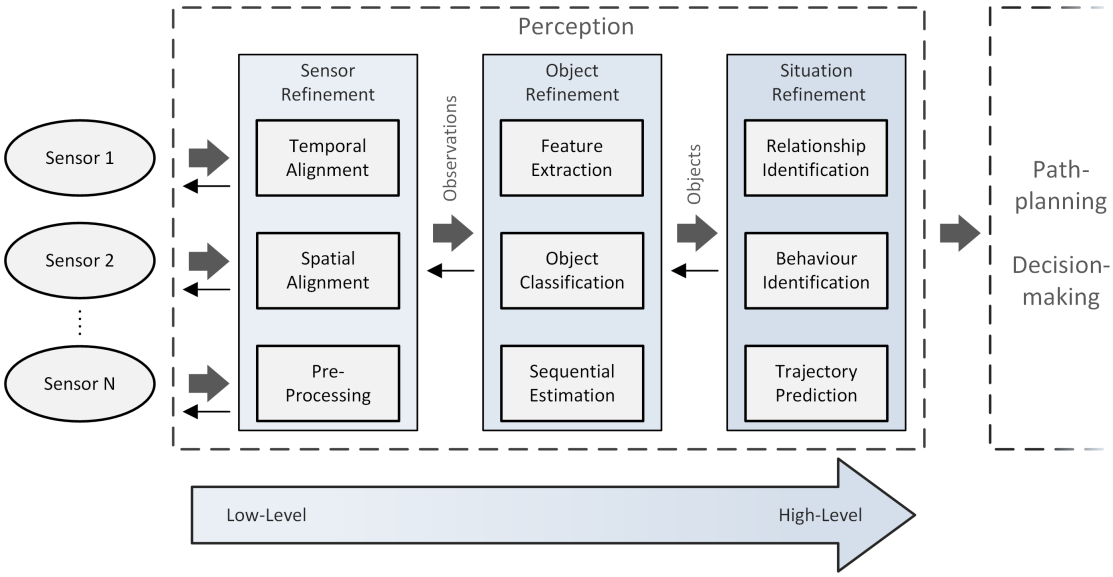


Figure 2.2: Different levels of perception for automated driving systems, adapted from [15].

To accomplish the perception tasks, the sensor data is received from the sensors, fused, and processed. The path-planning subsystem then utilizes the output of the perception module(s) for further manipulation. Figure 2.2 shows the different levels of perception from low-level to high-level. As seen in the illustration, the perception tasks can be classified into different levels. The lowest level deals with the reception of the raw data, while the highest level already deals with tracked and classified objects and their interactions with the environment. This work mainly focuses on the lower levels of the perception subsystem, the raw data reception, and the sensor refinement. The third level already deals with objects, which is beyond the low-level scope of this thesis.

Traditional software architectures of ADS combine multiple modules, with each module implementing a subtask of the corresponding overall perception task. However, since newly arising approaches rely more heavily on neuronal networks at different system levels, the traditional separation of the system's single modules starts to fade. Conventional algorithms often make use of hand-crafted heuristics, introducing a significant level of complexity. Neuronal networks are a popular tool capable of outperforming conventional approaches for certain (sub)tasks. These networks are capable of performing certain tasks automatically but require intensive training with pre-labeled data.

As stated by the authors of [14] and [16], there are still several remaining challenges on the way to fully automated driving. These include the operation in difficult weather/light conditions and complex environments (e.g., construction sites, urban streets). Current systems often rely on a-priori information (e.g., high-definition maps) to work properly in complex environments. These limitations are a result of the still insufficient perception performance of today's ADS. Reasons are the limited quality of the available sensor data and the limited capabilities of the state-of-the-art perception algorithms. Increasing the capabilities of the perception subsystem could highly contribute to solving the remaining challenges. Additionally, novel sensor setups and new sensor processing concepts have to be evaluated in order to pave the way to more robust systems.

2.1.2 Existing Perception Platforms and Datasets

The industry and academia target to improve the capabilities of today's perception systems. The development of novel algorithms is generally an iterative process, requiring a repetitive evaluation of the associated perception task(s) in real-world scenarios. In order to perform these evaluations, researchers can either utilize data from their own research vehicles or from open-available datasets. A number of research institutes and companies within the industry have built their own research vehicles in order to enable them to capture data independently. This enables full control of the sensor selection/arrangement, the data handling, and the pre-processing steps. However, a research vehicle's construction is a tedious task, requiring expert knowledge and a sufficiently high budget.

On the other hand, utilizing open-available datasets opens the doors for smaller groups and individual researchers to conduct research in the field of automated vehicles without the burden of building their own platforms. The datasets allow the implementation of novel processing algorithms (e.g., deep learning-based methods) utilizing the open-available data as input. Additionally, the datasets allow a quantitative comparison of newly developed approaches to the best-performing algorithms for a particular task (e.g., object detection performance). The downsides of the open-available datasets are the restriction to the respective sensor selection/arrangement, the inability to influence the recorded scenarios, and the predefined low-level data handling.

Software simulators represent a completely different approach to automated driving research [17]. These simulators are capable of emulating the environment and the vehicle's acquired sensor data, providing researchers the option to bypass the building process of a physical platform However, although the perception capabilities of simulators have improved significantly, they cannot fully replace a real-world platform with physical sensors. A physical platform experiences real-world effects and sensor imperfections, which today's simulation models cannot recreate. At its current development stage, the simulation data can be used as an additional source to train or test algorithms but cannot entirely replace the acquisition of real-world data.

This section presents a number of existing environmental perception platforms and open datasets, utilizing multiple perception sensors. The knowledge of similar approaches is essential to utilize the strengths of existing systems and to identify their restrictions. The section starts with the introduction of multiple environmental platforms utilized in research vehicles. Publicly available datasets are presented, which contain recorded perception data from multiple sensors in various scenarios.

Existing Research Platforms

Multiple institutions of industry and academia have published details about the hardware and software architecture utilized in their research vehicles. These research vehicles often contain a high number of expensive perception sensors, requiring high processing capabilities. Even though these vehicles are not feasible for mass production, they help to find the crucial sensors and processing modules to achieve robust perception. As algorithms improve and the prices of capable perception sensors decrease, multi-sensor environmental perception units can be expected to find their way into future mass-produced vehicles. Table 2.1 lists a set of selected research vehicles presented to the public within the last two decades. The table compares selected features of these vehicles' perception systems with

Table 2.1: A custom selection of recently published architectures of research vehicles and their respective support of certain features.

	Vision camera	Lidar	Radar	Low-level access	Synchronized sensors	Parameter adaption	Software details	Hardware details
Boss [5]	✓	✓	✓	✗	✗	∽	✓	✓
Junior [4]	✓	✓	✓	✗	✗	✗	✓	✓
CMU vehicle [18]	✓	✓	✓	✗	✗	✗	∽	✓
Bertha [19]	✓	✗	✓	✗	✗	✗	✓	∽
Autoware [20]	✓	✓	✗	✗	✗	✗	✓	✗
MIT vehicle [21]	✓	✓	✗	✗	✗	✗	∽	✓
Apollo	✓	✓	✓	✗	✗	✓	✓	✓
Zeus [22]	✓	✓	✗	✗	✗	✗	✓	✓
IARA [10]	✓	✓	✗	✗	✗	✗	✓	✓
Nvidia	✓	✗	✓	∽	∽	✗	✓	✓
This thesis	✓	∽	✓	✓	✓	✓	✓	✓

the presented platform of this thesis. Insufficiently documented features are considered as not available in this comparison.

In 2007, at the DARPA Urban Challenge, automated vehicles were targeted to drive through an urban environment. The research vehicle *Junior* from the Stanford University and *Boss* from the Carnegie Mellon University successfully completed the challenge and placed second and first. Details about the architectures of these two vehicles were published in [5] and [4]. Boss is a modified 2007 Chevrolet Tahoe, which utilizes multiple lidar sensors, radar sensors, and 2D cameras as perception sensors. The implemented perception system is capable of performing moving obstacle detection and tracking by fusing radar/lidar/camera, as well as static obstacle detection using the lidar sensors only. The vehicle utilizes a configuration library capable of adapting task parameters during runtime. Junior, a modified 2006 Volkswagen Passat, is equipped with a similar set of perception sensors (radars, lidars, cameras). The vehicle mainly relies on its Velodyne 64-beam spinning lidar sensor for static and dynamic obstacle detection. Boss and Junior utilize the publisher-subscriber principle to communicate between the single modules. Both publications provide a valuable high-level overview of the vehicles' architectures but lack the description of low-level details (e.g., sensor synchronization, calibration).

The A1 research vehicle, winner of the 2012 *Autonomous Vehicle Competition* (AVC) in Korea, is presented in [23]. The vehicle utilizes multiple laser scanners, cameras, an IMU, and GNSS sensors. The authors show the software's implementation on distributed computation units, including two industrial computers and 13 ECUs. They provide an overview of the utilized software architecture and provide detailed information regarding

the communication between the distributed units (via the FlexRay bus). Since their presented approach focuses on traditional automotive buses (FlexRay, CAN) with relatively low bandwidths, the system can only fuse perception data at higher levels. The authors do not provide details about the sensor synchronization/calibration and the structure of the sensor interfaces.

In 2013, scientists from the Carnegie Mellon University converted a Cadillac SRX to an automated driving research vehicle [18]. The authors provide a detailed description of the conversion of a regular passenger vehicle to enable drive-by-wire control. The vehicle's perception suite utilizes six lidar sensors, six radar sensors, and two vision cameras. The processing part consists of four small-scale-factor computers (Intel QX9300s, Nvidia GT530), cooled via the vehicle's air conditioning system. The vehicle's software framework is based on the same framework, as introduced by Boss in the DARPA Urban Challenge 2007 (Tartan Racing Urban Challenge System [24]). However, even though the authors describe the vehicle's modifications for the drive-by-wire conversion, the publication does not provide low-level details about the perception sensor setup. Due to the lack of an official name, the research vehicle is listed as *CMU vehicle* in Table 2.1.

Researchers from Daimler introduced *Bertha*, a slightly modified version of a Mercedes-Benz S-Class in 2013 [19]. The series-production vehicle was extended with *close-to-production sensors* in order to prove the capability of autonomous operation on public roads. The vehicle's perception setup consists of two wide-angle vision cameras, a stereo camera setup, and multiple radar sensors. In 2015, the authors of [25] further extended the vehicle's perception performance by enhancing its radar capabilities (eight radar sensors). They utilized Bertha's improved radar setup to perform tasks like generating a grid map of the static environment and classification of pedestrians. The published work does, however, not provide details about the calibration steps. Further, Berta deploys standard automotive radar sensors, which only provide pre-processed data to the remaining system.

The Autoware foundation[3] claims to be the first provider of an all-in-one software solution for automated vehicles. The first version of their software solution for automated vehicles is presented in [20], which utilizes components available on the open market. The lidar and camera-based software solution can be utilized to develop algorithms for ADS. The software platform *Autoware* is based on the ROS, the *Point Cloud Library* (PCL), OpenCV, and other open-available frameworks and libraries. In a more recent publication, the creators of Autoware show their approach to run the framework on embedded systems [26]. Autoware includes various modules which contribute to the development of automated vehicles (e.g., localization, detection, planning). The software supports various types of vision, lidar, IMU, and GNSS sensors as input for the perception module. Radar sensors have not been integrated yet. Autoware can also be used with a number of compatible drive-by-wire vehicles, utilizing an external controller to transfer the requested velocity and steering values to the vehicle's ECUs. The software does not provide details about the temporal and spatial alignment of the sensor data, nor the low-level interface to the sensors.

Another approach to building an automated driving research platform was published by researchers of the Massachusetts Institute of Technology [21]. The authors converted a 2015 Toyota Prius V into an automated driving research vehicle. The presented work

[3]The Autoware Foundation (`https://www.autoware.org`).

provides details of the conversion to drive-by-wire control. The stock vehicle was extended with relatively inexpensive sensors to be used for automated driving. The perception system includes four 2D lidars, a USB camera, and an IMU/GNSS combination for global positioning. A notebook is utilized as the system's computation platform, running software for perception (localization, object detection), planning, and control. The research vehicle is listed in Table 2.1 as *MIT vehicle*.

Another full open-source hardware/software stack, the *Apollo Open Platform*[4], is made available by *Baidu*. The software can be used with compatible drive-by-wire vehicles and includes software modules for various tasks of automated driving. The platform supports a number of market-available hardware components that can be utilized to equip a vehicle for automated driving. This includes an industrial-grade computer, radar/lidar sensors, vision cameras, and GNSS receivers. In its first versions, the software stack was built upon the ROS framework, but has recently moved to the *Apollo Cyber RT* framework, which is more feasible to meet the requirements of automotive systems. The software framework includes a calibration module capable of determining the sensors' intrinsic and extrinsic parameters. Since the Apollo Open Platform utilizes generic sensor data as input, it is not directly capable of processing low-level sensor data and does not provide a low-level interface to the sensors. Even though the platform is open source, it lacks proper documentation of the utilized modules and their interactions.

Zeus is an automated driving research vehicle built by researchers of the University of Toronto [22]. The team placed first in the *Society of Automotive Engineers* (SAE) AutoDrive Challenge[5] in 2018 and 2019. Based on a 2017 Chevrolet Bolt, Zeus is equipped with multiple perception sensors in order to perform basic automated driving tasks. The perception sensors include a lidar sensor, a vision camera, a stereo camera, and a GNSS/IMU combination. The computation system consists of two Intel Xeon processors with 22 cores each, running a ROS-based automated driving software. The authors present the implementation of multiple algorithms, including lane detection, stop sign detection, and static obstacle detection.

The Federal University of Espírito Santo in Brazil presents the architecture of their *Intelligent Autonomous Robotic Automobile* (IARA) research vehicle in [10]. Based on a 2011 Ford Escape Hybrid, the research vehicle is equipped with multiple perception sensors, including a stereo vision camera, a long range lidar, and a 360° spinning lidar. The computation platform consists of two six-core processors and an Nvidia GPU capable of running various automated driving algorithms. Similar to the other presented approaches, the publication provides only very limited details about the low-level handling of the sensor data.

The Nvidia corporation offers a reference platform, the *DRIVE Hyperion*[6], as a base to enable the development of automated driving algorithms. The Hyperion developer kit is built around Nvidia's in-vehicle supercomputing platforms capable of performing the demanding software tasks of automated vehicles. The reference kit includes multiple sensors (eight cameras, eight radars, IMU, and GNSS) and a drive-by-wire interface in order to be used in compatible vehicles. Nvidia also provides a multi-level software stack for automated driving that can be used with the reference vehicle and supports a variety

[4]Apollo: Open Source Autonomous Driving (`https://apollo.auto/`).
[5]SAE AutoDrive Challenge (`https://www.autodrivechallenge.com`).
[6]Nvidia Drive: Hyperion Developer Kit (`https://developer.nvidia.com/drive/drive-hyperion`).

Table 2.2: Custom selection of open-available datasets for research on automated driving systems and an overview of their main features.

	Vision camera	Stereo vision	High-level radar	Low-level radar	2D lidar	3D lidar	GNSS/IMU	Labels	Platform details	Dataset available
KITTI [8]	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
RobotCar [27]	✓	✓	✗	✗	✓	∽	✓	✗	✓	✓
nuScenes [28]	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓
H3D [29]	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓
Argoverse [30]	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
Astyx [31]	✓	✗	✓	∽	✗	✓	✗	✓	✓	✓
OLIMP [32]	✓	✗	✓	✓	✗	✗	✗	✓	✓	✓
WAYMO [33]	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓

of other perception sensors (e.g., multiple lidars). The software stack includes sample implementations of the different modules required for automated driving (e.g., mapping, perception, planning). These modules are mainly based on *deep neuronal networks* (DNN) (e.g., for obstacle/lane detection) in order to demonstrate the strengths of the Nvidia platform. The well-documented software framework includes solutions for tasks such as sensor calibration, accurate timestamping, efficient sensor data processing, and estimating the vehicle's pose. However, the ecosystem is tightly bound to the use of Nvidia hardware and might be too bulky and restricted to perform research on low-level data handling and low-level interactions with sensors.

Open-Available Datasets

Open-available datasets are a common tool to support the development of new perception algorithms. The performance of new software modules for ADS is commonly evaluated using multiple datasets [11]. Labeled datasets can be utilized to train and evaluate machine learning algorithms and to determine the performance of a detection task. Table 2.2 lists a selection of open datasets available to the scientific community and compares certain characteristics of these datasets. The datasets were introduced in associated publications, providing information about the composition, the desired purpose, and other related details. Some of the publications also include details about the utilized perception architecture, the data handling, and the utilized approach to record the data. These publications are a valuable source of information when designing a perception platform based on similar sensors.

The popular KITTI dataset provides sensor data from multiple perception sensors in various driving scenes recorded on German roads [8]. The utilized Volkswagen Passat station wagon was extended with a rooftop-mounted sensor rig, including a laser scanner, four vision cameras (two stereo setups), and a GNSS/IMU system. The authors address

the synchronization and calibration of the sensors in order to provide aligned data. In order to allow researchers to compare their computer vision algorithms based on the KITTI dataset, there exists an open benchmarking service on the project's website. The downside of the KITTI dataset is the limited 10 Hz frame rate and the lack of radar data.

The authors of [27] built the *Oxford RobotCar* platform, a modified Nissan LEAF for automated driving research. The vehicle was used to record the Oxford RobotCar dataset, containing perception data from a route in central Oxford. The platform is equipped with a stereo camera, three vision cameras, two 2D lidars, one 3D lidar, and a GNSS/IMU system. Two Intel Xeon processors with eight cores are used as computation modules, while solid-state drives are utilized for fast data storage. The associated publication includes information about the synchronization and calibration of the sensors. Similar to the KITTI dataset, the RobotCar recordings are open-available and have been used extensively by researchers worldwide.

The *nuScenes* dataset for research on automated driving was introduced in 2019 [28]. This dataset contains perception data from a typical sensor setup of automated vehicles and claims to be the first open dataset that includes radar data. The data was recorded with a Renault Zoe, equipped with six vision cameras, five radars, one lidar, and a GNSS/IMU solution. The authors provide valuable information about the composition of the sensor setup and the utilized approach to perform sensor calibration and synchronization.

The H3D dataset was published by the Honda Research Institute, providing perception data of crowded urban scenarios [29]. The authors provide researchers a time-synchronized and calibrated dataset for 3D multi-object detection and tracking. The vehicle utilized to record the dataset was equipped with three vision cameras, a 3D spinning lidar, an IMU/GNSS sensor, two eight-core processors, and solid-state drives for data storage.

The *Argoverse* dataset is provided by the automated driving company Argo AI [30]. The dataset's main goal is to support research in the field of 3D tracking and motion forecasting. The recording vehicles were equipped with two 3D spinning lidars, seven vision cameras for 360°vision, a stereo camera setup, and a GNSS sensor. The authors claim to provide synchronized data streams and include calibration data for the sensors.

A radar-centric platform, utilized to record the *Astyx* dataset, is presented in [31]. The sensor setup consists of a high-resolution radar sensor, a vision camera, and a 3D lidar sensor. The dataset's associated publication contains information about the calibration procedure and states that the dataset contains synchronized data. However, the dataset only contains a very limited number of scenarios (about 500 frames), and the sensors are operated at dissimilar frame rates.

The authors of [32] present the OLYMP dataset, an open-source dataset for advanced environment perception. The utilized sensor platform is equipped with a vision camera, an *ultra-wideband* (UWB) radar, a typical automotive radar, and a microphone. However, the data acquisition of the different sensors is not performed simultaneously. The corresponding measurement data is selected using the closest-matching reception timestamp.

Waymo, the succeeding company of the Google self-driving car project, released a large and diverse dataset for automated driving in 2020 [33]. The sensor setup consists of five lidar sensors and five vision cameras. The dataset consists of precisely synchronized and calibrated camera and lidar data. Each point of the lidar data contains a vehicle pose in order to address the varying poses between the single measurements.

2.1.3 Spatial Sensor Data Alignment

The raw measurements from the different perception sensors are available in sensor-dependent frames, defined by the respective sensor's mounting position and the sensor's field-of-view In order to combine data streams from multiple perception sensors, the spatial relationship between the respective frames has to be known. The static transformations between the rigidly mounted sensors are typically determined during a separate calibration procedure of the perception system. Traditional, manual approaches require multiple measurements in specific environments (e.g., with targets/markers) to estimate these transformations. Other calibration approaches allow marker-less calibration, exploiting features of the natural environment. Knowing the transformation between the individual sensor frames, the measurement data can be transformed into a common coordinate system for further processing.

In general, the sensor calibration procedure can be divided into intrinsic and extrinsic calibration. While the intrinsic calibration targets the compensation of sensor non-linearities (e.g., distortion), the external calibration is utilized to determine the relative poses of the sensors to each other or to a common reference. The intrinsic calibration is often already performed by the manufacturer and provided to the customer. In order to obtain accurate frame transformations, the extrinsic calibration shall be performed with sensor data, which already considers the corresponding intrinsic parameters.

A well-researched topic in that field is the calibration of multi-camera setups, a fundamental requirement to enable stereo vision. Traditional approaches utilize point correspondences of calibration patterns to estimate the transformation between the sensors. Other approaches utilize natural features available in the environment. The work presented in [34] utilizes street signs to obtain the extrinsic calibration of the multiple vision cameras mounted on the vehicle. The authors of [35] introduce a method to perform extrinsic calibration of multiple vision cameras mounted on a vehicle, utilizing the natural features in the environment and wheel odometry.

The utilization of heterogeneous sensors (e.g., lidar, radar, vision camera) in modern perception systems requires additional calibration concepts. The extrinsic calibration of a sensor pair usually involves the detection of feature points (e.g., markers, edges) in the scene, located in both sensors' common field-of-view. In order to obtain easily detectable and distinguishable features, this is typically done by placing artificial targets into the scene (e.g., 3D targets). The authors of [36] present a custom target design capable of being uniquely detected by radar, lidar, and vision camera. After performing multiple measurements of the target, placed at different positions, the relative pose between the sensors can be estimated. The authors of [37] describe a fast way to perform the extrinsic calibration of a lidar sensor and a vision camera using a checkerboard pattern. The authors of [22] utilized this method for the calibration of a research vehicle's perception sensors.

2.1.4 Temporal Sensor Data Alignment

Another important aspect of a multi-sensor platform is the accurate synchronization of the sensor data in the time domain. In order to allow spatial and temporal alignment of multiple sensor data streams, the measurement timestamps have to be precisely known. The relevant timestamp associated with a sensor's raw data stream is the measurement acquisition time, not the processing system's reception time. The sensor data's temporal

alignment is crucial when considering dynamic effects, such as the vehicle's ego-movement or moving objects in the scene. Thus, precise temporal alignment is inevitable for high-level perception tasks such as environment mapping or moving object tracking.

Simultaneous data acquisition, i.e., multiple perception sensors which acquire their measurements at the exact same time, is beneficial for low-level fusion tasks. The simultaneous acquisition eliminates the uncertainty introduced by scene changes due to different acquisition times. Non-simultaneous measurements are vulnerable to dynamic scene changes caused by moving objects or an ego-movement of the vehicle. While the effects of a vehicle's ego-motion can be compensated in certain cases, moving objects cause deviations in non-simultaneously acquired measurements.

However, simultaneous data acquisition has its limits in real-world sensor systems. A measurement acquisition requires a certain amount of time to utilize the desired physical effects to obtain the measurement data. Thus, a scene might be subject to change during the measurement acquisition (e.g., resulting in motion blur). Perfectly simultaneous data acquisition would additionally require equal measurement durations. This is not feasible for most multi-sensor perception systems since the required acquisition time depends on both the sensor type and the environment. Depending on the expected dynamics of the desired target environment, a tolerable deviation in the acquisition times can be defined. In the context of ADS, the dynamic behavior of the expected traffic participants is small enough to neglect a deviation in the range of single milliseconds for most applications.

The authors of [38] show an approach to analyze the timestamps of perception sensors. They utilize custom real-time capable hardware in order to detect the exact measurement acquisition timestamps (e.g., using a photodiode for lidar). This timestamp is utilized to determine the deviation from the sensor's provided timestamp (if available) or from the arrival time at the processing system (software timestamp). A proper analysis of the perception sensors' real timestamps can be utilized to compensate static deviations and to provide temporally well-aligned data streams.

The authors of [39] address the issues caused by unsynchronized sensors and show a method to mitigate the problem. In [40], the same authors present an approach to perform temporal sensor synchronization in the context of automotive vehicles and *advanced driver assistance systems* (ADAS). They also describe the effects caused by frame drops (lost frames) in the data streams and show a way to cope with them.

A common way to accurately synchronize multiple clocks within a local network is the *Precision Time Protocol* (PTP). The PTP requires dedicated hardware support and is able to achieve a synchronization precision in the range of single microseconds. The authors of [41] show an approach where they used a PTP capable microcontroller to obtain time-synchronized measurements from multiple perception sensors. However, the protocol is not applicable to most consumer sensors that utilize non Ethernet interfaces (e.g., USB 3) or do not come with native PTP support.

The sensors of the RobotCar platform [27] are synchronized using the *TICsync* library [42]. This library provides an efficient method to synchronize multiple distributed clocks within a few seconds up to milliseconds accuracy. The work published in [8] presents the sensor synchronization method used for recording the KITTI dataset. The platform's vision cameras are synchronized with the lidar sensor by triggering the acquisition when the spinning lidar crosses the forward-facing position. Since the GNSS/IMU data cannot be synchronized that way, the messages with the closest receive timestamps are used.

2.1.5 Context-Aware Sensor Configuration

Automated vehicles are intended to be deployed in various conditions (i.e., sun, snow, daytime, nighttime) and contexts (i.e., highway, urban streets). The sensors and the processing algorithms shall adapt their configuration to the context in order to provide the best possible output in any scenario. The basic concept of changing sensor parameters during runtime is presented in [43], introducing the term *active perception*. In order to enable context-aware parameter modifications, internal sensor/processing parameters have to be made available for dynamic adaptions. Even though automatic parameter adaption holds a high potential, many of today's perception systems are still operated in a partially static way.

There are different approaches known to the research community to adapt parameters dynamically. Traditional systems often require users or experts to manually adjust the static system parameters when deployed in a new target environment. Modern approaches utilize system knowledge, optimization techniques, or learning approaches to perform automatic parameter adaption [44]. Prominent examples of automatic parameter adaptions are a digital camera's exposure time and its focus distance.

One way to perform automatic parameter selection is to utilize system knowledge made available via an earlier built *knowledge base* (KB). A KB can be manually defined at design time by utilizing expert knowledge and heuristics to define a fixed set of rules. In [45], a KB and a program supervisor are used to perform rule-based parameter tuning for road obstacle detection. A KB can also be built during a training phase by exposing a system to various environments [46]. Satisfactory parameters are then either determined automatically (optimization techniques) or by manual human intervention. During runtime, the system adapts the parameters based on the KB entries of similar context states.

The authors of [47] show an approach that utilizes an offline learning phase and online parameter tuning for object tracking in video scenes. Offline learning is used to obtain a database of contexts with associated parameter values. The online parameter tuning extracts the current context from the video and utilizes the learned values to adapt the tracking algorithm. The authors of [48] present an adaptive robot perception architecture. Context changes are monitored during runtime and utilized to trigger an update of the perception pipeline's configuration. The new configuration is obtained from a repository that contains satisfactory configurations for various conditions determined during an earlier training phase.

For online parameter tuning, the system determines new values *on-the-fly* by optimizing them according to the current state. Tuning algorithms to adapt the parameters typically determine the new parameter set based on the optimization of certain metrics (e.g., image quality). Commonly used examples of parameter optimization algorithms are *grid search*, *random search*, and *gradient descent* [49]. *Bayesian optimization* (BO) is a more efficient method to obtain the optimal parameters (an example is presented in [50]).

The work presented in [44] addresses the cost-intensive customization required to deploy a generic robot in a certain environment. Human experts have to utilize their system knowledge to determine satisfactory values for static robot parameters manually. Since this process is generally time-consuming, it significantly increases the cost of deployment processes. The authors describe an approach that automatically tunes the parameters of a robot on-site without expert supervision.

2.1.6 Sensor Fusion

Sensor fusion describes the process of combining data from multiple sensors in order to enable joint processing. The authors of [51] provide a very generic definition of information fusion:

> *"Information fusion is the study of efficient methods for automatically or semi-automatically transforming information from different sources and different points in time into a representation that provides effective support for human or automated decision making."*

In the context of environmental perception systems, the fusion of multiple sensors is utilized to obtain a more robust representation of the environment. Multiple perception sensors are deployed to extend the system's field-of-view and increase the perception task's robustness by compensating individual sensors' weaknesses. Sensor fusion is then required in order to utilize the extra information in the joint processing of the heterogeneous data.

Multiple challenges arise when the fusion of data from multiple sensors is targeted. In order to surpass single-sensor solutions, these challenges have to be addressed and solved. The main challenges include the temporal and spatial alignment of the data, the different data types (i.e., structure, dimension, rates, abstraction, acquisition time), and the handling of dynamic and conflicting data. As pointed out in [52], there are multiple ways to classify sensor fusion techniques. The classification used in this thesis is based on the abstraction level of the input data [53]. The fusion methods can be classified into low-, mid-, and high-level sensor fusion.

Low-Level

Low-level sensor fusion describes the fusion of low-level perception data, e.g., at signal- or pixel-level. This includes the fusion of raw sensor data and the fusion of the so-called *observations* (according to Figure 2.2). These observations describe modified versions of the raw sensor data. Examples for low-level data include the output data after non-destructive pre-processing (e.g., range from time-of-flight values) or after a reversible change of representation (e.g., coordinate frame transformations). In low-level sensor fusion, the decision about object extractions is taking place after the data was fused. This avoids loss of information during data abstraction and enables the combined recognition of objects, which would have been lost during individual processing. The approach, however, demands a centralized processing unit with high computational power and high bandwidth to the sensors.

A popular way to fuse heterogeneous sensor data at a low level is via the common representation in an occupancy grid. An occupancy grid represents the probability of occupancy for a finite amount of grid cells around the ego-vehicle. The individual sensors' readings can be spatially assigned to occupancy grid cells and utilized to update the cell's probability of occupancy.

With the advent of machine learning-based methods, low-level data from different sensors can be directly utilized as an input to sophisticated neuronal networks [54]. The authors of [31] present an approach to fuse radar data and vision camera images for object detection using deep learning. The authors of [55] present a *super-sensor* for 360°

environmental perception, which provides a combined low-level representation of multiple perception sensors. They address the temporal data acquisition of the utilized sensors and the crucial ego-motion correction to fuse the data.

Mid-Level

Mid-level sensor fusion describes the fusion of data at feature-level, i.e., after significant data processing steps have been applied to the low-level perception data. This includes extracted feature points from the raw data (e.g., edges/corners in an image, radar targets) and the non-reversible conversions of the data to other representations (e.g., gradient image, segmented image).

The work published in [56] shows an approach to fuse radar and lidar sensor data. The proposed fusion architecture separately pre-processes the data streams in order to perform centralized sensor fusion at feature-level. Similar to most perception platforms, the utilized radar sensor acts as an independent sensor module and solely communicates processed data to the remaining system. The authors of [57] present a method to fuse radar data with images from vision cameras in order to improve the vehicle-detection performance at high distances. They incorporate radar data into a vision-based deep learning vehicle detection network. The utilized radar data is received from the sensor as targets. The combination of the data in the detection network is performed at a medium processing stage. The automated driving platform Zeus, presented by Burnett et al. [22], performs lane detection by fusing the extracted lanes from the vision camera's image and the lidar range data. The camera image is transformed into a bird's eye view in order to detect the lanes, while the lidar approach utilizes the different reflectivity of the road surface.

High-Level

High-level sensor fusion describes the combination of high-level perception data (e.g., objects, tracks) from multiple sensors. Each sensor's raw perception data is individually processed in order to obtain a high-level representation of the respective perception data. One challenge in high-level sensor fusion is the data association of the perception data (objects, tracks) from the different sensors. Additionally, high-level fusion is subject to information loss during the data-abstraction process. Observation details contained in the raw sensor data might not result in detected objects after the application of the processing pipeline. High-level sensor fusion enables the deployment of a decentralized system architecture, where each sensor module performs an individual detection up to the object level. The data abstraction reduces the communication load, enabling the different modules to be connected with various types of buses.

A number of approaches have been published, addressing the high-level fusion of vision and lidar data for vehicle detection [58] and pedestrian detection [59]. The authors of [58] compared the performance of the *global nearest neighbors* (GNN) and *joint probabilistic data association* (JPDA) techniques for data association. Other methods rely on the *multiple hypothesis tracking* (MHT) algorithm, an improvement over the JPDA technique [60]. Baidu's Apollo Open Platform is capable of fusing vision, lidar, and radar data on a high-level. Each sensor performs individual object detection and tracking before the data is fused to extract the final objects. The authors of [61] show an example of high-level

sensor fusion, utilizing lidar, radar, and vision sensors. They present a joint probabilistic perception algorithm for data association, tracking, and classification of vehicles. Since the sensors complement each other, the approach enables robust perception in all weather conditions.

Hybrid-Level

In practice, it is difficult to unambiguously assign fusion techniques to the above-mentioned levels of fusion. Hybrid-level fusion describes the process of fusing sensor data at different abstraction levels (i.e., observations, objects, features, tracks). A single system may also perform various types of sensor fusion techniques, depending on the targeted application and the individual sensor streams included in the respective fusion task.

A prominent example of hybrid-level sensor fusion is the projection of regions of interest, identified in range data, into the 2D image space to perform robust obstacle classification. In [62], mid-level perception data from the laser-scanner is included to augment the vision camera's low-level perception data. Boss [5] utilizes individual sensor layers within its perception system for moving-object detection. The features obtained after low-level processing are filtered using the currently predicted environment state from the fusion layer. The remaining verified features are then further processed and communicated to the fusion layer at different abstraction levels. A hybrid-level fusion is also utilized in a submodule of the Autoware software stack [20]. The authors project the detected objects from the 2D image into the 3D space in order to perform driving actions.

2.2 Applications

The intended purpose of an environmental perception system is the fulfillment of certain perception tasks. The output of a perception task is provided to succeeding subsystems in order to contribute to the accomplishment of the desired applications. In the context of ADS, common perception tasks include object detection, localization within a map, and the determination of drivable space. This section provides a short overview of common perception tasks implemented as use cases in this thesis' environmental perception system.

2.2.1 Obstacle Detection

This perception task targets the detection of objects of interest in front of a vehicle/robot. Objects of interest include street signs, trees, other vehicles, or pedestrians. Robust obstacle detection is among the most crucial requirements of automated vehicles and autonomous robots. Additionally, obstacle detection acts as a base for more advanced perception tasks, like object classification or tracking. Most perception systems utilize their range sensors (e.g., lidar, radar, stereo vision) to detect obstacles and to create a local map. The range data enables the direct estimation of the objects' positions with respect to the platform.

Lidar sensors are commonly used as the primary source for obstacle detection. Due to its active illumination, the sensor is less sensitive to lighting conditions and directly provides a 3D understanding of the scene. A common representation of static obstacles is an occupancy grid (either in 2D top-down or in 3D with voxels). Recently also a number

of algorithms have been developed, which utilize 3D point clouds to detect and recognize objects directly.

The resolution of radar sensors is typically too low to utilize them as standalone solutions for obstacle detection. However, since radar is resistant to harsh weather conditions, the provided positions and radial velocities are of high value for many applications (like obstacle detection). Thus, radar sensors are often used in today's ADAS and will likely be included in future perception systems of automated vehicles. The authors of [63] show a platform that uses a custom radar sensor to create an occupancy grid from low-level radar data.

Images from vision cameras do not comprise distance information and have to be analyzed at a deeper level in order to detect obstacles. Vision images are rather used for the recognition of objects by using learning-based algorithms on regions containing obstacles. State-of-the-art methods to detect obstacles directly from the 2D image stream are typically based on deep learning. Examples are the *You Only Look Once* (YOLO) detector [64] or the *Single Shot MultiBox Detector* (SSD) [65]. However, these methods are limited to pre-trained objects and tend to fail in harsh weather conditions since they solely rely on vision camera images. Additionally, the detected regions do not hold any information about the corresponding objects' real sizes and scales.

As demonstrated in [66], a local occupancy grid map can be utilized for obstacle avoidance. The authors utilize an ego-centered occupancy grid built from two radar sensors and one lidar sensor. They also provide a detailed description of how to compensate the ego-vehicle motion to get a consistent grid, even though the vehicle is moving.

2.2.2 Environment Mapping

Environmental perception systems can use their perception sensors to create a local representation of the environment. This local information about the environment can then be utilized for tasks like localization, mapping, free-space determination, or obstacle detection. The work published in [67] presents an overview of commonly used localization techniques for automated driving.

An automated vehicle's most crucial tasks include its absolute localization (on a global map) and relative localization (on a local map). GNSS sensors are popular for coarse global positioning but cannot achieve a highly accurate localization in any scenario (e.g., due to multi-path effects or tunnels) [10]. The inclusion of IMU data and wheel odometry results in precise relative pose-updates and counteracts GNSS fluctuations [68]. However, as stated in [10], the GNSS-based solutions are typically not reliable enough to estimate the vehicle's position relative to a map with sufficient precision (e.g., within a lane). Thus, perception sensors (e.g., lidar or vision cameras) are often used to perform local localization.

Simultaneous localization and mapping (SLAM) algorithms are utilized to create a local map and localize the robot/vehicle within that map. SLAM-based algorithms are popular in environments without prior availability of a detailed map. A local map of the environment is not only important for fine-grained localization. The knowledge about the local environment can also be beneficial to other tasks such as obstacle detection or local path-planning. Since SLAM algorithms are computationally expensive, the provided rate of accurate SLAM positions is limited by the system's computation performance and

the utilized perception sensors' update rates. Thus, the position obtained by the SLAM localization is temporary also updated by the vehicle's dead-reckoning sensors (odometry, IMU) and GNSS, which are provided at a higher rate.

The work presented in [4] shows an approach to create a local, static map by incorporating the 3D points from a Velodyne 64-beam lidar sensor. As stated in [11] and [69], the vehicle's ego-motion has to be considered when new data is added to a map. The authors of [70] present an approach, which provides high-frequency pose estimations of a vehicle by utilizing lidar odometry, an IMU, and wheel odometry. The IMU was utilized to compensate for the intra-frame distortion of the lidar measurements. The work published in [71] incorporates lidar measurements in addition to the odometry data in order to increase the precision of the estimated poses.

2.2.3 Pedestrian Detection

Robust pedestrian detection is among the key requirements in order to establish full acceptance of automated vehicles. The inability to reliably detect any pedestrians in the vehicle's surroundings is likely to lead to accidents, resulting in injuries and fatalities. The sensitivity of that topic was revealed after the first fatal crash induced by an automated vehicle operated in self-driving mode in 2018 [72]. Tragic accidents like this one lower the general public's acceptance of automated driving and further raise the demanded robustness level of the ADS. Safety-critical perception tasks, like pedestrian detection, have to be absolutely robust before their deployment. Thus, there is a high demand to raise the quality of pedestrian detection in automated vehicles to a higher level.

Traditional object detection algorithms extract features from images, which are then processed by a learning algorithm. *Histogram of oriented gradients* (HOG) features are a popular choice in order to detect pedestrians [73]. HOG-based pedestrian detection methods introduce a relatively low computation overhead and achieve a high detection rate. Recently, deep learning-based approaches for pedestrian detection have gained popularity. *Convolutional neuronal networks* (CNN) are applied to the full vision images in order to detect all pedestrians in the scene. Prominent examples of CNNs for pedestrian detection (and also object detection) are the YOLO [64] and the R-CNN [74] detection systems.

The authors of [62] present an approach to fuse lidar range data with 2D vision images for pedestrian detection. They extract laser segments from the lidar data and project them onto the image plane in order to obtain multiple regions of interest within the 2D image. An SVM classification is performed using multi-window detectors within the regions of interest to detect potential pedestrians.

Mercedes Benz's research vehicle *Bertha* also extracts regions of interest using its stereo vision setup [19]. Each region of interest is subject to a multi-cue pedestrian classification, utilizing HOG features and texture-based features. Another approach of detecting pedestrians with Bertha's radar sensors, utilizing the micro-Doppler effect, is presented in [25].

The work published in [75] shows an approach to detect pedestrians using the data from a lidar sensor and a vision camera. The lidar data is utilized to identify object clusters in the scene, while a neuronal network (YOLO [64]) detects 2D bounding boxes of pedestrians. The fusion of these two outputs provides the 3D information of pedestrians in the lidar sensor's domain and the vision camera's domain.

2.3 Summary and Difference to the State-of-the-Art

As elaborated in this chapter, there exists a vast amount of work about environmental perception systems which mostly origins from robotics and automated driving research. The systems are continuously improved, utilizing advances in perception algorithms and refined sensor setups. In addition, the processing modules benefit from a performance boost caused by the introduction of automotive-qualified high-end supercomputers. However, current perception modules are still far from flawless. They commonly fail in harsh weather conditions and have disadvantages regarding speed, precision, and robustness. Since the perception quality has a substantial impact on the overall system's performance, the perception module is subject to further improvements.

Within the last two decades, multiple environmental perception systems have been presented to the research community, providing an overview of their designs and implementations. Additionally, multiple open datasets were introduced to encourage research on perception algorithms and allow the performance comparison of developed algorithms. The open-available perception platforms lack crucial components in order to enable research, which targets the exploitation of low-level sensor data. The majority of the institutions do not expose their platforms' hardware/software details to a level required to rebuild a similar system (e.g., sensor selection, software framework). Although open datasets provide an excellent entry point to develop certain perception algorithms, they are incapable of enabling research on low-level hardware and software aspects of perception systems (e.g., synchronization, data handling). Additionally, researchers are limited to the sensor types and the configurations used in the respective dataset and cannot individually evaluate the algorithm in customized scenarios.

This work presents the system design of a heterogeneous sensor fusion platform capable of utilizing low-level radar data. The thesis extends the currently available work within the research community with details of a custom-built platform capable of performing a low-level fusion of sensor data. Similar open approaches (like [20] or [18]) present the implementation of full software stacks for ADS, but lack a number of the perception module's design and implementation details. In contrast to existing approaches, this work focuses solely on the perception subsystem and provides detailed information about its composition. The presented platform provides solutions to arising challenges during the design and the implementation of a multi-sensor perception platform. The calibration and synchronization of the heterogeneous perception sensors are presented, enabling the output of well-structured data streams for further processing.

The presented approach provides a detailed system architecture of a multi-sensor perception system, which implements a highly adjustable low-level interface to the perception sensors. In contrast to the majority of open-available perception platforms, the presented platform utilizes a low-level interface to a radar sensor in addition to a vision camera and a ToF camera. This work provides an open system, which can be utilized by the research community in order to conduct research on low-level fusion concepts with focus on radar data. In order to demonstrate the potential of the perception platform, multiple perception applications are implemented as use cases of the system (e.g., environment mapping, obstacle/pedestrian detection).

Chapter 3

Background

This chapter introduces the technical fundamentals of ToF cameras, automotive radar sensors, and vision cameras. The provided background information of the utilized sensor technologies is essential to entirely understand the conducted work of this thesis. Thus, this chapter is especially targeted to readers not yet familiar with the aforementioned sensors.

3.1 Time-of-Flight Cameras

This section introduces ToF cameras for environmental perception. The working principle of the ToF camera's pixels in order to measure and calculate distance information is introduced. Additionally, the most influential characteristics of ToF cameras are described.

3.1.1 Basic Principle

ToF cameras utilize the travel time of light in order to determine the distance to objects in the scene. One efficient way to implement indirect ToF cameras is to utilize modulated light and an array of *photonic mixing device* (PMD) pixels. As illustrated in Figure 3.1, the scene is illuminated with modulated *near-infrared* (NIR) light. The light is backscattered by the objects in the scene and, via an optical lens, projected onto an image sensor consisting of an array of PMD pixels. The output of these pixels can be utilized to efficiently determine the phase difference between the transmitted and received light. The corresponding range value can then be calculated for every pixel by utilizing the obtained phase difference and the speed of light.

Photonic Mixing Device

A PMD sensor is capable of measuring the cross-correlation $c_\tau(x, y)$ between the reflection of the modulated light signal $r(t)$ and a phase-shifted version of the transmitted signal $s(t + \tau)$. A simplified illustration of a PMD pixel's cross-section in *complementary metal-oxide-semiconductor* (CMOS) technology is shown in Figure 3.2. The photoelectric effect causes the incoming light to generate electrons in the CMOS substrate. The modulation signal controls the electrons' movement into the right and left capacitors (so-called *buckets*). Depending on the modulation signal U_{mod}, an electron either moves to the right

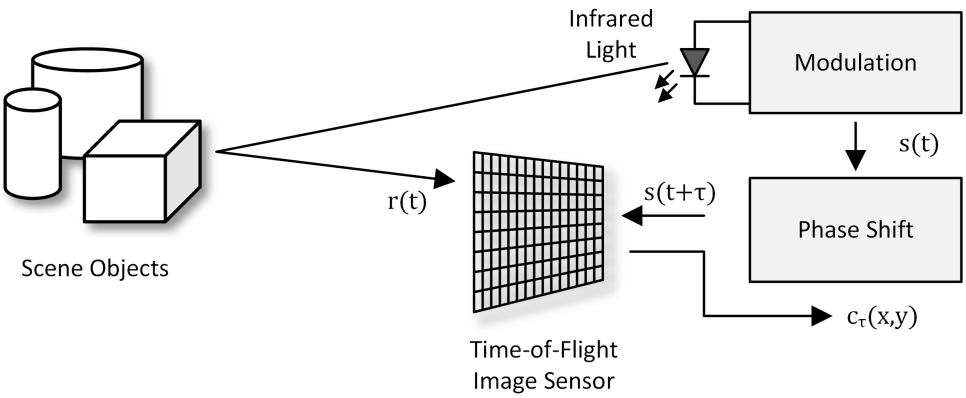


Figure 3.1: Illustration of the ToF principle, adapted from [76].

or the left capacitor (bucket A or B). The voltage difference between the two buckets U_A and U_B reflects the correlation between the incoming light and the applied reference signal. This voltage difference is then utilized to determine the phase difference between the reflected and the transmitted light.

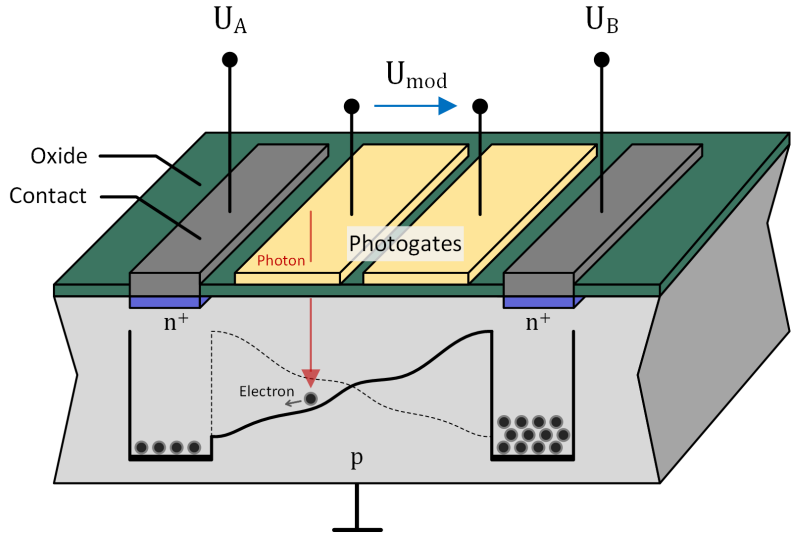


Figure 3.2: Simplified structure of a photonic mixing device pixel, adapted from [77].

Distance and Amplitude Calculation

A common way to determine the distance and the amplitude value for every pixel is the application of the *four-phase algorithm* [78]. First, the correlation value between the modulated light signal $r(t)$ and a phase-shifted version of the transmitted signal $s(t + \tau)$ is determined for four different phases (0°, 90°, 180°, and 270°). The raw sensor data holds the cross-correlation values C_x of the received signal and the phase-shifted versions of the transmitted signal.

The amplitude A of the correlation signal (also known as intensity) can be determined via the four samples of the correlation function C_x, as seen in (3.1). The amplitude value is a measure of the amount of received light and can be utilized as additional information in later processing steps (e.g., as confidence indication).

$$A = \frac{\sqrt{(C_{270^\circ} - C_{90^\circ})^2 + (C_{0^\circ} - C_{180^\circ})^2}}{2} \tag{3.1}$$

The phase difference $\Delta\varphi$ can be determined with four raw phase-images by utilizing the four-phase algorithm [78]. The arctangent function is calculated as stated in (3.2). Since the arctangent function is the most complex component of the distance calculation process, the selection of the arctangent algorithm has a high influence on the system's processing duration and latency. Popular choices are series expansion, iterative approaches, and lookup tables. The most suitable method depends on the utilized processing architecture and the desired application.

$$\Delta\varphi = \arctan\left(\frac{C_{270^\circ} - C_{90^\circ}}{C_{0^\circ} - C_{180^\circ}}\right) \tag{3.2}$$

The distance d to the reflecting object can be easily determined with the phase difference $\Delta\varphi$, the speed of light c, and the light modulation frequency f_{mod}, as seen in (3.3).

$$d = \frac{1}{2} \cdot \frac{c}{f_{mod}} \cdot \frac{\Delta\varphi}{2\pi} \tag{3.3}$$

The phase difference, determined by the four-phase algorithm, is limited by the 2π-periodicity of the correlation signal [78]. Thus, the provided distance values of the ToF camera are only unambiguous within the corresponding range interval. The camera's unambiguous range depends on the modulation frequency, which defines the spatial propagation of a single signal period. The dependency of the unambiguous range d_u on the modulation frequency f_{mod} is stated in (3.4). Since a reduction of the modulation frequency also decreases the measurement precision, the control of the unambiguous range via that parameter is only possible to a limited extent.

$$d_u = \frac{1}{2} \cdot \frac{c}{f_{mod}} \tag{3.4}$$

The *eight-phase algorithm* is a method to expand this range by combining two four-phase measurements acquired at different modulation frequencies. Figure 3.3 shows an illustration of the extended unambiguous range when using the eight-phase algorithm. In order to determine the eight-phase distance, the respective unambiguous ranges are added to each four-phase distance until the distance values of both modulation frequencies match. A suitable choice of the modulation frequencies and their relation can significantly extend the combined measurement's unambiguous range.

The different phases of an eight-phase ToF measurement are illustrated in an illumination intensity over time chart in Figure 3.4. The camera performs eight sequential phase measurements, utilizing four measurements for each of the two modulation frequencies. A single-phase measurement consists of an illumination phase (peaks in the chart) and a readout phase, in which the pixel values are sampled and digitized by the sensor.

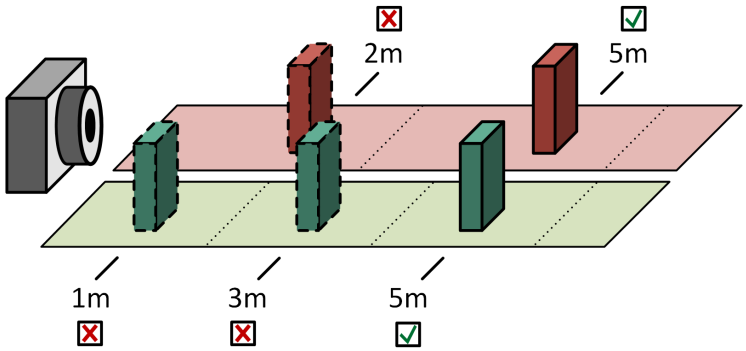


Figure 3.3: Unambiguous range extension using the eight-phase algorithm, adapted from [79].

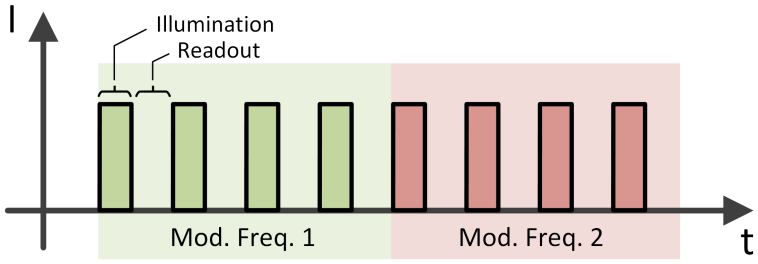


Figure 3.4: Timing of an eight-phase ToF measurement. Plot of the illumination intensity over the time.

Range and Precision

A ToF camera's maximum range is influenced by a number of factors, including the objects' reflectivity, the pixel size, the transmission power, and the illumination time. One of the major factors is the scene objects' ability to reflect the infrared light at the illuminated wavelength. As stated in (3.5), the amplitude A of the received infrared light depends on the surface reflexivity ρ, the distance r, and the angle of incidence θ. Figure 3.5 illustrates an object's reflection characteristics.

$$A \propto \frac{\rho \cdot cos(\theta)}{r^2} \tag{3.5}$$

An object can be detected if enough modulated light is received by the pixels of the sensor. A larger pixel size results in more received light and contributes to a higher range. The amount of transmitted light can be controlled with the illumination power and the illumination time. While the illumination power is usually fixed due to the used hardware (LED, laser), the illumination time can be adjusted to an arbitrary value. An increased illumination time increases the number of received photons, the amount of noise, and the measurement duration and may lead to saturated pixels. Additionally, eye safety has to be considered since infrared light can harm human eyes.

State-of-the-art ToF cameras claim to achieve ranges of over 10 m [80], [81]. This stated distance, however, does not guarantee that every object within that range is detected. The provided value states that a camera is capable of detecting standardized objects at that range (e.g., a surface with 10% reflectivity).

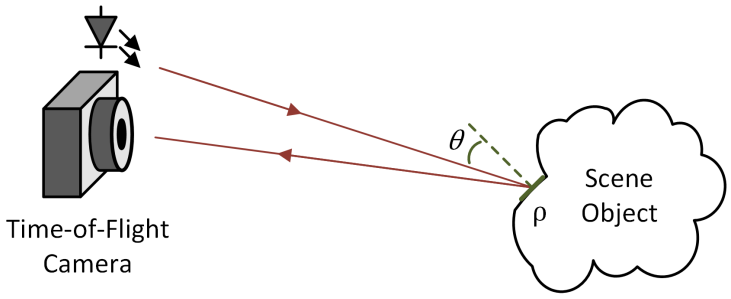


Figure 3.5: Reflection characteristics of an infrared-illuminated object, adapted from [82].

The accuracy of the detected measurement depends on a number of factors, including the *signal-to-noise ratio* (SNR) and the modulation frequency. Closer, highly reflective objects are detected at a higher accuracy than objects at a higher distance with low reflectivity.

The presence of background illumination lowers the SNR (e.g., due to photon shot noise). In addition, unmodulated light generates electrons into both buckets, increasing the likelihood of single pixels to saturate. Even though modern ToF cameras include additional circuitry to reduce the influence of background illumination [77], they still suffer from a decreased outdoor performance.

3.1.2 Characteristics

The performance of a ToF camera depends on various external and internal factors. The most influential characteristics of a ToF camera are listed below.

- Sensor resolution
  The resolution of a ToF camera defines the number of distance values provided by the image sensor. Current ToF cameras achieve resolutions of up to 640×480 pixels [81].

- Field-of-view
  The field-of-view of a ToF camera is given by the utilized optical lens mounted in front of the image sensor. While a wide-angle lens enables the coverage of a larger area, the provided image loses details. The impact area of the illumination unit has to be aligned with the field-of-view.

- Frame rate
  The maximum frame rate is limited by the illumination time and the measurement type (four- or eight-phase). Current systems are capable of providing frame rates of over 30 *frames per second* (FPS).

- Modulation frequency
  The utilized modulation frequency is typically in the range between 10 MHz and 100 MHz. The used value determines a measurement's unambiguous range and affects certain other performance values (e.g., the measurement precision).

- Illumination unit
  The illumination unit is equipped with LEDs or *vertical-cavity surface-emitting lasers* (VCSELs) and illuminates the scene with modulated infrared light. In general, infrared light with wavelengths between 850 nm and 1600 nm is utilized. The illumination power influences a measurement's range and quality but has to comply with eye-safety restrictions.

- Interface
  Standalone ToF cameras typically implement common interfaces (e.g., USB 3 or Ethernet) to transfer data and configure the sensor. Some cameras additionally provide interfaces to synchronize the camera (e.g., external trigger input).

3.1.3 Time-of-Flight Processing

The eight-phase algorithm uses eight raw ToF phase-images in order to calculate the distance and the amplitude image. The eight phase-images are sequentially captured and transferred to the processing system after the measurement was acquired. For the computation of the range image, the eight phase-images are split into two groups of four images for both utilized modulation frequencies. The four-phase algorithm is applied to each of these two groups to calculate the amplitude and distance image (according to (3.1) and (3.3)). The eight-phase algorithm combines these two outputs to an amplitude image and a distance image with an extended unambiguous range. Figure 3.6 shows the obtained

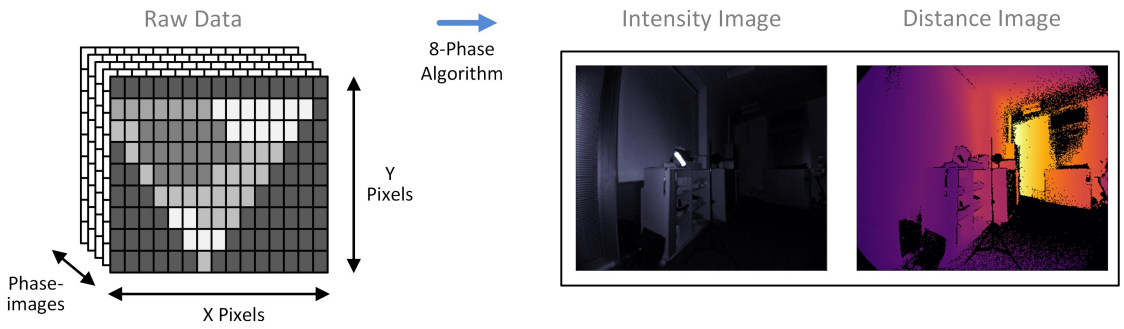


Figure 3.6: ToF raw data processing flow.

amplitude and distance image from the raw data after the application of the eight-phase algorithm.

The amount of raw data from a ToF measurement $D_{ToF,Raw}$ depends on the number of pixels (*pixel_cnt*), the data depth of each pixel (*data_depth*), and the measurement mode (four- or eight-phase). The amount of raw data for a single eight-phase measurement is stated in (3.6).

$$D_{ToF,Raw} = 8 \cdot pixel\_cnt \cdot data\_depth \tag{3.6}$$

A typical ToF camera with a resolution of 352×287 pixels and a data depth of 16 Bit provides about 12.9 MBit of raw data for a single eight-phase measurement. This raw data is transferred from the sensor to the processing system in order to calculate the amplitude and the distance image. In typical applications, the calculated amplitude image contains 16 Bit integer values, while the distance image contains 32 Bit float values. While these images can already be utilized to perform further processing, many applications additionally calculate a point cloud in order to provide a 3D representation of the range data.

In order to calculate the 3D point cloud from the distance an amplitude image, the camera and lens parameters have to be known. Using this information, each pixel of the distance image can be assigned to a 3D point in the ToF camera's coordinate system. The points of the resulting point cloud contain six data fields, holding the corresponding (x/y/z) position, intensity, confidence, and noise values. The resulting data size of a ToF camera's point cloud $D_{ToF,PC}$ depends on the number of pixels and the data type of the point cloud's fields, as stated in (3.7). A ToF point cloud consisting of 352×287 points with 32 Bit float fields amounts to a data size of about 19.4 MBit.

$$D_{ToF,PC} = 6 \cdot pixel\_cnt \cdot data\_depth \tag{3.7}$$

3.2 Automotive Radar Sensors

This section presents an overview of automotive radar sensors and introduces their basic functionality. The radar sensors' main characteristics are presented, and a short introduction of the raw data processing flow is provided.

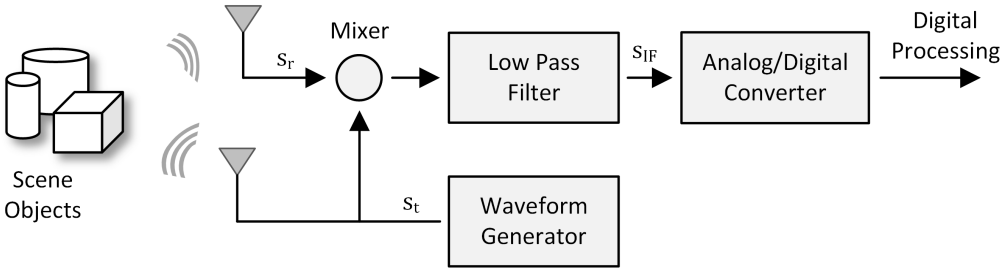


Figure 3.7: Overview of an automotive radar sensor's main modules.

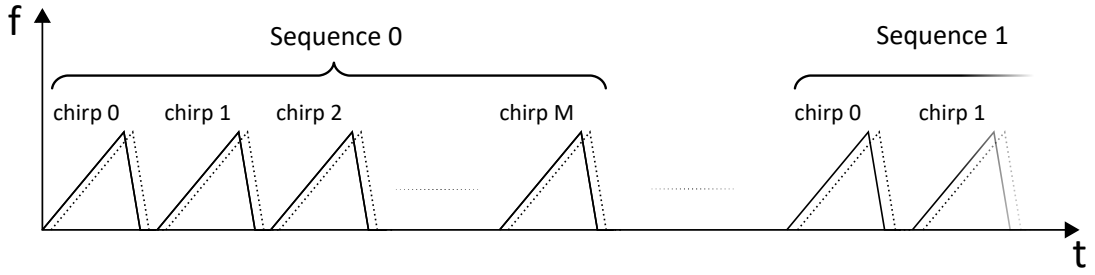
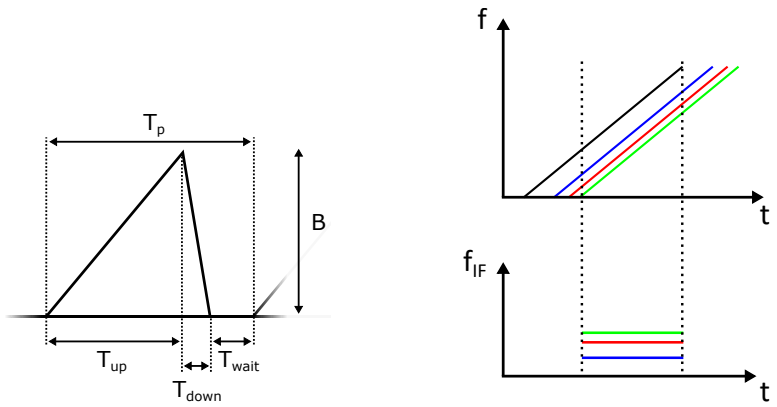


Figure 3.8: Waveform shape of fast-chirped frequency sequences, transmitted by automotive radar sensors. The dashed signal represents the delayed response of the transmitted signal, reflected by an object in the scene. Obtained with changes from [87].

3.2.1 Basic Principle

A radar sensor is capable of transmitting electromagnetic waves and receiving their reflections. The sensor utilizes the relationship between the transmitted and the received signal to obtain information about the environment (e.g., distance to objects, velocity of objects). The prevalent signals utilized by automotive radar sensors are *frequency-modulated continuous-waveforms* (FMCWs). This modulation principle enables the simultaneous detection of a target's range and velocity. A subtype of FMCW signals are sequences of short frequency ramps, *fast chirp sequences*. This waveform type allows the accurate detection of multiple targets located at a similar distance. Our publication [83] provides an overview of state-of-the-art automotive radar technology (see Chapter 8, Publication 2). Additional information regarding signal processing for automotive radar can be found in [84], [85], and in the book chapter published in [86].

An overview of an automotive radar sensor's main modules, utilizing fast chirp sequences, is illustrated in Figure 3.7. The sequences of fast frequency chirps are generated by a waveform generator and are transmitted to the scene. The transmitted signal's reflections are subsequently received, mixed with the transmitted signal, and low-pass filtered. The resulting low-frequency signal is called *intermediate frequency* (IF) signal and contains information about the distance to objects in the scene and their radial velocity. In order to extract this information, the IF signal is sampled at a high rate and digitized for further processing. *Fast Fourier transforms* (FFTs) are then applied to the sampled data to obtain the distance and the velocity to objects in the scene. Additionally, multiple receive channels are utilized to obtain the angular information to the reflecting objects in the scene.



(a) Single frequency chirp. **(b)** Frequency components.

Figure 3.9: Characteristics of a single frequency chirp. Multiple frequency components in the intermediate frequency signal are caused by reflections from multiple objects.

The signal form of the transmitted fast-chirped frequency sequences is illustrated in Figure 3.8. One measurement sequence consists of M fast frequency chirps. In practical applications, a single chirp period T_p consists of an upchirp T_{up}, a downchirp T_{down}, and a waiting portion T_{down}, as shown in Figure 3.9a. Due to the waveform shape, the frequency difference between the reflected and the transmitted signal holds information about the distance to a reflecting object and its corresponding radial velocity. Mixing these two signals and subsequent low-pass filtering results in the IF signal comprising that frequency difference. Multiple objects in the scene result in multiple frequency components in the IF signal (illustrated in Figure 3.9b).

As seen in (3.8), the frequency of the IF signal f_{IF} consists of a range-dependent component f_{range} and a Doppler-dependent component $f_{Doppler}$. The range component is determined by the bandwidth B, the range to the reflecting object R, the upchirp duration T_{up}, and the speed of light c. The Doppler frequency depends on the reflecting object's radial velocity v_r, the speed of light c, and the transmitted signal's center frequency f_c.

$$f_{IF} = \underbrace{\frac{2BR}{T_{up}c}}_{f_{range}} + \underbrace{\frac{2f_c v_r}{c}}_{f_{Doppler}} \tag{3.8}$$

Since the upchirp duration T_{up} of fast-chirped frequency ramps is generally set to a sub-millisecond interval, the obtained range component outweighs the Doppler component by by orders of magnitudes. Thus, the impact of the Doppler frequency can be neglected for fast-chirped sequences. The frequency of the IF signal f_{IF} is determined utilizing an FFT over the N samples of a single frequency chirp. The distance R to the reflecting object in the scene can then be calculated as stated in (3.9).

$$R = \frac{c\,T_{up}}{2\,B} f_{IF} \tag{3.9}$$

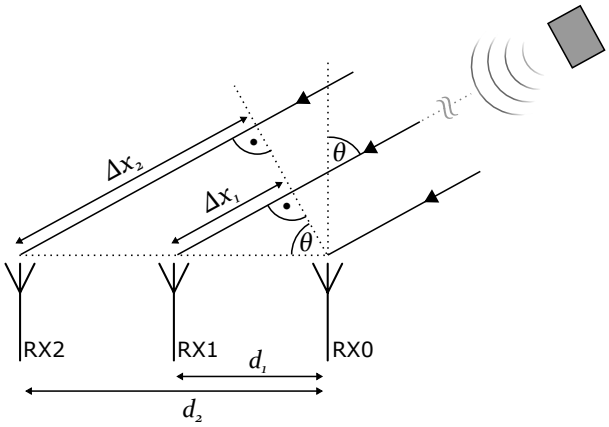


Figure 3.10: Phase delays between the individual receive channels due to the target's angle. The phase difference can be used to estimate the angle of arrival [87].

In order to obtain the radial velocity component of an object, the determined range value's phase changes between the consecutive chirps are considered. The phase value of an FFT bin can be utilized for that purpose since it is sensitive to a reflecting object's small radial movements. An object's radial velocity v_r causes small changes in the range $\Delta R = v_r\, T_p$ for each consecutive frequency chirp with chirp duration T_p. This results in a phase change $\Delta\Phi$ between consecutive range values of the chirp sequences, as stated in (3.10). The carrier signal's wavelength $\lambda = \frac{c}{f_c}$ defines the unambiguous range of the measurement.

$$\Delta\Phi = 2\pi\, \frac{2\, v_r\, T_p}{\lambda} \tag{3.10}$$

The radial velocity causes the obtained range phasor to oscillate at an angular velocity $\omega = 2\pi\, \frac{2\, v_r}{\lambda}$. The values of the oscillating range phasor are sampled once for each of the M chirps, with the sample period being the chirp duration T_p. A second FFT can be utilized to estimate of the phasor's angular velocity ω and, subsequently, the radial velocity v_r.

To obtain an angular estimation of the reflecting objects in the scene, multiple receive antennas are utilized. Figure 3.10 illustrates the different propagation paths for the different receive antennas, caused by an angle θ to a reflecting object. As stated in (3.11), the path difference Δx between the different receive antennas can be calculated using the phase difference $\Delta\phi$ between the received signal channels and the carrier signal's wavelength λ.

$$\Delta x = \frac{\Delta\phi}{2\pi} \cdot \lambda \tag{3.11}$$

The angle to the reflecting object θ can then be calculated using the path difference Δx and the baseline distance between the receive antennas d, as stated in (3.12).

$$\theta = \arcsin \frac{\Delta x}{d} \tag{3.12}$$

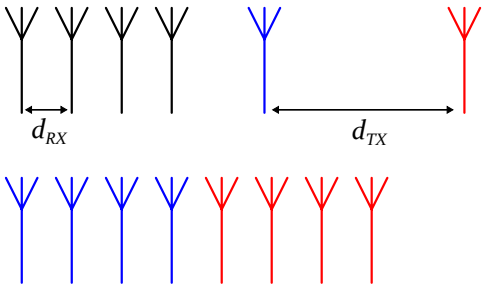


Figure 3.11: Arrangement of a virtual antenna array, obtained from [87].

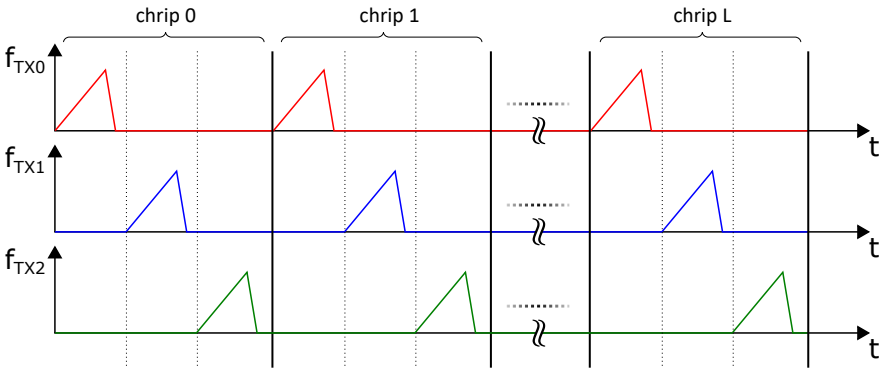


Figure 3.12: Temporal signal multiplexing applied to multiple transmit channels. The resulting virtual antenna array results inan improved angular resolution.

A well-established choice for the alignment of multiple receive antennas is the linear array arrangement with an equal spacing distance of $d = \lambda/2$. This spacing distance leads to the maximum aperture of the antenna while still maintaining unambiguous phase differences between the single channels. The phase differences between the signals of the equally-spaced receive antennas lead to an oscillating phasor, sampled at different spatial locations. Again, an FFT can be applied to estimate this phasor's angular velocity and, consequently, the reflecting object's angle. Depending on the antenna array's alignment (horizontal or vertical), the azimuth or the elevation angle of the detected objects can be obtained.

A higher number of receive channels results in an increased angular resolution. Since the number of physical receive channels is generally limited (e.g., due to size, cost, and hardware limits), other methods are utilized to increase the angular resolution. A common method is the formation of virtual antennas, where L_{TX} transmit antennas are utilized in combination with L_{RX} receive antennas. The combined antenna results in $L_{TX} \cdot L_{RX}$ virtual receive channels, as seen in Figure 3.11. The transmitted signals of the multiple transmit antennas have to be orthogonal in order to enable their separation at the receive antennas. In this work, temporal multiplexing of the transmit channels is implemented, where only one transmit channel is active at a time. As illustrated in Figure 3.12, the chirps are consecutively transmitted via each of the transmit antennas to enable orthogonal signals. Temporal multiplexing leads to an increased measurement time, limiting the frame rate and increasing the vulnerability to motion blur.

Range and Precision

The maximum range of a radar sensor depends on a number of factors. As seen in the radar equation (3.13), the received power P_r depends on the transmitted power P_t, the antenna gain G, the wavelength λ, the object's range R, and the object's radar-cross-section σ. The radar-cross-section is a measure of an object's ability to reflect radar waves, directly influencing the received power. Large, metallic objects cause strong reflections of radar waves (e.g., cars, trucks, planes) and allow a simple detection. Small, non-metallic objects (e.g., trees, humans, birds), have a lower radar-cross-section and are more difficult to detect. In general, the antenna gain and the radar-cross-section have a high angular dependency.

$$P_r = \frac{P_t\, G^2\, \lambda^2\, \sigma}{(4\pi)^3\, R^4} \tag{3.13}$$

As seen in the equation, the received power is inversely proportional to the fourth power of the reflecting object's range. In order to detect an object, the received signal has to be sufficiently high to separate it from the present noise. The major influences of an object's detectability are its shape, material, distance, and angular position.

The maximum unambiguous range of the radar sensor is limited by the upchirp duration T_{up} as stated in (3.14). An object at a higher range would not overlap with the transmitted pulse, resulting in an invalid range determination. In practice, the maximum detectable range is mainly limited by the sample frequency f_s of the analog-to-digital converter, which determines the frequency range of the FFT.

$$R_{max} = \frac{c\, T_{up}}{2} \tag{3.14}$$

The resolution of the determined radar range ΔR depends on the bandwidth B and the speed of light c, as stated in (3.15). In practice, the number of samples N limits the range signal's resolution since this number determines the spacing between the FFT bins. However, methods like zero-padding and windowing can be utilized to reduce that effect.

$$\Delta R = \frac{c}{2\, B} \tag{3.15}$$

The maximum unambiguous radial velocity $v_{r,max}$ of a fast-chirped radar sensor depends on the speed of light c, the chirp duration T_p, and the center frequency f_c, as seen in (3.16). The phase difference $\Delta\Phi$ between two consecutive chirps has to be lower or equal to π in order to allow an unambiguous estimation of the velocity.

$$v_{r,max} = \pm\frac{c}{4\, T_p\, f_c} \tag{3.16}$$

The resolution of the radial radar velocity Δv_r depends on the number of chirps M, the chirp period T_p, the center frequency f_c, and the speed of light c [88]. The impact of these components on the resolution is stated in (3.17).

$$\Delta v_r = \frac{c}{2\, M\, T_p\, f_c} \tag{3.17}$$

The angle to a reflecting object is calculated from the estimated phase difference using a nonlinear expression (as stated in (3.12)). Thus, the uniformly distributed bins after the application of the third FFT do not result in uniformly distributed angular resolutions. The estimated angle θ has its maximum resolution $\Delta\theta = \frac{\lambda}{d L_{RX}}$ at angle of zero while gradually decreasing towards $\pm\frac{\pi}{2}$.

3.2.2 Characteristics

The performance of an automotive radar sensor depends on a number of characteristics and key parameters. A selection of the most influential parameters of FMCW-based radar sensors is listed below.

- Radar frequency
  The utilized carrier signal frequency influences the radar wave propagation characteristics. Higher frequencies allow miniaturization due to smaller antennas but come with the price of increased atmospheric attenuation. The most common frequency band for automotive radar sensors is the 77-81 GHz frequency band.

- Antennas
  The number of receive antennas defines the radar's capabilities to estimate the angle to an object. Multiple transmit antennas can be utilized to create a virtual antenna with an increased number of receive channels. Depending on the antennas' arrangement, the targets' azimuth and/or elevation angles can be estimated.

- Transmitted signal shape
  The exact composition of the transmitted waveform has a strong influence on the radar's performance. For example, a higher number of transmitted chirps increases the frequency resolution while decreasing the maximum possible frame rate. The waveform structure has to be selected in compliance with the targeted application.

- Sample frequency
  The sampling frequency directly affects the range resolution and the maximum detectable range of a system. However, the maximum sample frequency is limited by the physical limitations of the utilized ADCs. Additionally, a higher frequency also increases the raw data size and the processing load.

- Frame rate
  A high frame rate is beneficial for many applications, especially in dynamic scenarios. The maximum possible frame rate depends on the set measurement characteristics (e.g., number of chirps/samples, data load), defining the measurement duration. The measurement precision can be increased by expanding the measurement duration, limiting the maximum frame rate.

- Transmission power
  A radar sensor's transmission power mainly affects the maximum achievable range. However, the extension of the range by increasing the transmission power is only possible to a limited extent. As stated in the radar equation, the received signal power decreases with the fourth power of the distance.

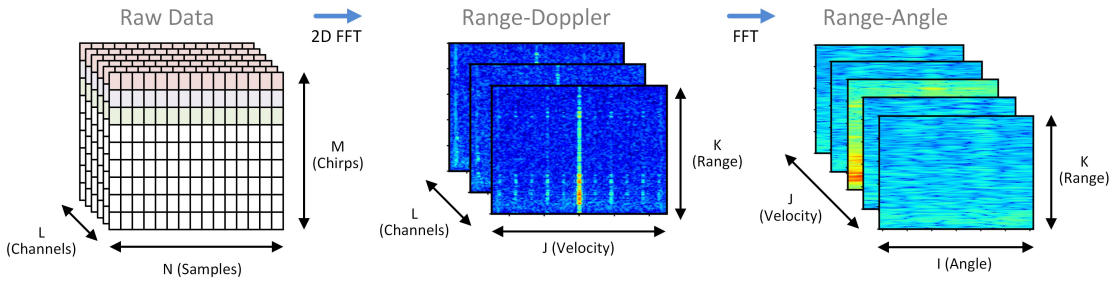


Figure 3.13: Radar raw data processing flow. The raw sample data is processed in order to obtain a range-velocity-angle data cube.

- Interface
  The interfaces of standalone automotive radar modules are often limited to automotive buses (e.g., CAN or FlexRay). These sensor modules typically provide their output data at high compression and abstraction levels. Radar sensors, capable of outputting raw data, typically provide USB 3 or Ethernet interfaces, enabling sufficiently high data rates.

3.2.3 Radar Processing

The digital radar signal processing starts after the analog-to-digital conversion of each receive channel's IF signal. As seen in Figure 3.13, a digitized measurement sequence results in a data cube of N samples, M chirps, and L channels. The received IF signal is sampled and converted to digital for each measurement sequence. The digitized measurement samples are transferred to the radar receive module. In this software module, the samples are reshaped into a 3D matrix of N samples, M chirps, and L receive channels (see Figure 3.13).

A 2D FFT (J- and K-point FFTs) over the samples and the measurement channels results in L range-Doppler images, one image for each of the receive channels (see Figure 3.13). Each image has a dimension of J velocity bins and K range bins, determined by the bins of the 2D FFT. A range-Doppler image displays the velocity and the range of objects in the scene. A third FFT (I-point FFT) is then calculated to obtain the range-angle-velocity data cube from the radar data with $I \times K \times J$ points. Possible representations of the range-angle-velocity data cube are J range-angle images with one image for each velocity bin (see Figure 3.13). In this representation, every range-angle image has a dimension of $I \times K$ points.

The dimensions of the raw radar data cube are the samples per chirp N, the number of chirps M, and the receive antennas L. The utilized data of a single radar measurement $D_{Radar,Raw}$ depends on the data depth of the samples $data\_depth$ in addition to the dimensions of the data cube, as stated in (3.18). A measurement utilizing N=2048 samples, M=128 chirps, L=16 antennas, and a data depth of 16 Bit results in a data utilization of about 67.1 MBit for a single measurement.

$$D_{Radar,Raw} = N \cdot M \cdot L \cdot data\_depth \tag{3.18}$$

3.3 Vision Cameras

This section provides an overview of vision cameras, a commonly deployed sensor type in environmental perception systems. Since vision cameras are well-established in the targeted research field, only a brief introduction to the basic principle is provided. In addition, the most influential characteristics of the sensor type are presented.

3.3.1 Basic Principle

In contrast to ToF cameras, vision cameras work without a light-emitting element. They utilize the scene's reflections of ambient light (e.g., sunlight, artificial light) in order to capture an intensity image of the environment. In general, a vision camera consists of an imaging device (with an image sensor) and an optical lens, which projects light onto the sensor. Each pixel of the image sensor is sensitive to visible light and outputs an intensity value after each measurement. An electronic or a mechanical shutter is used to control the exposure time of the camera. Due to the photoelectric effect, the incoming photons result in a charge stored in each pixel. After the exposure, the resulting voltage is amplified and digitized to an intensity value. The major parameters to control an image's acquisition are the exposure time, the amplification gain, and the lens aperture. These three parameters can be adjusted during runtime in order to provide a well-exposed image comprising the details of a scene.

3.3.2 Characteristics

The performance of vision cameras depends on a number of characteristics and parameters. Some of the major impact factors are presented here.

- Sensor type
  The majority of available image sensors are implemented using the *charge-coupled device* (CCD) or the CMOS technology. CCD sensors provide a higher dynamic range, while CMOS sensors can be manufactured at a lower cost and be operated at lower power consumption.

- Sensor size
  A larger sensor area is able to capture more light. This is an advantage in low-light situations and allows lower exposure times, resulting in less motion blur. However, a larger sensor is typically also more expensive and requires larger and heavier lenses.

- Resolution
  An image sensor's resolution defines the number of pixels in the output image. More pixels result in a more fine-grained perception of the captured image and reveal more details of the scene. However, a higher resolution also increases the amount of data per image, which increases the transfer and the processing time.

- Optical lens
  The focal length of the utilized lens specifies the camera's field-of-view. The aperture of a lens defines the amount of light that can get onto the sensor. A lens has to be compatible with the optical size of the sensor. Since the projection of real-world

lenses is never optimal, effects like distortion, vignetting, and chromatic aberration occur.

- Frame rate
  The frame rate of a camera defines the number of images captured per second. A camera's maximum frame rate is limited by the illumination time and the sensor's readout capabilities. Typical frame rates for vision cameras are 30 FPS, 60 FPS, and 120 FPS.

- Grayscale vs. color
  Color cameras typically use a pattern of red, green, and blue pixels on the sensor array. The output image is then calculated by interpolation of the neighboring pixels. Thus, the effective resolution of grayscale cameras is higher and favorable if color information is not needed.

- Global vs. rolling shutter
  Rolling shutter cameras do not read out all pixel values simultaneously, which can lead to a skewed output image for dynamic scenarios. Global shutter cameras perform the exposure and the readout of all pixels at the exact same time, but require more complex and expensive readout circuitry.

- Interface
  Two commonly used interfaces for industrial vision cameras are USB and Ethernet. USB 3 cameras are well suitable for high-speed imaging but have a limited cable length (around 5 m). Gigabit Ethernet cameras enable higher cable lengths (up to 100 m) but provide lower bandwidths.

A vision camera has to be selected according to the desired application. If a camera is used on a mobile platform or shall be able to capture moving objects, a global shutter is beneficial. A large sensor and a lens with a large aperture are favorable if low-light situations are targeted. When used for real-time processing, a lower resolution and a high-speed interface are favorable to minimize the introduced data load.

3.3.3 Video Camera Processing

A vision camera typically already provides an image stream, which can be directly utilized for further processing. The data size $D_{vision,Raw}$ of a vision camera's provided 8 Bit image stream depends on the number of pixels $pixel\_cnt$ and the data depth of each pixel $data\_depth$, as seen in (3.19). A Full HD image stream ($pixel\_cnt = 2.3$ Mpixels) with a data depth of 8 Bit for each of the three color channels ($data\_depth = 24$ Bit) results in a data size of 55.3 MBit for a single image.

$$D_{vision,Raw} = pixel\_cnt \cdot data\_depth \tag{3.19}$$

While vision cameras traditionally utilize a data depth of 8 Bit for each color channel, modern cameras are often also able to provide 10 or 12 Bit values for each pixel. Some vision cameras also offer the ability to apply certain image compression steps at sensor-level, significantly decreasing the camera's transfer load.

Chapter 4

Design

This chapter provides a detailed overview of the environmental perception platform's system design. First, the requirements of the desired perception system are elaborated and listed. Then, a system design is presented, able to meet these requirements. The underlying structure and the methodologies of the contained subsystems are described in detail.

4.1 Requirements

This section describes the design requirements of an environmental perception platform capable of accomplishing the goals set in Chapter 2 of this thesis. The system's main purpose is to enable fast and easy-accessible research on environmental perception systems and their underlying concepts. The perception platform shall include multiple heterogeneous perception sensors, expected to be deployed in future automated vehicles and autonomous robots.

To enable easy integration of new software modules, the system shall be based on a modular structure. The *base perception system* shall perform sensor-specific communication, data alignment, low-level data handling, and fundamental processing steps. The system's modules shall contain adjustable parameters (e.g., for the sensor configuration) and provide interfaces to request the dynamic adaption of these parameters during runtime. The well-structured output data streams of these modules shall be made available to succeeding subsystems for further processing. Certain common perception tasks (e.g., pedestrian detection) shall be added to the system as *use cases* in order to show the capabilities and the potential of the environmental perception platform. Additionally, the system shall allow an independent and mobile operation and provide the ability to record datasets for later evaluation.

An overview of the system architecture utilizing multiple environmental perception sensors is shown in Figure 4.1. In order to enable a modular structure, the architecture is split into two main parts: the base perception subsystem and multiple use-case subsystems. The base perception system handles the low-level interaction with the perception sensors. Multiple use cases utilize the base perception system's output to perform various perception tasks.

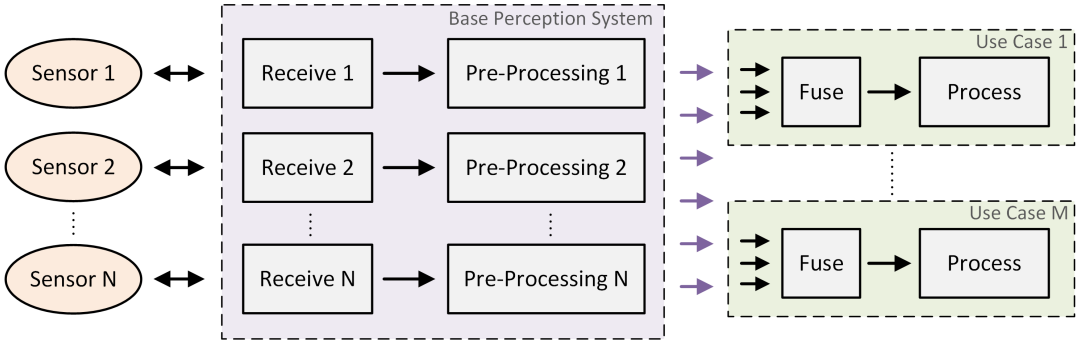


Figure 4.1: Proposed overall system architecture. The architecture is divided into the base perception system and multiple use-case subsystems.

4.1.1 Base Perception System

The base perception system is responsible for the low-level communication with the sensors, the early-stage data handling, and the pre-processing (see Figure 4.1). The base perception system's output, a number of temporally and spatially aligned data streams at different abstraction levels, can then be utilized by various use cases for application-specific processing. The following requirements for the design of the base perception system were elaborated.

- **Sensor selection**
  The selected sensors shall be able to provide raw measurement data to the system and shall offer the ability to adjust their configuration parameters.
- **Spatial data alignment**
  The relative alignment between the individual sensor frames has to be determined and considered to combine multiple sensor data streams into a common space.
- **Temporal data alignment**
  The acquisition times of the individual sensor measurements shall be included in the data streams in order to allow the temporal alignment of the different measurements.
- **Sensor data processing**
  The base perception system's data processing tasks are divided into a low-level layer and a pre-processing layer
  - **Low-level sensor interface**
    The *receive* modules shall handle the low-level communication with the perception sensors. This includes the reception of the raw sensor data as well as the configuration of the sensors.
  - **Sensor pre-processing**
    The *pre-processing* modules shall output processed versions of the raw sensor data at different abstraction levels. The modules' workload also includes error compensations, incorporation of calibration data, and reshaping/restructuring the data.

- **System parameters**
  Since the system parameters (sensor configurations, module parameters) have a strong influence on the perception performance, they shall be customizable. This enables a custom selection of the parameters according to the targeted environment.

4.1.2 Use Cases

Multiple common perception tasks shall be implemented as use cases to demonstrate the capabilities of the environmental perception platform. The aligned and structured data streams from the base perception system shall be used as input streams of the use-case subsystems. The data streams shall be made available at multiple abstraction levels to enable the evaluation of various sensor fusion concepts for each use case.

The use-case subsystems are, in general, composed of *fuse* and *process* modules. A fusion module utilizes a subset of the base perception system's data streams and combines them into an application-specific representation. If a desired input data stream is not directly available, additional pre-processing may be required by the respective use case. After the sensor data is fused, the use case's application-specific perception task is performed. The use case's output shall then be made available to the system for visualization or external systems and modules for further processing (e.g., path-planning).

The following use cases shall be implemented in order to demonstrate the capabilities of the base perception system architecture:

- **Context-aware parameter adaption**
  The use case shall dynamically adapt the system parameters during runtime. The current context shall be utilized to determine new parameter values and provide a satisfactory system performance in changing environments.

- **Obstacle detection**
  This use case shall detect obstacles (i.e., occupied space) in front of the perception platform. The perception task shall output the information in a visualizable format for succeeding modules.

- **Environment mapping**
  The data from the range sensors shall be utilized in order to build a 3D map of the environment while the platform moves through a scene.

- **Pedestrian detection**
  Multiple perception sensors shall be combined at different abstraction levels in order to perform an enhanced pedestrian detection.

- **Data visualization**
  This use case shall visualize the data streams provided by the base perception system or by other use cases. The use case shall implement different visualization approaches of heterogeneous sensor data and enable a direct interpretation of the perception output.

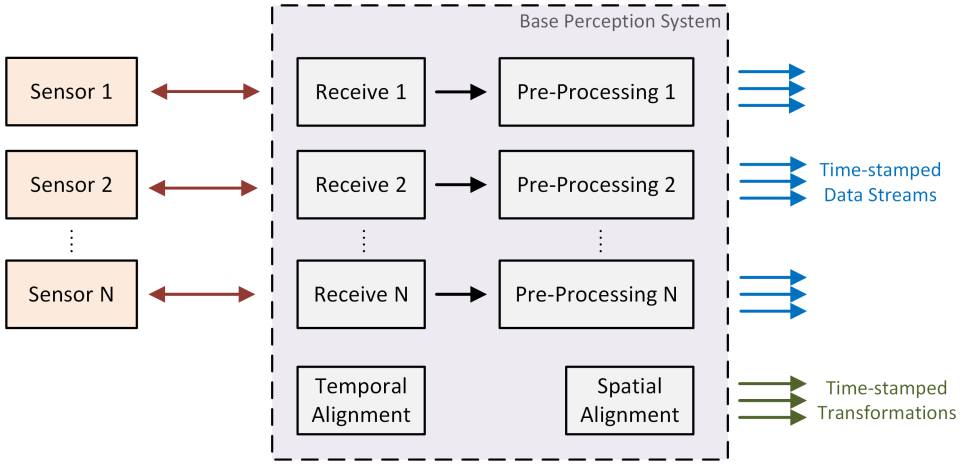


Figure 4.2: Main building blocks of the base perception system.

4.2 Base Perception System

The base perception subsystem is in charge of the low-level data handling and the communication with the individual perception sensors. The subsystem receives the raw data directly from the perception sensors, pre-processes it, and outputs multiple data streams at different abstraction levels. Additionally, the data streams are extended with temporal and spatial information of the measurement. The output data streams are then made available to the remaining system and can be utilized by further subsystems (e.g., subsequent use cases).

The base perception system's overall design is depicted in Figure 4.2. The sensors are connected to the base perception system via individual low-level receive modules, which are in charge of receiving the raw sensor data and setting the sensor configuration. Early processing of the sensor data (e.g., compression, error correction) is performed in the pre-processing modules, which output multiple versions of the sensor data at different abstraction levels. The *temporal alignment* module is in charge of triggering the data acquisition and assigning measurement timestamps to the data streams. The *spatial alignment* module keeps track of the environmental perception platform's frame positions over time (e.g., sensor measurement frame) and can be used to obtain the platform's position with respect to a global reference point.

4.2.1 Sensor Selection

One of the perception sensors' requirements is their ability to provide raw sensor data. Another requirement is the usage of sensors, which are expected to be deployed in future automated vehicles. Many commercially available sensor modules targeting the automotive industry are closed-source and only available as isolated sensor modules. Since this thesis's perception sensors are required to be openly available and provide low-level access, the considered sensors mainly consisted of development kits and evaluation boards. These sensors are neither-automotive qualified nor weather-resistant but rely on similar principles as commercially available modules. Thus, the results obtained with these sensors remain valid for all sensors of similar types, regardless of their certification level.

(a) ToF camera.

(b) Radar development kit.

(c) Vision camera.

Figure 4.3: The selected perception sensors of the environmental perception platform.

Since most research vehicles for automated driving are equipped with vision cameras, laser scanners, and radar sensors, similar sensors were desired for this work's platform as well. Due to the cooperation with Infineon Austria and the access to know-how and documentation of their semiconductor chips, Infineon-based perception sensors were prioritized during the selection process. A radar development kit based on Infineon's radar semiconductor chips was selected as the platform's radar sensor. A ToF camera was chosen as an infrared-based range sensor, acting as a substitute to a lidar sensor which utilizes a strongly related technology and provides a comparable output (i.e., point clouds). In addition, an industrial vision camera was selected, providing low-level access and offering numerous customization options. Pictures of the utilized perception sensors are depicted in Figure 4.3. This section presents details of the selected perception sensors and briefly introduces the system's additionally utilized sensors.

Time-of-Flight Camera

As ToF camera, the *CamBoard pico monstar*[1] was selected, utilizing the indirect ToF principle to obtain range information of the environment. Since the obtained range data from indirect ToF sensors (e.g., ToF cameras) and direct ToF sensors (e.g., lidar) is comparable,

[1]CamBoard pico monstar: a 3D imaging development kit (`https://pmdtec.com/picofamily/`).

they can be easily exchanged in existing sensor setups. Research on data processing based on the data from one of these two sensor types can often be generalized to be valid for both types. A picture of a CamBoard pico monstar ToF camera is shown in Figure 4.3a.

The CamBoard pico monstar is a 3D depth-sensing development kit offered by PMD Technologies AG, a German semiconductor company focusing on 3D imaging. PMD Technologies AG is cooperating with Infineon Technologies AG in order to develop and produce highly integrated 3D ToF image sensors. The first generation of the cooperatively developed 3D image sensors was released in 2013 [89]. Since then, the sensors have continuously been improved, and multiple generations have been released. The offered sensor portfolio includes products targeting the consumer market and products targeting the automotive market.

The selected ToF camera provides frame rates of up to 60 FPS and a resolution of 352×287 pixels. With the deployed wide-angle lens, the camera's field-of-view covers an area of 100°×85°. The ToF camera comes with a USB 3 interface and the software library *royale*, which handles the communication with the camera. Using the library, the camera configuration (e.g., illumination time, frame rate) can be adjusted during runtime. Additionally, certain pre-processing steps (e.g., error correction) can be performed before the data is provided to the user software. The CamBoard pico monstar can be configured to perform measurements via an externally applied trigger signal.

In the utilized configuration, the camera provides an array of points to the user software after an eight-phase measurement was performed. The array contains a (x/y/z) 3D position for each point, as well as an intensity value, a confidence value, and a noise value. The calculation of that data from the raw phase measurements is performed by the software library (for details see Chapter 3). The library provides various customization options in order to control the behavior of the early processing steps, including flags to en/disable certain processing steps and processing parameters (e.g., thresholds).

In [90], we evaluated the selected ToF camera's feasibility for parking assistance (see in Chapter 8, Publication 5). An example of a successful deployment of a ToF camera to perform parking assistance is described in [91]. The authors utilize the ToF camera's intensity image as well as its distance image in order to perform parking assistance. Our work, published in [92], presents an approach to perform localization based on the 3D data from ToF cameras and provides an overview of multiple applications utilizing this technology (see Chapter 8, Publication 3).

Radar Sensor

The *RadarLog*[2] development kit was selected as the radar sensor of this work's platform. The development kit is equipped with a 77 GHz *radio frequency* (RF) frontend, capable of performing fast-chirped FMCW measurements. The RF frontend deploys four transmit and 16 receive antennas, which provide a horizontal beamwidth of 76.5° and a vertical beamwidth of 12.8°. The RadarLog houses an FPGA and provides a USB 3 interface to configure the sensor and to receive the raw data. Additionally, the kit provides a trigger input to trigger measurements via an external signal. Figure 4.3b shows a picture of the radar development kit.

[2]RadarLog: a platform for microwave radar data capturing and logging (`http://www.inras.at`).

In contrast to the isolated radar sensor modules offered by the automotive industry, the utilized development kit is able to provide low-level data (i.e., sample data). The transmitted waveform is highly configurable, including parameters like the number of chirps, signal timings, or the signal's bandwidth. Parameters of the analog processing chain can also be adjusted (e.g., sample rate, number of active transmit/receive channels). The sampled data is buffered in the FPGA, extended with an internal timestamp, and transmitted via USB for further processing.

A successful application of a comparable radar development kit mounted on a vehicle is shown in [93]. The authors mounted the radar sensor on a vehicle in order to perform measurements and obtain real-world data. In the presented approach, the radar data is not fused with perception data from other sensors. Another application of a similar radar kit uses machine learning to create an occupancy grid from the radar data [63].

Vision Camera

As the platform's vision camera, the *Basler ace 2*[3] vision camera was chosen, combined with a 4 mm wide-angle lens[4]. The camera houses a Sony IMX392 CMOS color image sensor (1/2.3") with a resolution of 2.3 MP (1920×1200 pixels). Together with the 4 mm lens, the 1/2.3" image sensor provides an approximate field-of-view of 95°(horizontal) and 80°(vertical). The camera comes with a USB 3 interface and an additional GPIO interface, including an input line, which can be used to trigger the camera externally. The Basler ace camera can operate at a frame rate of up to 160 FPS and uses a global shutter mechanism for image acquisitions. Figure 4.3c shows a picture of the vision camera setup (i.e., camera and lens with an applied heat sink).

The Basler ace camera directly provides a Full HD image stream with three 8 Bit color channels. Since the vision camera captures a similar field-of-view as the ToF camera, it provides complementary data of the same scene section. The *pylon camera software suite* provides an *application programming interface* (API) to the camera. This interface can be used to configure and control the camera and to receive the raw image data. The camera allows the dynamic adjustment of the shutter speed and the pixel amplification gain to expose the image correctly. The utilized focus and aperture settings of the deployed lens have to be manually adjusted to the targeted environment.

Additional Sensors

The *SparkFun 9DoF Razor IMU* is used to keep track of the platform's movement. This sensor comes with a three-axis accelerometer, gyroscope, and magnetometer. The sensor can be used to obtain the relative positional change of the platform over a small period of time. The IMU can also be utilized to calculate an estimation of its orientation with respect to the earth's gravitation and magnetic field.

In addition, the analog input of a microcontroller is used to keep track of the platform's battery level. If the battery is low, the system initiates a controlled shutdown to prevent data loss and avoid deep discharging of the battery.

[3]Basler ace 2, a2A1920-160ucBAS (`https://www.baslerweb.com`).
[4]Basler Lens C125-0418-5M F1.8 f4mm - Lenses (`https://www.baslerweb.com`).

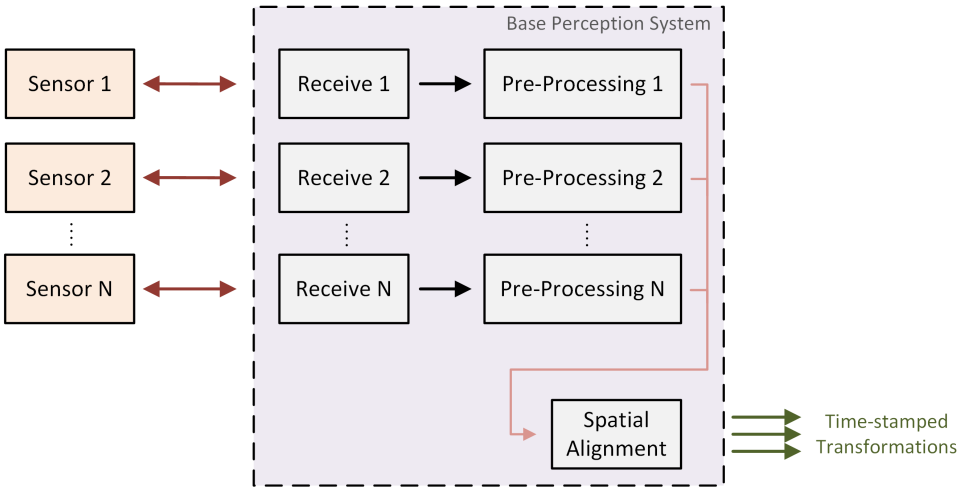


Figure 4.4: Incorporation of the spatial alignment module in the base perception system.

4.2.2 Spatial Alignment

The exact knowledge of the spatial relations between the individual measurements is mandatory to fuse the data into a common representation. Each of the platform's different perception sensors captures data in its own coordinate space, the sensor's individual frame. In order to fuse the data within a common coordinate space, the relative alignment between the single sensors' frames has to be known. The spatial relationships between different frames are represented as coordinate frame transformations. Static transformations remain constant during operation (e.g., between rigidly mounted sensors), while dynamic transformations change during runtime.

The base perception system's spatial alignment module handles the transformations between the single sensors' frames. Figure 4.4 shows an overview of the base perception subsystem with emphasis on the spatial alignment module. The module keeps track of the transformations between all coordinate systems of the subsystem. The temporal course of these transformations is included in the subsystem's output to enable the spatial and temporal alignment of the output data streams by consecutive subsystems. Each transformation update is assigned with a timestamp, denoting the validity period of the corresponding transformation.

The transformations between the single coordinate frames are expressed as rigid transformations. Rigid transformations are transformations, which do not change the distances between any transformed point pair A point, given in coordinate system A, can be transferred into coordinate system B by the multiplication with the corresponding transformation matrix T_A^B, as seen in (4.1). The transformation matrix consists of rotation components and a translation components and can be used to transform points from coordinate system A to B. The individual components of the transformation matrix are presented in (4.2). And as seen in the equation, homogeneous coordinates are used in order to apply the transformation with a single matrix multiplication.

$$\mathbf{p}^{(B)} = \mathbf{T}_A^B \cdot \mathbf{p}^{(A)} \tag{4.1}$$

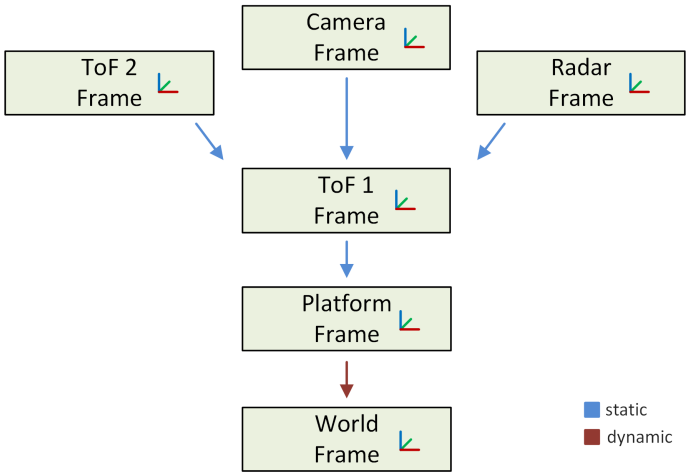


Figure 4.5: Transformations between the single sensors' frames. One ToF camera acts as the reference frame.

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}^{(B)} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}_{A}^{B} \cdot \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}^{(A)} \tag{4.2}$$

The spatial alignment module manages a transformation tree, holding the relative transformations between the tree's connected coordinate frames. Figure 4.5 shows the transformation tree for the different sensor frames of this work's environmental perception platform. The coordinate frames include the individual sensors' measurement frames (e.g., radar frame), as well as certain reference frames of the platform (e.g., platform frame). As seen in the figure, one ToF camera's measurement frame (e.g., ToF 1 frame) acts as the reference frame for all other sensors' frames. Additionally, there exists a global root frame (world frame), defining a reference point for the entire tree. The global reference frame may be set to the ground beneath the platform or to a fixed point within the system's local environment if the platform's positional changes shall be considered.

The transformations between the single sensors' frames and the platform's reference frame are assumed to be static since the sensors are rigidly attached to the platform. These static transformations are determined during an offline calibration procedure and are statically defined in the system's spatial alignment module (via system parameters). The dynamic transformation is continuously re-calculated during runtime and updated in the transformation tree. The system's modules can query the time-accurate transformation between any frames of the transformation tree during runtime.

The coarse transformations between the sensors can be determined manually by measuring the translations and the rotations between the respective sensor pairs. However, more accurate transformations are required in order to meet the requirements of most perception applications. Thus, individual calibration procedures have to be conducted for the connected sensor pairs within the transformation tree.

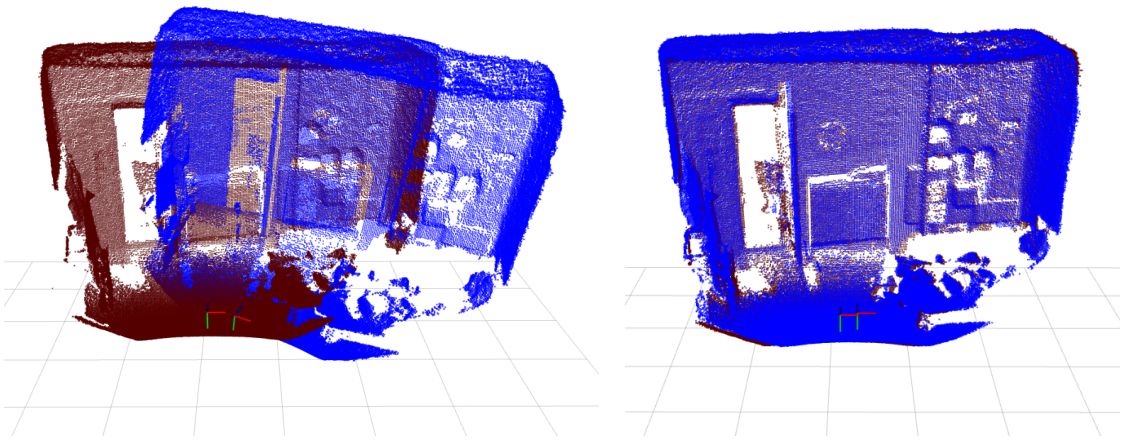
The following static and dynamic transformations are required to build the entire transformation tree and to provide the spatial alignment between arbitrary frames during runtime. The calibration procedures to obtain these transformations are presented in the upcoming subsections.

- **Static transformations**
  The static transformations describe the spatial alignment of the system's coordinate frames, which do not change over time.
  - **ToF 1 → ToF 2**
    The transformation between the reference ToF camera's frame and a second ToF camera's frame, rigidly mounted on the perception platform.
  - **ToF 1 → radar**
    The transformation between the reference ToF camera's frame and the radar sensor's frame.
  - **ToF 1 → vision camera**
    The transformation between the reference ToF camera's frame and the vision camera's frame.
  - **Platform base → ToF 1**
    This transformation describes the spatial alignment between a reference frame on the platform (e.g., a certain point) and the reference ToF camera.
- **Dynamic transformation**
  The system's only dynamic transformation describes the spatial alignment between two coordinate frames, which may be subject to change during operation.
  - **World frame → platform base**
    The transformation between a stationary global reference frame and the platform's reference frame.

Time-of-Flight to Time-of-Flight

The ToF cameras' point clouds are utilized to obtain the relative transformation between the corresponding ToF cameras' frames. This method requires the two point clouds to contain a common region (i.e., an overlapping field-of-view). The points within this overlapping area are aligned to obtain the transformation between the two point clouds. An iterative approach is used, which performs a stepwise improvement, starting from an initial estimation. A popular choice for the initial transformation is a coarse estimation of the transformation using the sensors' mounting positions.

The importance of correctly estimating the transformation between the ToF cameras is shown in Figure 4.6. The alignment of the two point clouds before the calibration method was applied is shown in Figure 4.6a. The utilized transformations of the point clouds are based on a coarse initial estimation. Figure 4.6b depicts the two point clouds after the application of the calibration method to estimate the transformation. As seen by the correctly aligned point clouds, a precise estimation of the transformation was successfully obtained.



(a) Unaligned point clouds. **(b)** Aligned point clouds.

Figure 4.6: Alignment of two ToF point clouds, before and after the calibration.

Since the sensors are mounted rigidly on the platform, it is valid to assume that the transformation stays valid during operation. The calibration has to be performed each time the sensors' alignment has changed (e.g., manual re-arrangement or re-positioning). The calibration task estimates the spatial transformation between the two ToF cameras based on an initial estimation. No dedicated calibration targets are required. The only requirement is that there are enough points present in the acquired ToF point clouds' overlapping regions. For the correct alignment (also called registration) of multiple ToF point clouds, a customized point cloud registration technique is used. The developed alignment algorithm is capable of efficiently and robustly determining an estimation of the transformation. The point cloud alignment process is illustrated in Figure 4.7. As seen in the figure, the algorithm enters an iterative phase after initial pre-processing of the input data streams.

The algorithm is triggered when two point clouds with a similar measurement time are available. Depending on the number of valid points within the point clouds' overlapping region, downsampling is performed. Reducing the number of points speeds up the algorithm and decreases the chance of false alignment caused by small structures. The coarse initial alignment estimation is utilized to extract the overlapping point cloud regions for further processing. The iterative portion of the algorithm starts with the initial guess for the transformation. Next, the point clouds' *fitness* is evaluated, a measure for the distance between corresponding points of both point clouds. If the fitness level is within a certain range, the *iterative closest point* (ICP) algorithm [94] is used to improve the alignment by re-calculating the transformation matrix. If the fitness level is very low, indicating a wrong alignment, the *random sample consensus* (RANSAC) global registration method is applied to obtain a new estimation of the initial alignment, which is then fine-tuned using the ICP algorithm. The process continues until the fitness reaches the desired threshold or the maximum number of iterations is reached. The determined transformation matrix is then saved and can be utilized as an input parameter of the base perception system's spatial alignment module.

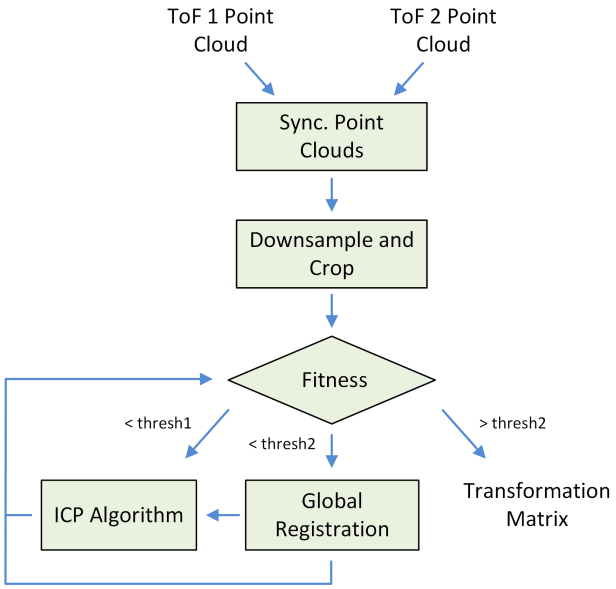


Figure 4.7: Flow chart of the ToF to ToF calibration procedure.

Time-of-Flight to Radar

The radar sensor provides the range-angle information in a top-down representation (x/z coordinates). To correctly align the radar 2D information with the ToF 3D information, the relative 3D alignment between these two frames has to be known. Depending on the desired precision, different approaches can be utilized to obtain the transformation.

Due to the limited absolute range resolution of the radar sensor, a manual estimation of the alignment can be sufficient for a number of applications. The translation between the frames can be approximately determined by utilizing the distance between the two sensors' mounting positions. While the pitch and yaw angle can be estimated from the mounting position, the roll angle offset can be determined when visualizing the data. Although this approach is sufficient in many cases (especially at low ranges), it is typically incapable of providing a highly accurate transformation.

Thus, a semi-automatic approach is introduced, which determines the full 3D position of the radar frame with respect to the ToF frame. The transformation is obtained using the principle of least squares fitting for 3D point sets, as published in [95]. A version of that algorithm is utilized for this work, requiring a minimum number of three corresponding points

A rigid transformation between the two coordinate systems can be obtained by utilizing multiple corresponding points in the different coordinate systems. Two sets of points P and Q contain corresponding points, expressed in the two different coordinate systems A and B, see (4.3).

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_N \end{bmatrix}^{(A)} \tag{4.3}$$
$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_N \end{bmatrix}^{(B)}$$

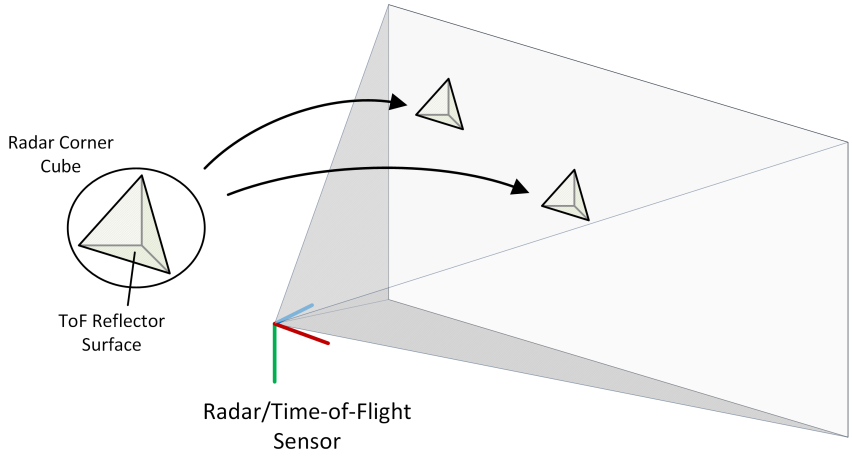


Figure 4.8: Placement of artificial radar targets within the sensors' common field-of-view.

Thus, a transformation $\mathbf{T}_A^B$ is desired which fulfills the relationship stated in (4.4).

$$\mathbf{q}_i = \mathbf{T}_A^B \cdot \mathbf{p}_i \tag{4.4}$$

Due to real-world effects like noise, a transformation $\mathbf{T}$ shall be determined, minimizing the error, as stated in (4.5). Similar to the approach published in [95], a *singular value decomposition* (SVD)-based algorithm is used to obtain an estimate of this transformation matrix.

$$\mathbf{T} = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{i=0}^{N} \|\mathbf{T} \cdot \mathbf{p}_i - \mathbf{q}_i\|^2 \tag{4.5}$$

In order to obtain point correspondences between ToF and radar points, an artificial target is utilized (similar to the targets presented in [96] and [97]). The target consists of a radar corner-cube with a white, highly reflective, and triangular-shaped surface, which can be distinctly detected by ToF and radar. Figure 4.8 shows an illustration of the radar target and how the targets are placed into the common field-of-view of the ToF camera and the radar sensor. The ToF camera determines the center of the triangle, while the radar sensor determines the center of the radar reflector. The radar sensor's detected target distance has to be corrected in order to obtain the corresponding distance at the surface of the target. The artificial target has to be placed in an empty space in front of the platform in order to perform an unambiguous calibration.

Since the radar sensor only provides the (x/z) coordinates of the points, the corresponding measurement points shall be acquired at a y-position of zero. This can be done by varying the artificial target's position at a fixed radar range and angle. The maximum value of the target's response is measured at the peak of the radar lobe, occurring at zero y-position. Figure 4.9 illustrates the search space for different point correspondences in the 3D space. The point correspondences are evaluated using (4.5).

Since a manual variation of the target on the radar sensor's y-axis is a time-consuming task, a software-supported method is used. For this approach, the target's position is

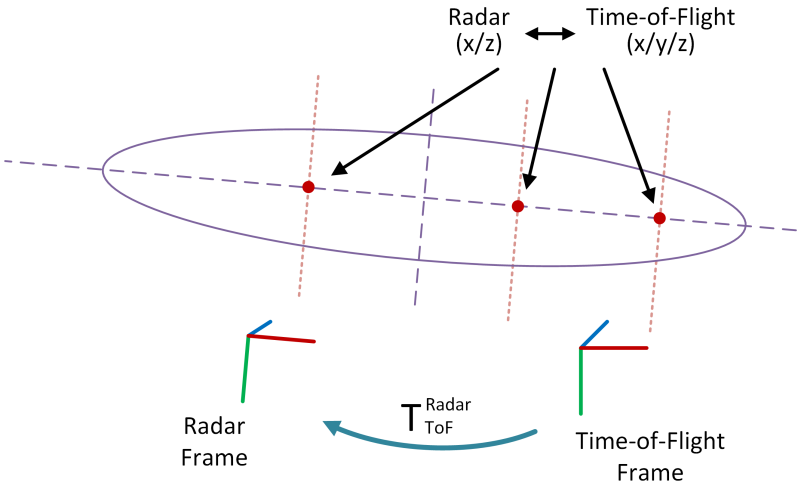


Figure 4.9: Point correspondences between 2D radar data and 3D ToF data.

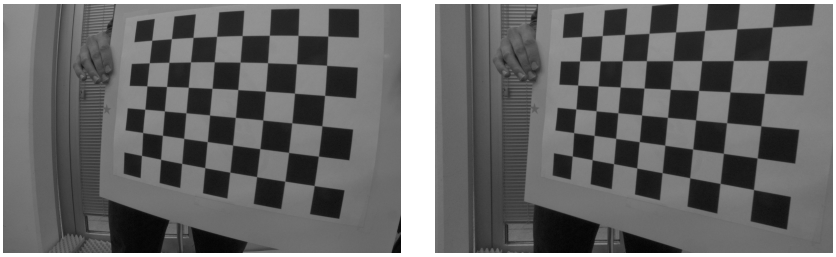
altered around multiple desired corresponding points. The software then automatically detects the maximum reflection power at each position and extracts the associated correspondence point. The set of corresponding points, determined with this method, is then used to estimate the transformation matrix.

Time-of-Flight to Vision Camera

In order to align the ToF camera and the vision camera, both cameras have to be intrinsically calibrated first. The intrinsic calibration reveals the camera matrix and the distortion coefficients of the camera setup for the pinhole camera model. After applying the calibration parameters, the model can be utilized to obtain the pixel positions from the corresponding 3D points in space.

A checkerboard pattern is deployed to determine the intrinsic calibration parameters of both cameras. For that purpose, the checkerboard pattern is placed in the scene at different poses (altering position, distance, and skew). The OpenCV library [98] is utilized to find the checkerboard corners in all images, which are then used to determine the camera's intrinsic parameters. The OpenCV library makes use of the methods presented in [99] and [100]. Figure 4.10a shows an example image of the checkerboard pattern used for the calibration of the camera. The intrinsic calibration parameters can also be used to undistort the images from the camera. Figure 4.10b shows the image's undistorted version from the vision camera after the distortion was corrected.

In order to obtain the relative real-world poses between the two cameras, an extrinsic calibration is utilized. Cameras with known intrinsic parameters can determine the position and pose of the checkerboard (with a given pattern size) in the camera's coordinate system. The checkerboard is placed in the common field-of-view and captured by both cameras. Figure 4.11 shows two pictures of a scene containing the checkerboard pattern, captured with the vision camera and the ToF camera (intensity image). Using this method, the transformation $\mathbf{T}_{ToF}^{CB}$ between the ToF camera and the checkerboard can be obtained, as well as the transformation between the vision camera and the checkerboard $\mathbf{T}_{Cam}^{CB}$. Fig-



(a) Input image. **(b)** Undistorted image.

Figure 4.10: Calibration images from the vision camera, before and after the calibration.

(a) Vision camera image. **(b)** ToF intensity image.

Figure 4.11: ToF and vision camera images, utilized to estimate their relative alignment.

ure 4.12 shows an illustration of these transformations. The overall transformation $\mathbf{T}_{ToF}^{Cam}$is calculated from the two transformations as stated in (4.6). The transformation is determined for multiple positions of the checkerboard pattern (>10) and averaged in order to reduce the uncertainties introduced by the single measurements.

$$\mathbf{T}_{ToF}^{Cam} = \mathbf{T}_{ToF}^{CB} \cdot \left(\mathbf{T}_{Cam}^{CB}\right)^{-1} \tag{4.6}$$

World Frame to Platform Base

The transformation between the global world frame (location fixed) and the platform dynamically changes if the platform is moved during operation. Thus, this transformation has to be continuously determined and updated. The estimated transformation can be utilized for multiple perception tasks, like local positioning or to create a map of the local environment.

An *extended Kalman filter* (EKF) with input data from multiple sensors is utilized to track the platform's movement. The approach is based on the work published by the authors of [101]. The EKF utilizes several sensor inputs to calculate the transformation between the world frame and the platform's frame. The filter incorporates the measurement (un)certainties as well as a dynamic model in order to estimate the current transformation. The inputs to the EKF are the pose change (position and orientation)

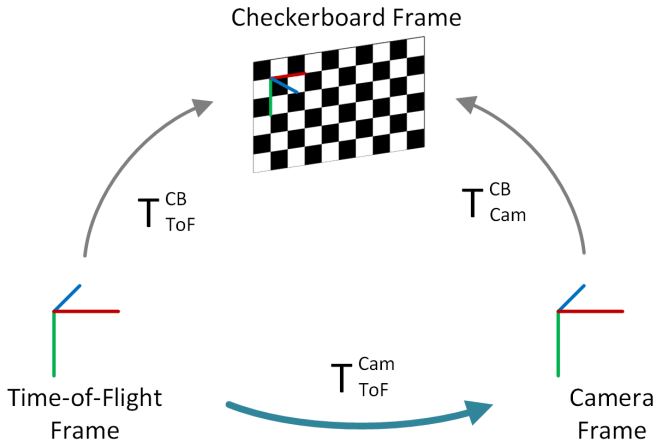


Figure 4.12: Transformations between the sensors' frames and the checkerboard pattern.

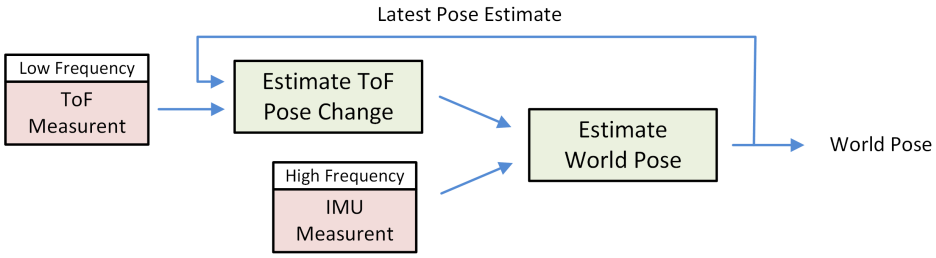


Figure 4.13: Pose estimation process between the world frame and the platform's base frame.

calculated from consecutive ToF point clouds, as well as the linear acceleration and the angular velocity from the IMU sensor (see Figure 4.13).

Since the IMU is operated at a higher measurement frequency (>100 Hz), the EKF is frequently updated using the IMU measurements. However, since the IMU measurements have to be integrated to obtain a position, they introduce a significant error over time (measurement drift). The pose change obtained from two consecutive ToF measurements is more precise but is computationally more expensive and provided at lower rates. Thus, both sensors are combined in order to generate a smooth and accurate estimation of the pose between the platform and the fixed-location world frame.

The pose change between two consecutive ToF point clouds is determined using the ICP algorithm, similar to described in the approach for the ToF to ToF alignment. However, instead of point clouds from different cameras, the point clouds are obtained from the same camera at different acquisition times. The obtained pose change between the two ToF measurements (at time t_0 and t_1) is expressed by the transformation $\mathbf{T}_{ToF}^{ToF(t_1 \to t_0)}$. This transformation is then utilized to update the platform's pose in the global world frame, expressed by $\mathbf{T}_{pl}^{w(t_1)}$. Figure 4.14 shows relationships between the system's transformations for both time points. As seen in (4.7), the desired transformation $\mathbf{T}_{pl}^{w(t_1)}$ can be calculated using the platform's previous pose, the obtained ToF pose change, and the static transformation $\mathbf{T}_{ToF}^{pl}$ between the ToF camera and the platform's base frame.

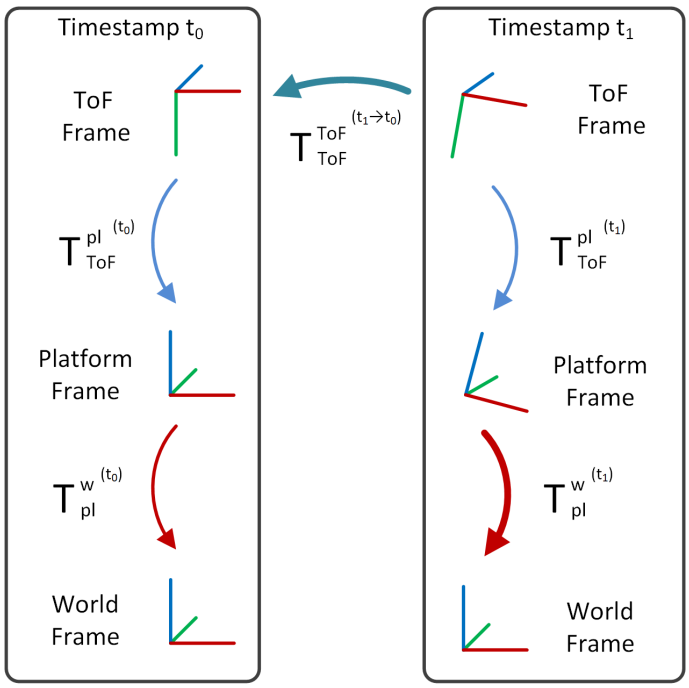


Figure 4.14: Transformation of the ToF camera's pose change projected into the world frame.

$$\begin{aligned} \mathbf{T}_{pl}^{w(t_1)} \cdot \mathbf{T}_{pl}^{ToF(t_1)} &= \mathbf{T}_{pl}^{w(t_0)} \cdot \mathbf{T}_{\mathrm{ToF}}^{pl\ (t_0)} \cdot \mathbf{T}_{ToF}^{ToF(t_1 \to t_0)} \\ \mathbf{T}_{pl}^{w(t_1)} &= \mathbf{T}_{pl}^{w(t_0)} \cdot \mathbf{T}_{\mathrm{ToF}}^{pl\ (t_0)} \cdot \mathbf{T}_{ToF}^{ToF(t_1 \to t_0)} \cdot \left(\mathbf{T}_{ToF}^{pl\ (t_1)} \right)^{-1} \end{aligned} \tag{4.7}$$

The ICP algorithm to determine the ToF pose change uses the Kalman filter's current pose estimate as its initial transformation. Starting the algorithm from this pose estimation increases the speed of the ICP algorithm and prevents misalignment. Since the obtained ToF pose changes also contain inaccuracies, the absolute pose estimation using these relative measurements will drift over time. If necessary, the drift can be compensated using additional methods (e.g., by keeping track of a 3D map) or sensors (e.g., GNSS). However, a valid estimation of the relative transformation within a limited period is sufficient for many applications (e.g., local mapping).

4.2.3 Temporal Alignment

In order to allow a feasible combination of measurements from multiple sensors, the temporal alignment of the single data streams has to be known and available to the fusion module(s). For that purpose, the individual data streams are extended with an accurate estimation of the respective measurement's acquisition time. This is achieved by simultaneously triggering the perception sensors, estimating the common trigger times, and assigning the timestamps to the corresponding data streams when they are received by the system.

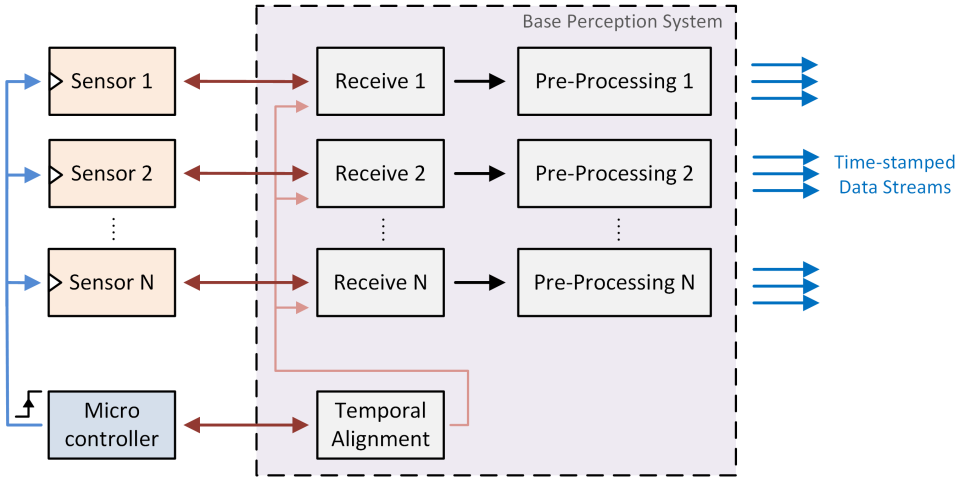


Figure 4.15: Incorporation of the temporal alignment module in the base perception system. Adapted from [102].

Synchronized Data Acquisition

Simultaneous data acquisition is advantageous for the low-level fusion of sensor measurements from multiple sensors. Simultaneously acquired measurements are not affected by the sensor platform's movements and dynamic objects in the scene between the individual sensors' acquisitions. The scene objects (also dynamic) appear in the same position for each sensor. The proposed platform is able to simultaneously acquire measurements by triggering the utilized perception sensors via an external signal.

The synchronization of the individual measurements is achieved by generating trigger signals for each perception sensor via a real-time capable microcontroller. The base perception system's temporal alignment module configures the trigger activities, communicates with the microcontroller, estimates the measurement timestamps, and forwards the determined timestamps to the respective reception modules. Figure 4.15 shows an overview of the base perception system, emphasizing the modules involved in the temporal alignment process.

Radar, ToF, and vision measurements can be simultaneously performed without any interferences or disturbances. However, the simultaneous acquisition of measurements from multiple sensors utilizing the same principle can lead to disruptions. This is the case for the multiple ToF cameras deployed in the proposed perception platform. Even though the cameras implement a technique called spread spectrum illumination in order to counteract the interference, occasional disturbances were observed. In spread spectrum illumination, the cameras vary the utilized modulation frequency during the measurement to prevent disruption from other ToF cameras working at the same frequency. However, if the modulation frequencies are in unfavorable compositions, disruptions can still occur.

A sequential acquisition of the full ToF measurements would introduce a significant delay between the single data acquisitions, too high to be neglected. Thus, a nested trigger mode for the ToF cameras is introduced, enabling an almost parallel measurement acquisition without the drawback of possible interferences. The ToF cameras are triggered sequentially with small delays (duration of a single illumination phase) to avoid possible

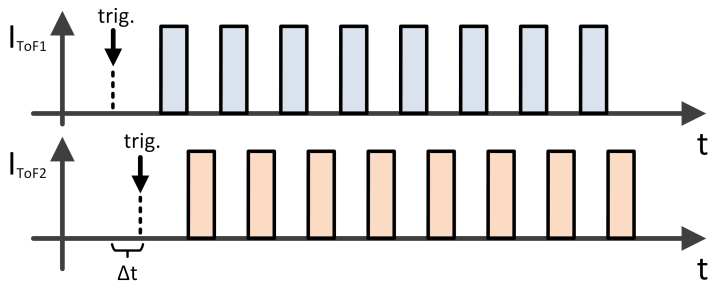


Figure 4.16: Nested triggering of multiple ToF cameras (illumination intensity over time plot). The trigger signals of the ToF cameras are slightly delayed in order to enable an almost parallel mode.

interferences and allow an almost parallel operation. Figure 4.16 shows the temporal course of two ToF cameras' measurements, acquired in nested trigger mode. The readout phases of the first ToF camera are utilized to perform the second ToF camera's illuminations and vice versa. The introduced delay between the measurements (<2 ms) can be neglected for the majority of applications. The radar and vision measurements are triggered at the same time as the first ToF camera.

Timestamp Estimation

Timestamping of individual data streams is crucial in order to enable time-accurate handling of multiple measurements acquired at different times. Figure 4.17 shows the individual delays added to the data streams during the different stages of a triggered data acquisition. After a perception sensor is triggered via an external signal, a measurement is acquired and transferred to the base perception system for further processing. As seen in the figure, the measurement delay includes the sensor-specific measurement duration, the raw data transfer delay, and the reception delay. Pure software timestamping assigns a software timestamp t_{sw} to the individual data streams, based on the time of reception by the processing system. The delays included in the software timestamp include non-deterministic components (e.g., scheduling or communication). Since these delays cannot be robustly compensated, software timestamps are often not able to meet the requirements of time-critical applications.

This work's approach utilizes a novel method, which enables an accurate estimation of the exact measurement timestamp $\hat{t}_{meas}$, based on the hybrid software timestamp t_{hsw}. One of our publications [102] presents the details of the hybrid timestamping approach and is included in Chapter 8 of this thesis (Publication 9). In the presented approach, an external microcontroller sends a low-latency notification message to the trigger module, each time a trigger signal is generated. The system then estimates the measurement time based on the reception time of the notification message. With the help of sequence numbers, the estimated measurement time is then assigned to the raw data streams. As seen in Figure 4.17, the short delay included in the trigger notification message can be estimated with adequate precision. All output data streams originating from the corresponding trigger event are then extended with the estimated measurement timestamp.

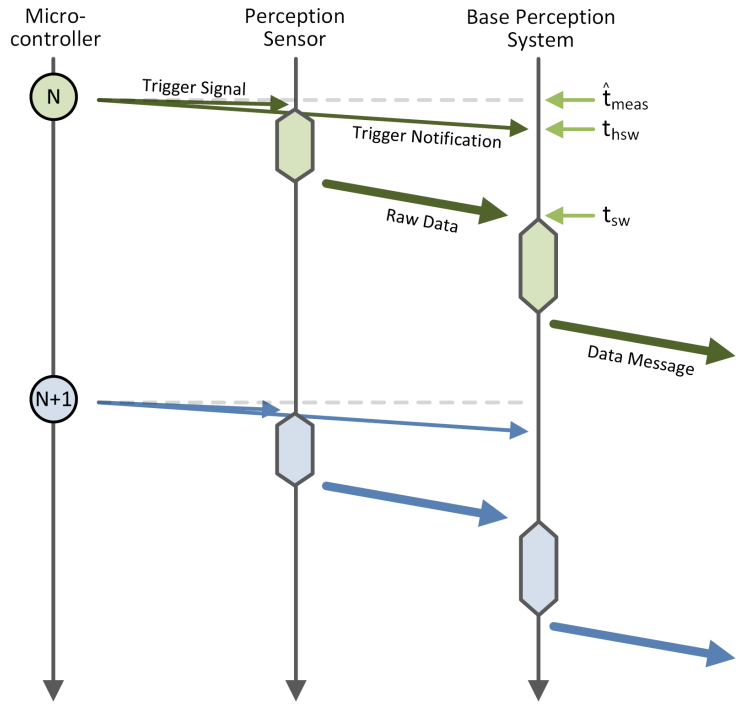


Figure 4.17: Sequence chart of a triggered data acquisition. The chart shows how a trigger notification message is utilized to timestamp the received measurement data. Obtained, with changes from [102].

4.2.4 Sensor Data Processing I: Low-Level Sensor Interface

Each perception sensor's receive module is in charge of receiving the associated sensor's output data and handling the sensor's configuration. The receive modules perform a sensor-specific re-structuring of the data in order to provide organized data streams to succeeding modules. Each receive module outputs a measurement data stream and one or multiple metadata streams, represented as standardized messages. These output streams are available for further processing in other modules of the base perception system and other subsystems. The two types of data streams provided by the receive modules are composed of multiple fields (e.g., data, timestamp).

- **Measurement data stream**
  The fields typically included in the measurement data stream are listed below.
  - **Sensor output data**
    The sensor output data is received from the sensor (e.g., sample points) and highly depends on its configuration. Certain data manipulation steps may have already been applied by the sensor's hardware or by the utilized receive library.
  - **Sensor frame**
    The sensor frame identifier defines the coordinate system in which the measurement data was acquired (e.g., radar frame). This identifier is defined individually for each perception sensor.

  - **Timestamp**
    The timestamp field describes the estimated measurement time of the received measurement data. The estimated timestamp is made available by the temporal alignment module. The receive module adds the timestamp to the corresponding measurement data stream.

- **Metadata stream**
  The fields typically included in the metadata stream are listed below.

  - **Metadata**
    The metadata contains different types of supportive information for the measurement data (e.g., module parameters or calibration values). Since these parameters add expressiveness to the provided measurement data, the knowledge of this additional information can be highly beneficial for later processing steps.
  - **Sensor frame**
    The sensor frame field consists of an identifier, defining the corresponding measurement's coordinate frame (e.g., radar frame).
  - **Timestamp**
    The timestamp defines the acquisition time of the metadata stream's corresponding measurement. This field is crucial for the unambiguous assignment of a metadata stream to the corresponding measurement stream.

The messages are made available to the pre-processing modules and upcoming subsystems (e.g., use cases). Subsequent processing modules can make use of the messages' timestamps in order to synchronize the metadata and measurement data streams. Upcoming modules can utilize the metadata streams to perform enhanced processing or as supportive information to optimize system parameters (e.g., sensor configuration parameters). The low-level interface, the shape, and the structure of the provided sensor data vary significantly for the different sensor types. Hence, the measurement data and metadata streams provided by the receive modules are highly sensor-dependent. Figure 4.18 shows the simplified composition of the output messages provided by the sensors' receive modules.

Time-of-Flight

The ToF camera's receive module outputs a measurement data stream and two metadata streams. The ToF camera provides the raw measurement data via its sensor interface. The reception library can be configured to perform a number of error correction steps and filtering steps to the raw data before it is handed over to the user program. Additionally, the library can utilize the camera and lens parameters to calculate the point cloud from the raw data efficiently. The provided point cloud contains entries for each point's (x/y/z) position, intensity, noise, and confidence. This point cloud is converted into a standardized message, also containing the corresponding ToF camera's coordinate frame identifier and the estimated measurement timestamp. The ToF receive module's parameters, relevant for further processing, are included in a metadata message and made available to succeeding

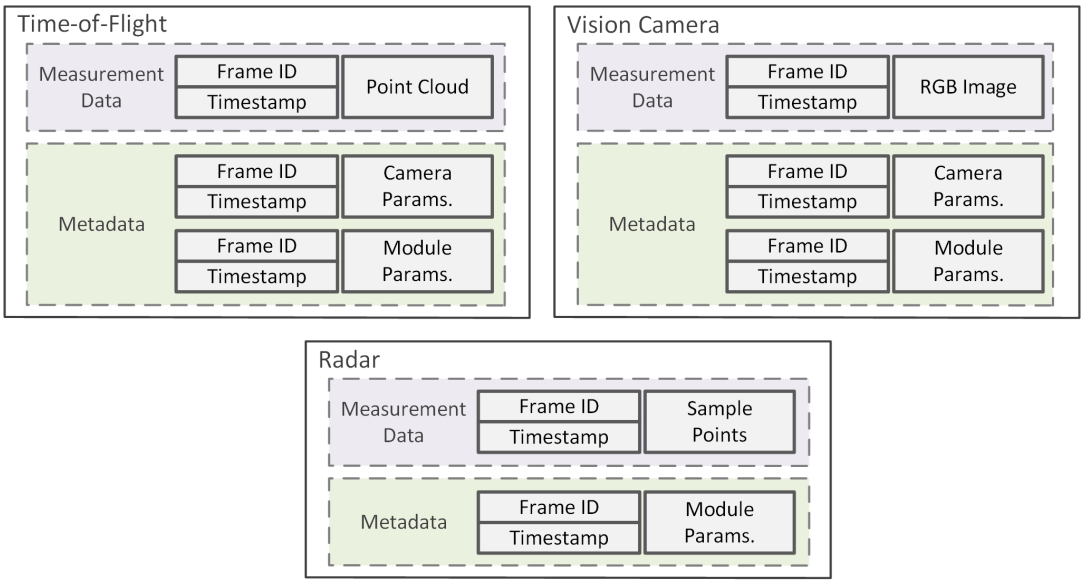


Figure 4.18: Output data streams from the perception sensors' receive modules.

modules (e.g., illumination time, processing flags). A second metadata stream contains information on the camera/lens setup. This includes the intrinsic parameters determined during an earlier calibration procedure.

Radar

The radar sensor's receive module outputs a measurement data stream and one metadata stream. The array of sample points for each receive channel is re-structured and converted to a one-dimensional array. The measurement data stream consists of this raw data array, the radar sensor's frame identifier, and the estimated measurement timestamp. The radar receive module's active parameters, relevant for further processing, are packed into a metadata message and made available to succeeding modules.

Vision Camera

The vision camera's receive module outputs a measurement data stream and two metadata streams. The camera provides an uncompressed Full HD color image of the scene. The measurement data stream contains this image, the camera's frame identifier, and the estimated measurement timestamp. One metadata stream contains a subset of the receive module's parameters as supportive information for succeeding processing tasks. The other stream consists of metadata of the utilized camera/lens setup (e.g., intrinsic parameters).

4.2.5 Sensor Data Processing II: Sensor Pre-processing

The receive modules' output messages are utilized by the pre-processing modules, which re-structure and process the data streams. Each pre-processing module performs a customizable, sensor-dependent amount of manipulation steps and outputs multiple data streams at different abstraction levels. Early pre-processing of the data can be beneficial

since it lowers the communication load between the base perception system and upcoming subsystems (i.e., the use cases). Although any processing stands in contrast with low-level sensor fusion, early non-destructive processing of the sensor data can be beneficial for later low-level fusion. The early pre-processing tasks can be performed with minimum overhead in the course of the raw data conversion into standardized messages.

The behavior of the pre-processing modules can be highly customized via the corresponding modules' parameters. Unused outputs and their associated processing branch(es) can be disabled, while the processing parameters can be adapted to fit the desired use case. This subsection provides an overview of the major pre-processing steps and the provided output data streams of the different perception sensors' pre-processing modules.

Time-of-Flight

The ToF pre-processing module receives the ToF point cloud from the reception module. Further processing is performed on the input point cloud to provide the following output streams:

- Subsampled point cloud
  A voxel grid filter is used to subsample the point cloud and significantly reduce the data size. The size of a voxel and the minimum amount of points within a voxel is configurable via parameters.

- Cropped point cloud
  An (x/y/z) bounding box is used to filter the point cloud in order to remove areas out of interest. This is beneficial for considering points with the same height as the ego-vehicle.

- Amplitude and distance image
  The amplitude and distance image are extracted from the full point cloud and provided as additional data streams. These images can be beneficial for certain processing steps (e.g., segmentation) and add additional flexibility for further processing.

Figure 4.19 illustrates the input and output data streams of a ToF camera's pre-processing module. The output data streams are assigned with the same measurement timestamp as the input data stream and the corresponding frame identifiers.

Radar

The radar pre-processing module's input data is the raw data cube, containing the samples points of the frequency chirps for all receive channels (as described in Chapter 3). This data cube is processed to provide the following four data streams to succeeding modules and subsystems.

- Range-Doppler image
  A 2D array containing the radar response in the range-velocity domain, useful to detect moving targets in the scene. The resolution of the image depends on the waveform parameters (e.g., number of chirps, bandwidth), as well as certain processing parameters (e.g., FFT bins).

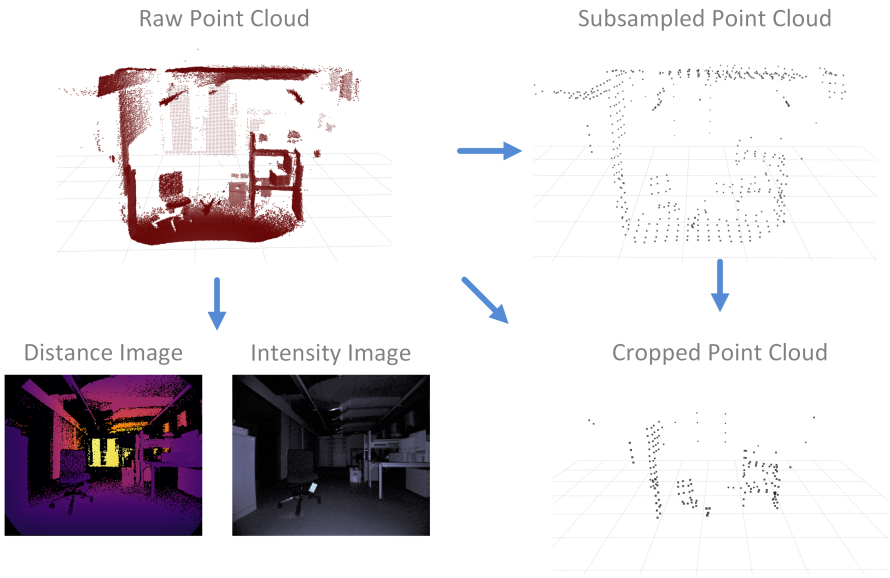


Figure 4.19: ToF pre-processing module: input and output data streams.

- Range-angle image
  A 2D array containing the radar response in the range-angle domain. This representation is beneficial to obtain the angular positions of detected targets.

- CFAR image
  The CFAR image holds the range-Doppler image after applying an adaptive threshold. The identified peaks can be used as a base for further tasks like object detection or tracking.

- Peak list
  The detected peaks of the CFAR image are utilized to obtain a list of radar peaks. The list contains each detected target's top-down position, radial velocity, and reflection magnitude.

An illustration of the radar pre-processing module's input and output data streams is shown in Figure 4.20. A 2D FFT is applied to the raw data cube in order to obtain the range-Doppler images of the scene. An additional FFT is applied to obtain the range-angle images. A possible method to compress the range-angle data cube and to prepare it for further processing is to remove the velocity information and to generate the *overall range-angle image* of the scene. The $I \times K$ range-angle bins of the map are composed of the maximum value over all J velocity bins. Thus, all peaks in the range-angle dimension are preserved, independent of their velocity (see Figure 4.20).

One way to extract target information from the radar data is based on the range-Doppler image. First, the range-Doppler images from all receive channels are summed up, and a CFAR peak detection is performed to detect the peaks. These peak points are then provided to a beam-forming algorithm in order to obtain the corresponding angle values. All detected peaks are stored in a list containing the (x/z) position, the velocity, and the magnitude of each target.

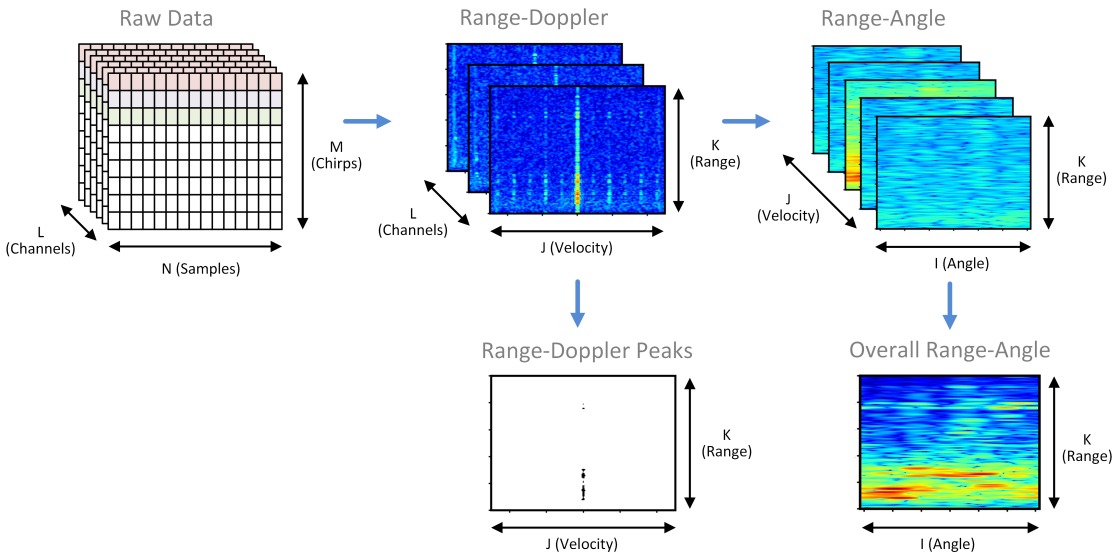


Figure 4.20: Radar pre-processing module: input and output data streams.

The four different output data streams are packed into standardized messages and provided to upcoming modules. All of these messages contain the estimated measurement timestamp and the frame identifier of the corresponding coordinate system. The pre-processing module utilizes the metadata stream from the receive module to interpret the input data and adjust the processing parameters accordingly. Additionally, the behavior of the pre-processing module is highly configurable via the module's parameters.

Vision Camera

The input to the vision camera's pre-processing module is the uncompressed Full HD color image stream from the receive module. This input data stream is pre-processed, and the following two output streams are provided to upcoming modules.

- Undistorted image
  The Full HD color image with corrected lens distortion, utilizing the intrinsic camera parameters. The corrected image is more feasible for further processing and allows the application of simplified camera models.

- Subsampled image
  A color image with reduced size, obtained by subsampling the Full HD image. The reduced resolution leads to a decreased communication time but causes the loss of image details. The subsampling factor is configurable via the module's parameters.

The vision pre-processing module's input and output data streams are illustrated in Figure 4.21. The distorted input image is corrected, utilizing the camera/lens parameters in the associated metadata stream. The rectified image is then subsampled with a custom factor in order to reduce the file size. The rectified image (full resolution) and a subsampled version are provided to subsequent modules via standardized messages. The messages also include the estimated measurement timestamp and the vision camera's frame identifier.

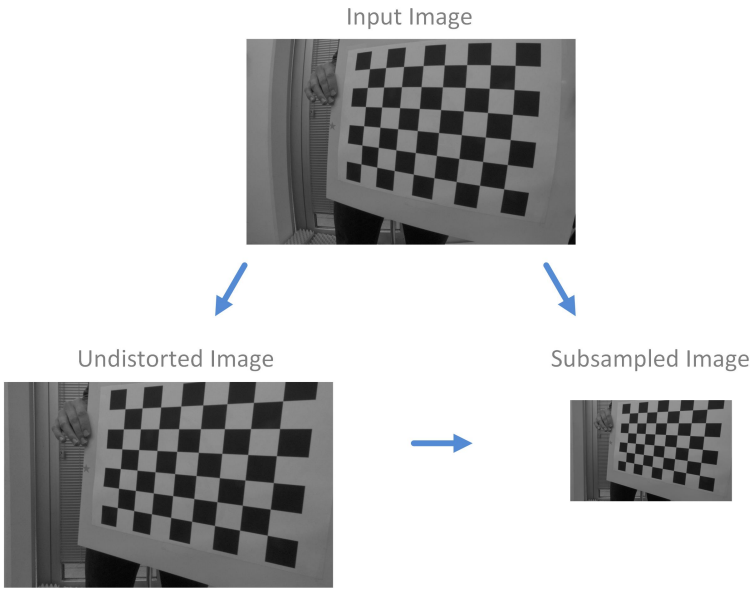


Figure 4.21: Vision camera pre-processing module: input and output data streams.

4.2.6 System Parameters

The behavior of the base perception system's modules can be controlled via system parameters. These parameters consist of two major categories: sensor parameters and processing parameters. Sensor parameters define the sensors' configuration settings (e.g., illumination time, sample rate) and control the sensors' individual behavior. These parameters are managed by the receive modules and are utilized to configure the sensors at startup. The processing parameters control the behavior of the base perception system's individual modules. These parameters are used to control the data flow, enable and disable certain processing branches, and define algorithm parameters. The applied system parameters have a high impact on the composition and the quality of the base perception system's provided output streams. Thus, the values have to be selected in compliance with the perception platform's target environment.

The system parameters are assigned to initial values during the system's startup and allow dynamic adaption during runtime. The initial parameters are defined in a global configuration file, queried by every module at startup. In addition, a subset of the modules' parameters can be dynamically adapted during operation. Figure 4.22 illustrates the base perception system's modules and their ability to dynamically adjust parameters. The base perception system's modules provide interfaces, which can be utilized to propose new parameter values during runtime. The corresponding module adopts newly proposed parameter values after a basic examination (e.g., range check).

Table 4.1 shows an excerpt of the base perception system's parameters for each of the three utilized perception sensors. As seen in the table, the adjustable parameters include highly influential sensor configurations and processing parameters. The optimal parameter values depend on the system's current environmental context state and the demands of the active use case(s). Thus, the selection of feasible parameters is crucial in order to provide a high perception performance.

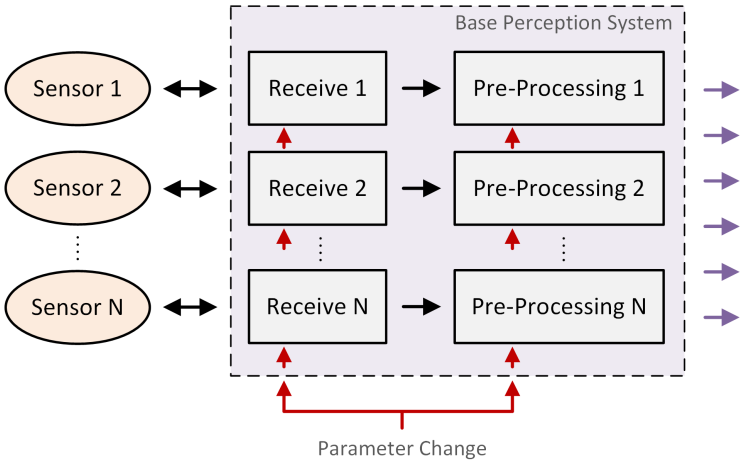


Figure 4.22: Interfaces for the adaption of the base perception system's parameters.

Table 4.1: Selected parameters of the base perception system's modules.

Time-of-Flight	Radar	Vision
Exposure time	Start frequency	Exposure time
Global binning	End frequency	Gain
Operation mode	Number of samples	Gamma
Frame delay	Frame delay	Frame rate

4.3 Use Cases

This section introduces multiple applications, making use of the data streams provided by the base perception system. A use case is a subsystem, which utilizes the base perception system's structured output information to fulfill its desired functionality. As seen in Figure 4.23, use cases typically consist of multiple fusion and processing modules. A fusion module combines different input data streams into a common representation. The extent and the complexity of the fusion step depend on the targeted use case and the selected input streams. One or multiple processing modules utilize the fused data in order to provide the desired use case's functionality. Additional processing modules may be present at arbitrary positions within the use case in order to prepare data streams for upcoming modules.

Each of the use cases takes a subset of the base perception system's output data streams as input. The selection of the data streams depends on the desired task of the corresponding use case. The three main categories of data streams provided by the base perception system are listed below.

- Measurement data streams
  A measurement data stream from the corresponding perception sensors. This may be the sensor's output data provided by the receive module or an already processed version from the pre-processing module. The associated frame identifier and the estimated measurement timestamp are included in the header of the data message.

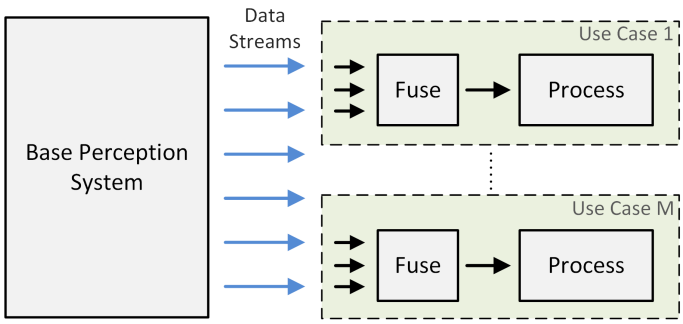


Figure 4.23: Interaction between the base perception subsystem and the use-case subsystems.

- Metadata streams
  A data stream containing supportive data to the measurement data stream. This data stream may contain calibration values or the current sensor configuration of the corresponding measurement. The stream also contains the frame identifier and the measurement timestamp.

- Transformations
  In contrast to the other two categories, the transformations are not directly made available as streams. They are provided to the system via a globally available buffer. This buffer enables upcoming subsystems to query time-accurate transformations for the currently processed data. Some of the available transformations are static (e.g., between rigidly mounted sensors). Others change dynamically and have to be updated continuously (e.g., between the platform and the world frame).

4.3.1 Context-Aware Parameter Adaption

A static selection of the system parameters does, in general, result in a poor perception performance in changing environments. To guarantee a satisfactory perception performance, the system parameters have to be dynamically adapted to the environmental state. This use case has the goal to automatically adapt the perception platform's parameters during runtime. The subsystem considers the system's current state (e.g., measurement data streams) and uses data from context sensors (e.g., IMU, rain sensor, light sensor). The context-aware parameter adaption enables the system to react to changing environmental conditions, which would otherwise lead to a decreased perception quality. For example, a system might benefit from a high frame rate in dynamic scenarios, while the focus on a high precision may be advantageous in static scenarios.

Figure 4.24 shows an overview of the use case's modules. The parameter handlers determine whether certain parameters within the base perception system shall be changed and calculate values for the adaption. The proposed parameter values are passed on to the base perception system's modules. These parameters are then inspected by the corresponding modules and, if feasible, changed in the module. One parameter handler is in charge of one or more parameters, but any parameter may only be changed by one handler. A subset of the available data streams is additionally pre-processed before being

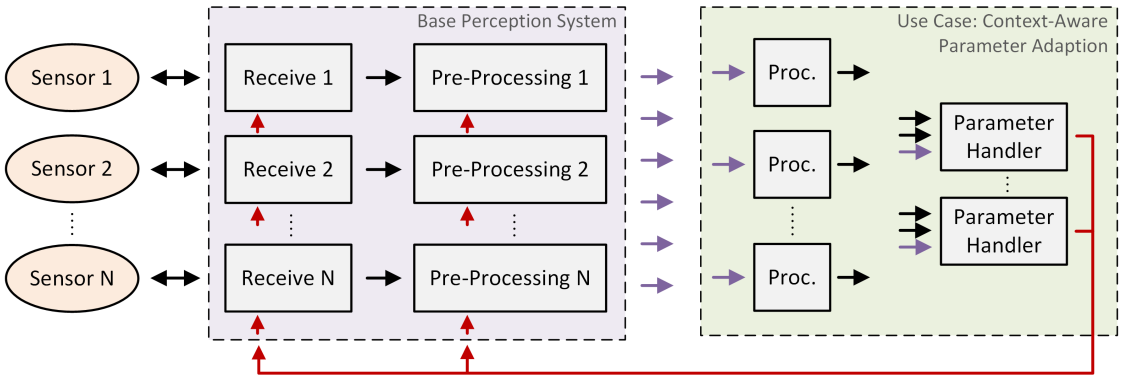


Figure 4.24: Context-aware parameter adaption: overview of the use case and its interaction with the base perception system.

passed to the parameter handlers, while other data streams are directly provided to the parameter handlers.

In addition to adapting parameters of the base perception system, this subsystem is also able to adapt parameters of other use cases' modules. One possible application is to limit the confidence range of a sensor's measurement data stream based on the system's environmental context state. Certain environmental conditions affect the perception sensors' data quality, leading to faults in the perception output if not handled by the system. For that purpose, dedicated parameter handlers are used to degrade a perception sensor's measurement data stream by adjusting its confidence value in the associated fusion module. The fusion modules can make use of the additional information in order to discard the sensor values outside the high-confidence range.

The parameter handlers determine the proposed parameter values by using models, heuristics, and system knowledge. Parameters are only changed if the estimated gain (e.g., an increase of the perception quality) exceeds the cost (e.g., duration, processing effort). Certain parameters are adapted in multiple iterative steps. In that case, the resulting impact of the previous adjustments is included in the determination of the iterative parameter updates.

4.3.2 Obstacle Detection

This use case detects obstacles in the perception sensors' common field-of-view. The goal is to recognize free and occupied areas in front of the platform. In the robotic/automotive context, occupied spaces are not drivable and have to be avoided in the path-planning process. Occupancy grids are commonly used to represent free and occupied space within a robot's/vehicle's local environment. In this use case, an obstacle grid is created by fusing data from multiple perception sensors into a common representation. The obstacle detection module then utilizes the occupancy grid in order to locate obstacles in front of the platform.

Figure 4.25 shows an overview of the obstacle detection's data flow. The fusion module inputs the subsampled and filtered ToF point cloud(s) and two radar data streams at different abstraction levels (range-angle image and peak list). If multiple ToF cameras are utilized, the individual point clouds are combined before the fusion with the radar data.

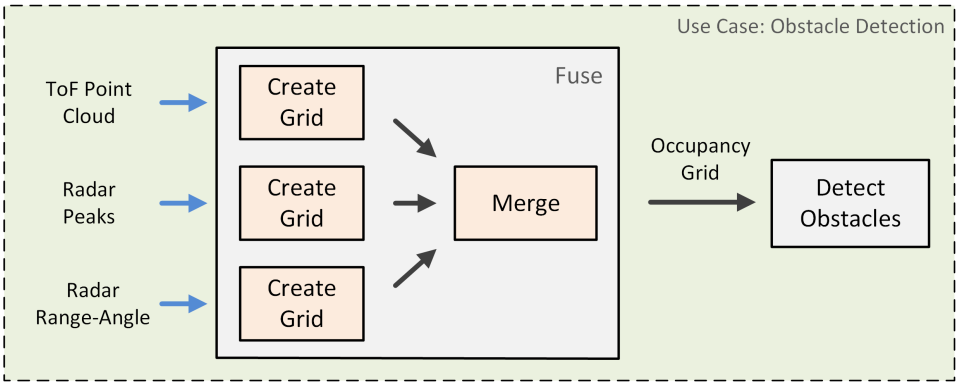


Figure 4.25: Obstacle detection: overview of the use case's main modules and the data flow.

The ToF pre-processing chain is configured to perform z-filtering of the point cloud in order to discard points originating from the ground floor. Individual grid structures are created for each input stream, assigning the likelihood of occupancy to all grid cells. These individual grids are then fused into a common occupancy grid, holding a probability of occupancy for each of the cells. The fusion algorithm considers the individual grids and a combined grid structure to create the overall occupancy grid.

The obstacle detection module considers the last N occupancy grids in order to determine whether a cell is occupied or not. The module applies a threshold to the integrated occupancy grids in order to find occupied cells. The output of the obstacle detection module is a 2D occupancy grid. We published a more detailed description of this occupancy grid creation approach in [7] (see Chapter 8, Publication 6).

4.3.3 Environment Mapping

The environmental mapping use case utilizes data from the ToF cameras and an IMU sensor to create a 3D map of the environment. The use case requires the dynamic transformation between the platform and a global coordinate system (world frame) to be known at all included measurement times. This transformation and the point clouds from one or multiple ToF cameras are then utilized to create a map of the environment while the platform is moving. The base perception system's spatial alignment module can be configured to constantly determine this transformation and include it in the provided transformation tree. In this case, the spatial alignment module utilizes an EKF in order to fuse ToF and IMU sensor with the goal to estimate the pose changes of the platform.

An overview of the environmental mapping subsystem is depicted in Figure 4.26. The fusion module transforms the ToF point cloud(s) into the world frame and combines the point cloud(s) with the latest map. The mapping module keeps track of a common environmental map and outputs it as a point cloud. In order to reduce the file size of the output map, subsampling of the ToF point clouds can be performed on the input point cloud(s) as well as on the continuously updated map (e.g., using a voxel grid filter). Stationary periods are recognized in order to disable the extension of the map if the platform is not moving.

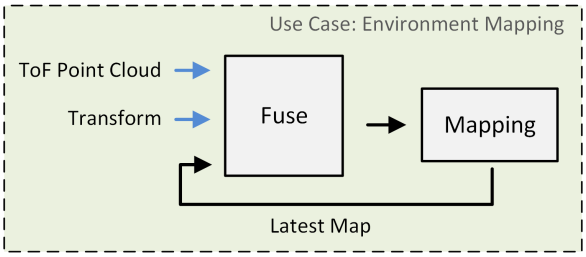


Figure 4.26: Environment mapping: overview of the subsystem's data flow.

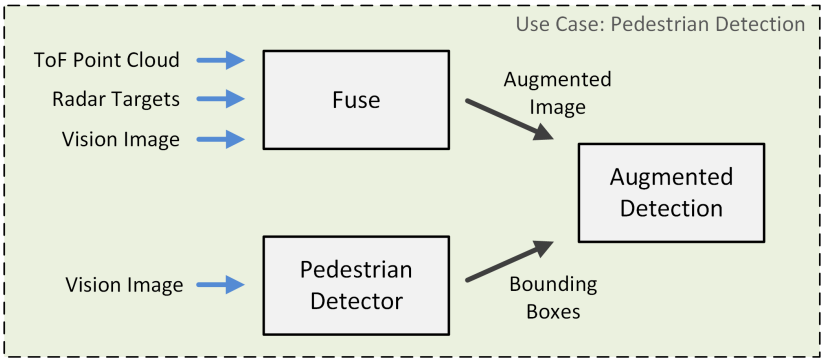


Figure 4.27: Pedestrian detection: overview of the use case's major modules and their interactions.

4.3.4 Pedestrian Detection

The pedestrian detection use case is able to detect human persons within the perception sensors' common field-of-view. A simplified flow chart of the pedestrian detection use case is depicted in Figure 4.27. The use case takes the data streams from the vision camera, the radar sensor, and one ToF camera as input and synchronizes them using their timestamps. The range measurements from multiple perception sensors are combined to enhance the performance of a vision-only pedestrian detection flow.

The range data is projected onto the vision camera's image by utilizing the spatial alignment between the sensors and the vision camera's intrinsic parameters. The 3D points from the ToF camera are added to corresponding positions on the vision camera's image as single dots. The radar targets are projected onto the image in the form of vertical lines since the elevation angle of the radar targets is not available. The implemented pedestrian detection approach uses the vision camera's image to perform a coarse, low-threshold detection of regions within the image, which potentially contain pedestrians. For this step, a HOG-based classifier is utilized with a low detection threshold. A window of multiple scales is shifted over the image in order to detect bounding boxes of potential pedestrians at different positions and distances. The range data from the other perception sensors is then utilized to amplify the bounding boxes and to obtain the most confident ones. These bounding boxes are then marked as pedestrians in the final image. We published a more detailed description of this approach in [102]. The published conference paper is also included in Chapter 8 of this thesis (Publication 9).

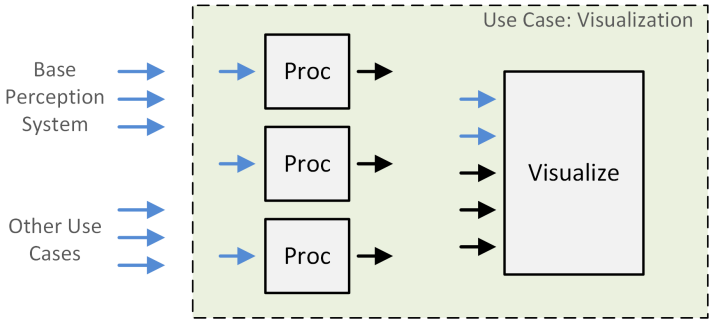


Figure 4.28: Data visualization: overview of the use case's data flow.

4.3.5 Data Visualization

This use case performs the visualization of data streams from the base perception system as well as from other use cases. The visualization module uses the transformation tree, the data stream's timestamp, and the data stream's frame identifier to correctly visualize the data in a common coordinate space. A human-interpretable visualization of the data streams is beneficial for evaluating the system's perception quality and general performance of the overall system and its components.

The visualization subsystem is highly configurable and can be adapted to the environmental perception system's active composition (e.g., the active use case). While some data streams can be directly visualized, others require additional processing in order to convert them into a visualizable representation. Only the relevant data streams of the platform's targeted subsystems shall be prepared for visualization. Thus, predefined configurations were composed for each of the environmental perception system's subsystems (e.g., base perception system, pedestrian detection use case). Each of these predefined configurations activates the required processing and the visualization of the relevant data streams for the associated subsystem. In addition to these predefined configurations, the parameters for the visualization can be further customized. The exact composition of the subsystem (e.g., disabling certain modules) can be individually refined, depending on the desired output.

Figure 4.28 shows the processing flow of the visualization use case. Some data streams have to be pre-processed since they are not directly visualizable (e.g., radar raw data cube). In such cases, the corresponding data stream is converted to a visualizable representation (e.g., using points and markers). The figure also illustrates the ability to visualize data streams from other use cases. This enables the visualization of high-level data from other use cases in combination with low-level data streams from the base perception system.

Well-suited data stream types for the visualization include ToF point clouds, vision camera images, and radar target lists. Point cloud data streams can be directly visualized in the 3D coordinate space. The intensity/confidence can be color-coded, while the (x/y/z) values and the associated transformation define each point's spatial position in the scene. Image streams can be directly visualized as color or grayscale images. Depending on the image type (e.g., resolution, data depth), the image requires the conversion into a visualizable format. Targets at specific positions can be visualized with points/markers in the 3D space using the corresponding (x/y/z) positions. The magnitude of the target's reflection can be indicated via the size or color of the utilized point/marker.

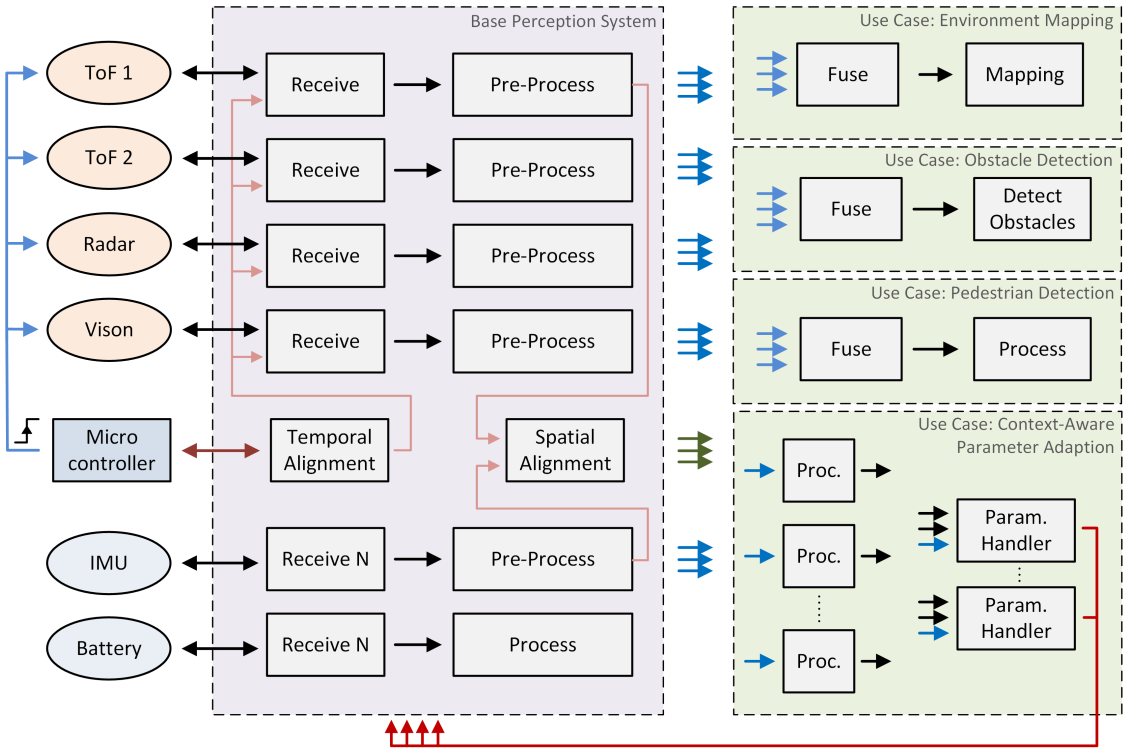


Figure 4.29: Final design of the environmental perception platform's software architecture.

4.4 Final Design

The final design of the environmental perception system consists of the base perception subsystem and multiple use-case subsystems. Figure 4.29 shows a simplified overall illustration of the final system architecture. The base perception system provides all data streams of its comprised modules to the use cases for further processing. Individual use cases subscribe to a subset of these streams in order to perform application-specific processing. The use cases environment mapping, obstacle detection, and pedestrian detection demonstrate the system's abilities, based on the structured output data from the base perception system. A customizable subset of the system's data streams can be utilized for direct human interpretation via the visualization use case (omitted in Figure 4.29).

All data streams are assigned with a measurement timestamp and a frame identifier, allowing their correct spatial and temporal alignment. The spatial alignment module maintains a transformation tree, holding the temporally accurate pose information of the platform's single frames. The temporal alignment module triggers the sensors, estimates their acquisition timestamps, and makes them available to the reception modules. The behavior of the base perception system and the use-case subsystems can be customized via system parameters. These parameters can be defined via a global configuration file and are provided to the system at startup. Additionally, the modules provide interfaces to request parameter changes during runtime. The context-aware parameter adaption use case is capable to dynamically adapt a subset of the system parameters (including sensor configurations) during runtime.

Chapter 5

Implementation

This chapter presents the implementation details of the environmental perception platform. First, the selected strategy to accomplish the implementation task is introduced, and the utilized software tools and hardware compositions are presented. The main part of the chapter describes the ROS-based implementation of the system in order to meet the elaborated requirements.

5.1 Development

An agile method was selected to perform the platform's development, enabling flexible recompositions of the planned development phases depending on the system's performance. Due to the novel composition of the platform and the absence of related work, an iterative bottom-up approach was selected for the development of the perception platform. In the utilized approach, the system's performance was continuously evaluated and considered for the decision about upcoming hardware and software modifications. This section introduces the agile development workflow and presents the main software tools used during that process.

5.1.1 Workflow

The development workflow consists of multiple modular implementation phases, enabling dynamic decisions about upcoming phases. The five major phases of the iterative implementation workflow are listed below.

- **Compose platform**
  In this task, the hardware composition of the platform is extended or modified. This includes the modification of the hardware structure, adding components, or re-aligning sensors. This development phase compromises the initial platform setup as well as later modifications.
- **Update base perception system**
  Within this phase, the software of the base perception system is updated. One task is the adaption of the subsystem to hardware modifications of the platform (e.g., newly added sensors). Another task of this phase is the improvement of existing modules (e.g., adding new functionality, fixing issues).

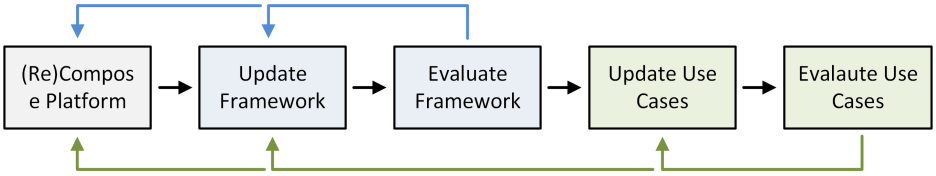


Figure 5.1: Development workflow.

- **Evaluate base perception system**
  The perception performance of the base perception system is evaluated in order to decide whether the system meets the targeted requirements. This includes inspection of the data quality, processing duration, data alignment, and the interaction of the components in various environmental situations. The evaluation results are considered in order to decide about the necessity and the priority of additional hardware and software modifications.

- **Update use cases**
  During this phase, the implementations of certain already existing use cases are updated, or new use cases are added to the system. One reason to update use cases is a modification of the base perception system's provided data structure (e.g., hardware/software modifications). Other reasons include the improvement of existing use cases (e.g., algorithm improvements, fixing issues) and the implementation of additional use cases.

- **Evaluate use cases**
  After new use cases were added or existing use cases were modified, the corresponding subsystems have to be evaluated. The evaluation criteria include the use cases' performance and robustness in varying environmental conditions. Based on the obtained performance, possible modifications of the platform's hardware and software modules are elaborated.

Figure 5.1 illustrates a coarse overview of the development workflow and points out the major iteration loops during the implementation process. The utilized bottom-up approach allowed the rapid development of a first prototype and enabled the continuous refinements based on real-world evaluations. Additional hardware and software requirements were derived during the evaluation steps of the base perception system and the use cases. After the implementation of these requirements, the iterative workflow loop was continued by re-evaluating the system's performance. After minor refinements, only the associated modules were evaluated, while a complete re-evaluation of the system was performed after major changes.

5.1.2 Tools

Multiple software applications and tools were utilized for the implementation of the environmental perception platform. This section provides a short overview of the most extensively used components during this phase.

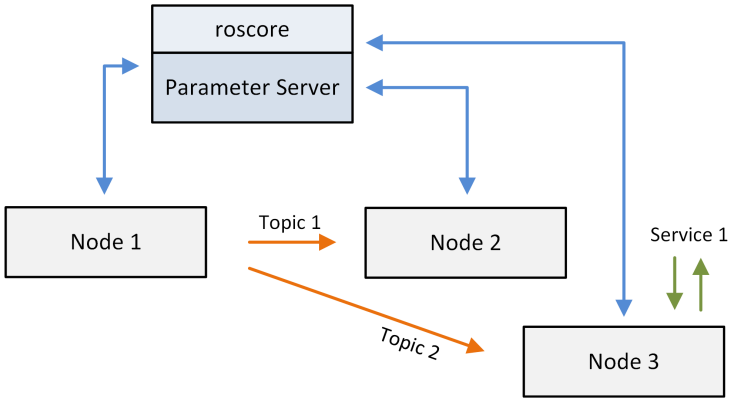


Figure 5.2: Main components of a ROS-based application.

Robot Operating System

The ROS is a commonly used software framework targeting robotics as well as automotive vehicle research. The open-source project aims to provide robotic researchers with a fundamental base framework to speed up the development process of robotic systems. A large number of commonly used robotic software modules and interfaces to sensors are provided and maintained by the project's active community.

The ROS framework allows the seamless integration of modular, distributed ROS-based components into a common system. This thesis's environmental perception platform is intended to be mounted on research vehicles/robots using the ROS. Thus, the ROS is also utilized as the base framework of this work's perception platform in order to enable interactions and data exchange between these two systems. This thesis' software implementation makes use of the framework's tenth distribution release: ROS Kinetic Kame[1]. A comprehensive overview of the ROS, its functionality, and similar frameworks can be found in [103]. This section provides a short overview of the main principles of the ROS.

Figure 5.2 shows a simplified illustration of the major ROS components. Custom user code is executed in so-called *nodes*. The utilized ROS version supports python and C++ for the implementation of a node's functionality. To exchange data between the nodes, the ROS makes use of the publisher-subscriber principle. A node's output data stream is made available via *topics*. The providing node *publishes* the topic, while the system's other nodes can *subscribe* to that topic. A topic is defined by a unique name and communicates a message of a certain type (e.g., integer value, float array). At startup, every node defines its published and subscribed topics. Each time a message of a node's subscribed topic is available, the receiving node gets triggered, and a callback method is executed.

A node can additionally provide *services*. A service is defined by a unique name and contains a request message and a response message. A service is called by other nodes via a service request containing the request message. The providing node executes the service using the request message and returns a response message to the caller. While topics are mainly designated for continuous data streams, services are intended for non-consistent events. In this thesis, the main application of services is to request the change of a node's parameters during runtime.

[1]ROS Kinetic Kame, released May 23, 2016 (`https://www.ros.org`).

The ROS provides a global database, the *parameter server*, accessible by all nodes of the system. The nodes can read and write parameters to and from the parameter server during runtime. For the approach presented in this thesis, the parameter server is utilized to store each node's startup parameters. The registration of ROS components (e.g., nodes, services, parameters) and their interactions are administrated by the *roscore* entity. The roscore handles the *Transmission Control Protocol* (TCP) connections between the single nodes and keeps track of the system's state. Additionally, the ROS provides tools to record and playback messages while preserving the messages' timing information. *Roslaunch* is a tool, which allows the custom startup of multiple nodes and the transfer of specified parameter values onto the parameter server. The ROS also includes *RViz*, a visualization tool capable of displaying various types of data streams at different abstraction levels (e.g., point clouds or grid maps).

Time-of-Flight Framework

The ToF cameras come with the royale software suite, a framework containing libraries to interact with the cameras and perform fundamental processing tasks. The software suite provides an API for several programming languages, including C++ and python. The latest versions of the framework also include sample code for receiving and visualizing data via a ROS node. The royale API can be configured to provide the received measurement data at different representations, including as raw phase-images, as distance/amplitude image, or as point cloud. In addition, the API enables the static and dynamic configuration of different camera parameters (e.g., exposure time, measurement mode). Multiple ToF cameras can be simultaneously connected and managed by utilizing the cameras' unique serial numbers.

For this thesis, the point cloud representation is utilized, including a 3D position (x/y/z), an intensity value, and a confidence value for each pixel. A callback can be registered and is invoked whenever a new measurement has been received and the corresponding point cloud is available. The software library can be configured to perform additional pre-processing steps before providing the point cloud (e.g., filtering, noise reduction). The amplitude and distance image of a ToF measurement can be re-obtained from the point cloud. Thus, these images are not requested via the API in order to reduce the data size of the provided sensor data.

Radar Framework

The RadarLog development kit includes a software framework to configure the radar sensor and interact with the kit during runtime. The framework includes the *RadServe*, a interposed tool in charge of handling the development kit's high-speed USB 3 connection. Additionally, software libraries are included to communicate with the RadServe, to access the raw radar data, and to send commands to the radar sensor. As seen in Figure 5.3, the RadServe acts as intermediate software module, which efficiently handles the high-speed USB 3 interface. In addition, the RadServe provides a TCP interface to custom applications running on local or remote processing systems. An application (e.g., a ROS node) can use the provided software libraries (python and Matlab) to communicate with the RadServe to interact with the radar sensor.

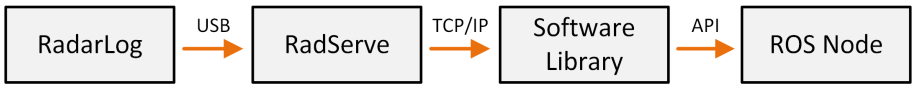


Figure 5.3: Interactions between the radar interface's different components.

Camera Framework

Basler machine vision cameras come with the *pylon Camera Software Suite*. The software suite includes a *software development kit* (SDK), providing an API interface to C/C++ applications in order to configure the camera, acquire images, and receive the image data. Basler also provides a sample ROS node, which is able to provide the majority of the API functions to the ROS system. The node can adjust the vision camera's settings (e.g., exposure time, gain), control the camera (e.g., trigger the image acquisition), and receive the provided stream of camera images. For the work presented in this thesis, the provided sample ROS node was extended in order to implement the required functionality for the desired perception tasks.

Matlab

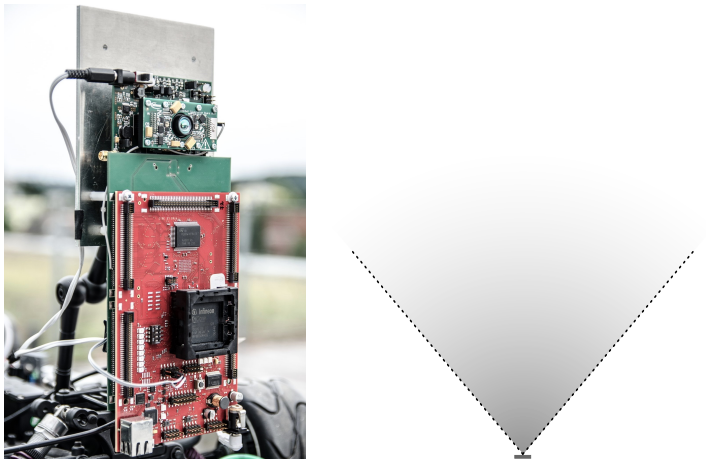
The numeric computing application Matlab was used to develop algorithms since it provides a range of capabilities for rapid data analysis and manipulation. Matlab already implements built-in functionalities to inspect and visualize data from various perception sensors at different representation levels. The software also includes convenient debugging tools to analyze the behavior of algorithms during the execution. Thus, the raw measurement data from the perception sensors was saved to a file (in the ROS receive node), which was then loaded into Matlab to visualize the data and to develop processing algorithms. Since Matlab scripts are similar to python code, the developed algorithms could be easily ported to ROS nodes while keeping the implementation overhead low.

Jupyter Notebooks

Jupyter Notebooks are open-source web applicationsn, which allow the creation of structured documents containing live python code. The live-code functionality allows fast evaluation of python code snippets and can be used to visualize data (e.g., plots, images). The tool was used to develop python code for ROS nodes since debugging and algorithm development within the ROS ecosystem introduces a significant overhead. ROS nodes were used to save measurement and configuration data into files, which were then loaded into the Jupyter Notebooks in order to perform the desired processing (e.g., algorithm development).

5.2 Environmental Perception Platforms

During the work presented in this thesis, the utilized environmental perception platform evolved from a single-sensor solution to a highly capable multi-sensor platform. This section presents two intermediate versions and the final version of the built environmental perception platform.



(a) Platform. **(b)** Field-of-view.

Figure 5.4: First version of the environmental perception platform.

5.2.1 Version I: Time-of-Flight Only

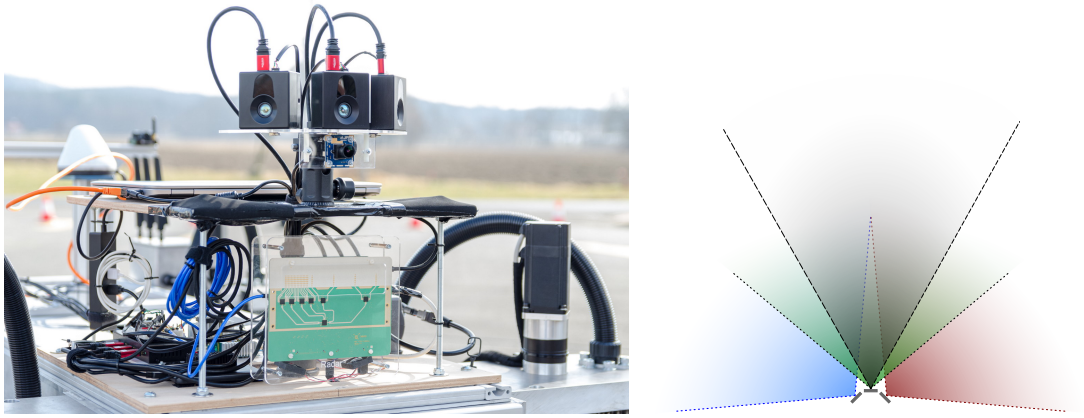
The first version of the environmental perception platform consists of a single ToF camera and an automotive-qualified microcontroller as the processing unit. The utilized ToF sensor evaluation kit and the microcontroller (Infineon Aurix) evaluation board were made available by Infineon. The platform provides a CAN and an Ethernet interface, enabling communication with external systems (e.g., a vehicle controller). Since the automotive microcontroller's image processing capabilities are limited, only lightweight image processing tasks are feasible to be performed on the safety-focused microcontroller. Advanced processing can be performed externally on additional processing units connected via Ethernet.

A picture of the platform and its corresponding field-of-view is shown in Figure 5.4. The platform is built on a solid aluminum base plate with a tripod thread, allowing easy mounting on solid objects (e.g., mobile vehicles or robots). The perception platform can be powered via a 7.2 V battery pack or via a line power converter. Since the platform solely deploys one ToF camera as perception sensor, the platform's main purpose is this sensor's evaluation for perception tasks.

5.2.2 Version II: Time-of-Flight/Radar

The second version utilizes three ToF cameras, a radar sensor, and a fisheye vision camera as perception sensors. The market-available ToF cameras are facing in different directions in order to provide an extended field-of-view. We published a detailed overview of this platform's hardware and software architecture in [87] (see also Chapter 8, Publication 4). The development of the sensor modules was not part of this thesis. The modules were made available by Infineon or obtained as market-available modules.

In addition to the perception sensors, the platform is equipped with context sensors in order to perceive various environmental conditions (e.g., the light intensity). The platform's data processing is performed on a notebook with an external hard disk to log the measurement data. Additionally, a microcontroller is used to trigger the perception sensors simultaneously.



(a) Platform. **(b)** Field-of-view.

Figure 5.5: Second version of the environmental perception platform.

Platform Composition

A picture of the platform is presented in Figure 5.5a. The perception sensors are mounted in front-facing direction. While one ToF camera is pointing straight ahead, the other two are rotated by 45° in order to extend the covered field-of-view. The radar sensor provides a narrower field-of-view but can achieve a higher range. The overlapping field-of-view of the three ToF cameras and the radar sensor is illustrated in Figure 5.5b. The fisheye vision camera provides a field-of-view of almost 170° and is used to obtain human-interpretable reference measurements of the scene.

A standard notebook is utilized as the processing platform, running the ROS and Ubuntu 16.04. Depending on the target application, all processing modules can be executed on a single computer but also distributed upon multiple computation units. An external 1 TB hard disk drive is connected to the notebook via USB to store data recordings.

The platform can be powered from a lithium-ion battery (mobile use) or the 230 V line via a power converter (stationary usage). Figure 5.6 shows the power distribution down to the single modules of the system. The battery outputs a voltage between 21.0 V and 29.4 V, provides a capacity of 270 Wh, and can power the platform for up to four hours. The varying input voltage is converted to a stable level using two DC/DC converters. Both converters output 19.5 V and can, in combination, handle the maximum current consumption of the system. One voltage converter is used to supply the notebook, while the other supplies the radar sensor (12 V - 36 V) and the USB hub (7 V - 24 V).

5.2.3 Version III: Time-of-Flight/Radar/Camera

The third and final version of the environmental perception platform utilizes two ToF cameras, one radar sensor, and one vision camera. Since the vision camera allows external triggering, the platform can acquire synchronized measurements of all utilized perception sensors. As seen in Figure 5.7a, the platform is constructed using aluminum profiles. These profiles are robust and allow fast and easy mounting of the platform on various surfaces.

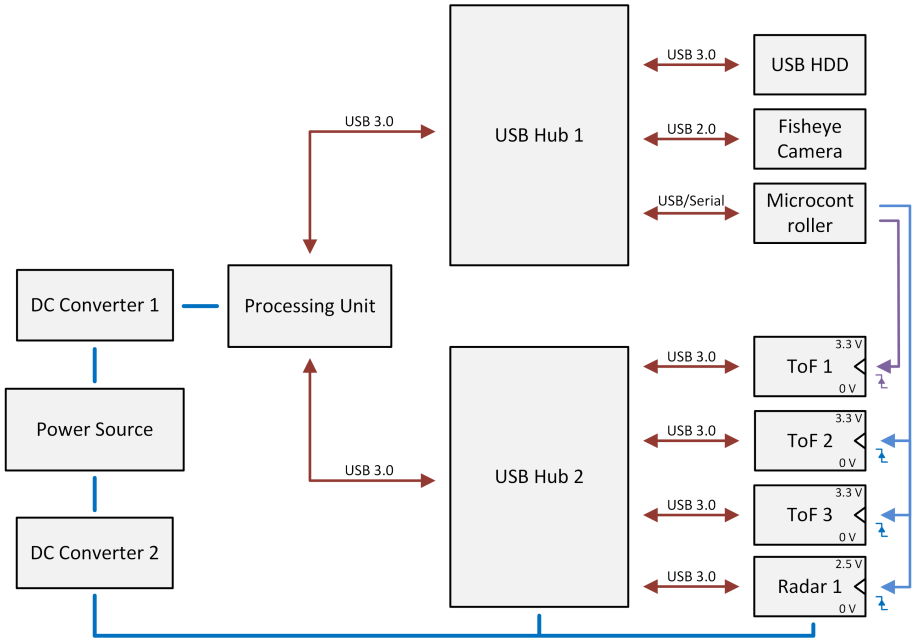
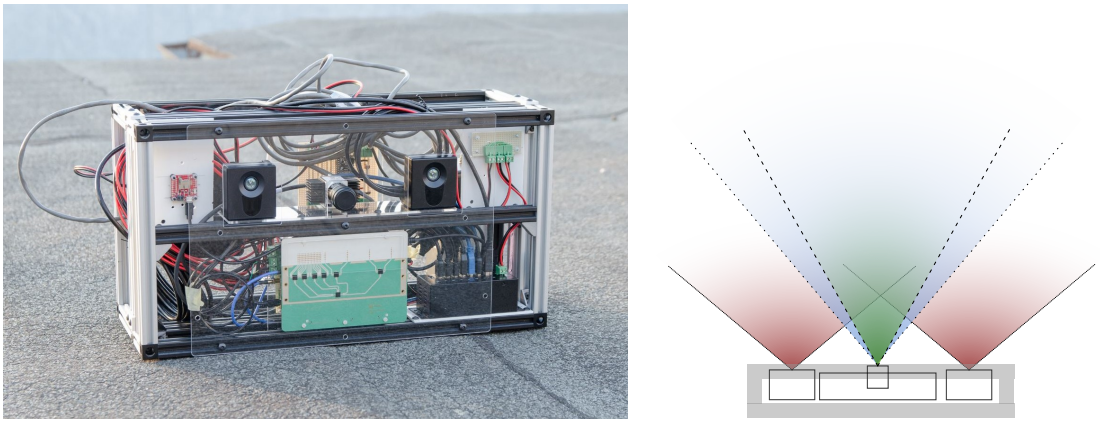


Figure 5.6: Hardware architecture of the perception platform, version II.



(a) Platform. **(b)** Field-of-view.

Figure 5.7: Final version of the environmental perception platform.

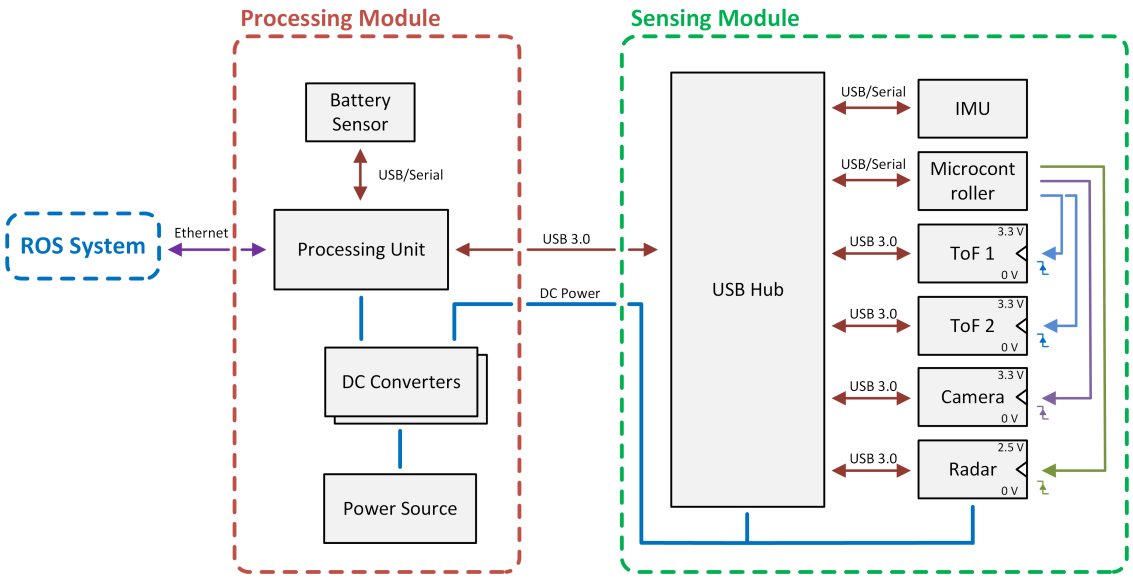


Figure 5.8: Hardware architecture of the perception platform, version III.

Platform Composition

An overview of the platform's major hardware components and their interconnections is illustrated in Figure 5.8. In order to allow flexible mounting abilities, the platform can be split up into two distinct parts: the *sensing* part and the *processing* part. As indicated in the figure, a USB and a power cable can be used to connect the two parts. The deployment of an active USB 3 cable allows the separation of the two individual two parts for distances of up to several meters (maximum tested: 8 m).

The sensing part consists of two ToF cameras, one radar sensor, and one vision camera. All perception sensors are mounted in front-facing direction and cover a common area in front of the platform (see Figure 5.7b). The platform can also be equipped with multiple context sensors, including an IMU and a light/temperature sensor. A microcontroller is used to externally trigger the perception sensors in order to acquire the measurements simultaneously. The development of the sensor modules was not part of this thesis. They were either obtained from the open market or made available as development kits by Infineon Technologies.

The final version's processing part contains an Intel NUC as the perception system's processing unit[2], running ROS Kinetic Kame on Ubuntu 16.04. The processing unit is in charge of (pre-) processing the sensor data (base perception system) and can additionally run one or multiple perception tasks (use cases). The processing part supplies the platform's components, utilizing multiple voltage converters. Similar to the second platform version, the power supply can either be provided via a battery (lithium-ion battery, integrated into the platform) or via a 230 V line converter. In order to monitor the platform's remaining battery capacity, the battery level is continuously communicated to the processing unit.

[2]Intel NUC8i7BEK, 16 GB RAM, 1 TB solid-state drive.

5.3 Base Perception System

This section provides an overview of the base perception system's ROS-based implementation. Particular focus is set on the assignment of the initial parameters, the interaction between nodes, and the the system's startup. Additionally, the platform's different operating modes are presented, allowing live and offline operation.

5.3.1 Spatial Alignment

The ROS utilizes a tree structure to store and organize the transformations between the system's utilized coordinate frames. The static transformations are obtained during an earlier one-time calibration procedure and are added to the tree at startup. The dynamic transformations are constantly re-determined and updated during operation. This section presents the ROS-based approach to manage multiple transformations during operation and to perform the calibration procedures.

Transformation Tree

The ROS keeps track of the environmental perception system's transformations by organizing them in a hierarchical transformation tree: the *TF tree*. Figure 5.9 illustrates the hierarchy of the final platform's TF tree stored in the ROS. The tree's node names are the unique identifiers of the corresponding coordinate frames. The individual nodes are connected via the corresponding transformations. The static transformations are illustrated as blue arrows, while the dynamic transformation is indicated with a red arrow. The transformation tree can be extended or updated by broadcasting new transformation messages within the ROS framework. A transformation message contains the transformation matrix between a frame and its child frame, the associated frame identifiers, and a timestamp. The TF tree enables nodes to query the time-accurate transformations between arbitrary frames during runtime.

The base perception system's implementation utilizes individual ROS nodes for each of the TF tree's transformations. Figure 5.9 shows this ROS nodes and their associated transformation in the TF tree. These nodes are contained in the base perception system's spatial alignment module. One ROS node exists for each static connection of the transformation tree, which is in charge of broadcasting the transformation message. The values of the static transformations (i.e., rotation, translation) are obtained during an earlier calibration procedure and added to the global configuration file. The respective ROS nodes fetch these parameters at startup and broadcast the corresponding TF messages. These static messages are specially labeled and are only added to the TF tree once since the respective transformation remains valid over time.

There also exists one separate ROS node to publish the dynamic connection of the transformation tree (`world_frame - ifx_platform_frame`). The node utilizes the openly available `robot_localization` ROS package [101], which provides methods for nonlinear state estimation. This node constantly estimates the transformation between these two frames, utilizing the platform's sensors and an EKF. The node repeatedly broadcasts a transformation message, defining the relationship between these two frames. In contrast to the static transformation messages, the included timestamp of the dynamic message allows time-accurate transformations.

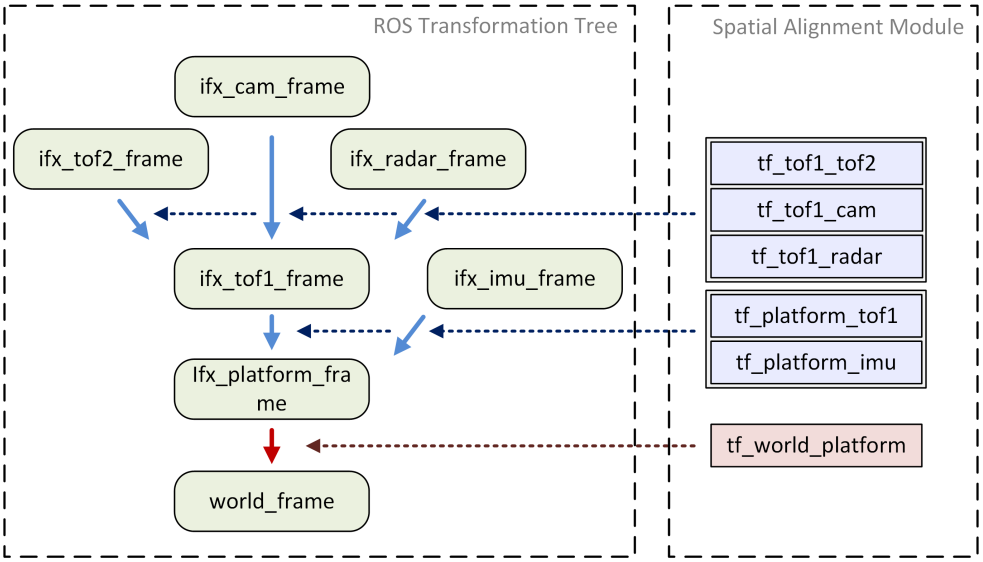


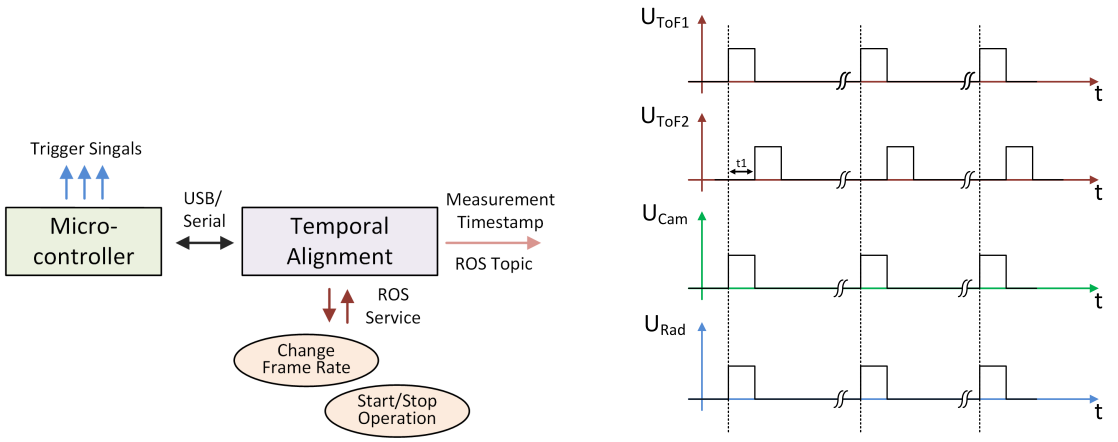
Figure 5.9: Transformation tree, organized by the ROS. The ROS nodes are in charge of updating the respective transformations, stored within the transformation tree.

Calibration

The intrinsic calibration of the ToF cameras and the vision camera is performed via standalone python scripts, utilizing saved and exported measurement data from the base perception system. For this purpose, multiple images of an 8×6 checkerboard pattern with 40 mm squares are captured with the ToF camera and the vision camera. The checkerboard's position, angle, and tilting are varied between the sequential acquisitions. OpenCV is utilized to find the pixel coordinates of the given pattern's checkerboard corners at the different positions. Next, OpenCV's camera calibration function is applied to this data to obtain the corresponding camera/lens characteristics. The function provides the camera matrix and the distortion coefficients, which can be used to correct the distorted camera image.

Since the radar sensor's intrinsic calibration requires specialized hardware, the calibration is performed by the manufacturer and provided with the sensor. The calibration data contains corrections of each receive antenna and information about the antenna's irradiation characteristics. This correction can be applied to the raw measurement data upon its reception in the radar sensor's receive module.

The perception sensors' extrinsic calibration procedures are implemented as ROS nodes, one node for each transformation. The desired calibration nodes can be manually started in addition to the base perception system. The nodes determine the corresponding transformations by implementing the methods described in Chapter 4. Since some of the nodes rely on the user's actions and feedback, the implementation supports interactive user inputs. In order to enable the direct visual evaluation of the determined transformations, the calibration nodes publish the proposed transformation messages to the base perception system. These messages are then received by the spatial alignment module and used to update the corresponding entries of the base perception system's transformation tree.



(a) Temporal alignment node. **(b)** Trigger signals.

Figure 5.10: Simultaneous measurement acquisition via external trigger signals.

5.3.2 Temporal Alignment

The sensor data's temporal alignment is controlled by simultaneously acquiring sensor measurements and estimating the corresponding timestamps. The timestamps are assigned to the data streams in order to enable temporally aligned processing at data-stream level (e.g., the synchronization of corresponding measurement streams). This section provides implementation details of these concepts within the ROS-based perception system.

External Triggering

The perception sensors are externally triggered in order to allow the simultaneous acquisition of the sensors' measurements. A microcontroller generates the sensor-dependent trigger signals for each of the perception sensors. As illustrated in Figure 5.10a, the temporal alignment node handles the communication with the triggering microcontroller. The ROS node can individually configure each perception sensor's trigger characteristics. In addition, the node implements services to reconfigure the trigger settings during runtime (e.g., the frame rate). If a simultaneous data acquisition is desired, the individual frame rates have to be set to integer multiples of a common base rate. Figure 5.10b shows an example of the microcontroller's trigger signals for each of the perception sensors. The trigger signals are adjusted accordingly to the sensors' different trigger signal interfaces.

Multiple ToF cameras are used and all of them actively illuminate the scene with infrared light. Thus, an advanced trigger scheme is required to avoid any interference between the individual ToF cameras. The trigger of the second ToF camera is delayed in order to prevent simultaneous illuminations (see Figure 5.10b). In this nested illumination mode, each additional ToF camera delays its measurement by the duration of a single phase-image's illumination duration. In this case, the ToF camera's illuminations take place while the other camera performs the sensor readouts. An entire ToF measurement acquisition takes about 38 ms for a single phase-image's illumination time of 1.5 ms. The introduced delay between the individual ToF measurements (1.5 ms) is small enough to

be neglected for the proposed system. Thus, the radar, the vision, and the two ToF measurements can be approximately assigned to a common acquisition time.

The common acquisition time of the individual measurements is considered as the reference for the temporal alignment. Since the different sensors' measurement durations depend on the individual configurations, the measurements' end times are generally not aligned. Depending on the use case, dissimilar measurement durations can have a considerable impact on the output data (e.g., a long ToF exposure time vs. a short exposure time of the vision camera). For this work's use cases, the effect of different measurement durations is neglected since the sensors' measurement durations are restricted to typical intervals.

Timestamp and Sequence Number

After a measurement has been acquired, the perception sensors transfer the measurement data to the processing unit via USB. The base perception system's receive nodes obtain the transferred data via individual API calls. In order to keep the data streams synchronized, the receiving ROS nodes add increasing sequence numbers to each measurement upon reception. As seen in Figure 5.10a, the temporal alignment node publishes a ROS topic containing an estimated measurement timestamp and the corresponding sequence number. Using this information, the receive nodes can assign the estimated measurement timestamp to their data streams.

The microcontroller sends an empty notification message to the trigger node right after a new measurement is triggered. This message's introduced transmission delay is estimated using the mean transmission time of empty messages via the serial connection between these two modules. The node estimates the acquisition time of the measurement and assigns it to an accumulating sequence number.

A disadvantage of this approach is the vulnerability to temporary data loss from a single sensor (i.e., frame drops). Thus, the receive nodes are capable of detecting frame drops, which may occur if the sensor experiences connection problems or is operated outside its limits. The nodes utilize the reception times (software timestamps) of consecutive measurements and the configured frame rate to detect missing frames and adapt the sequence number accordingly.

5.3.3 Sensor Data Processing

The base perception system's main task is to implement the low-level interface to the perception sensors and to provide structured data streams to upcoming subsystems. Since various upcoming subsystems use the provided data streams, the base perception system's processing modules focus on fundamental and generic processing steps. The output data streams consist of multiple representations of the sensors' measurement data after certain processing steps (e.g., compression, correction, abstraction). System parameters can be utilized to adjust the pre-processing configuration and the provided data streams to the desired use case(s) (e.g., disable the calculation of unused output streams).

All processing steps of the base perception system are implemented in ROS nodes. The nodes are split into receive and (pre-) processing nodes for each perception sensor. Each receiving node publishes a measurement data stream containing the low-level measurement

data after a minimum amount of processing (e.g., error compensation). In addition, each node publishes one or multiple metadata streams, including supportive information (e.g., node parameters, sensor settings). The pre-processing nodes subscribe to these streams, perform further processing, and publish the processed data as additional measurement data streams (e.g., subsampled data). Nodes of other subsystems (e.g., use cases) can subscribe to any stream published by the base perception system's nodes. The data streams are exchanged as ROS messages, containing a header with a frame identifier and a timestamp in order to allow later synchronization of multiple data streams. While the measurement data streams contain different representations of the sensor measurement, the metadata streams contain supportive information (e.g., current sensor parameters, calibration parameters).

Depending on the available sensor API, the nodes are implemented using C++ or python. Since the C++ implementation is generally more efficient, C++ was preferred for time-critical modules. Non time-critical modules were implemented using python due to its rapid-prototyping capabilities. The nodes of the base perception system's vision and ToF data flow are implemented as *ROS nodelets*. Nodelets allow multiple nodes to be executed in a single process and introduce zero copy overhead for the transfer of data between these commonly managed nodes (via shared pointers). Figure 5.11 shows an overview of the base perception system's ROS nodes and their input and output topics.

5.3.4 System Parameters

The designed ROS architecture allows the customization of the base perception system's node parameters. As indicated in Figure 5.11, there are two ways of controlling the parameters of the subsystem's nodes. The startup configuration for all nodes is provided via the ROS parameter server, while the dynamic adaption of parameters is supported via ROS services.

Startup Configuration

In the proposed architecture, the base perception system's nodes fetch their initial parameter values from the ROS parameter server at startup. The provided values include algorithm parameters (e.g., thresholds, gains), sensor configurations (e.g., exposure time, sample rate), data flow parameters (e.g., name of input/output topics and services), and debug parameters (e.g., enabling of debug output). These parameters are uploaded to the ROS parameter server from a single configuration file containing entries for all parameters within the base perception subsystem. In order to ensure a well-defined system behavior, all customizable parameters of the base perception system have to be defined in the configuration file. If any parameter is missing on the ROS parameter server, the corresponding node terminates the whole system's startup process with an error message.

Table 5.1 shows a selection of default startup parameters used to configure the radar sensor's receive and pre-processing nodes. The presented parameters configure the radar sensor to transmit fast chirp sequences with 128 chirps per measurement, 1024 samples per chirp, and a bandwidth of 2 GHz. These parameters result in a maximum range resolution of 7.5 cm, a velocity resolution of about 0.17 m/s, and a maximum detectable velocity of 4.8 m/s. Additionally, the frame delay can be customized in order to control the rate of the

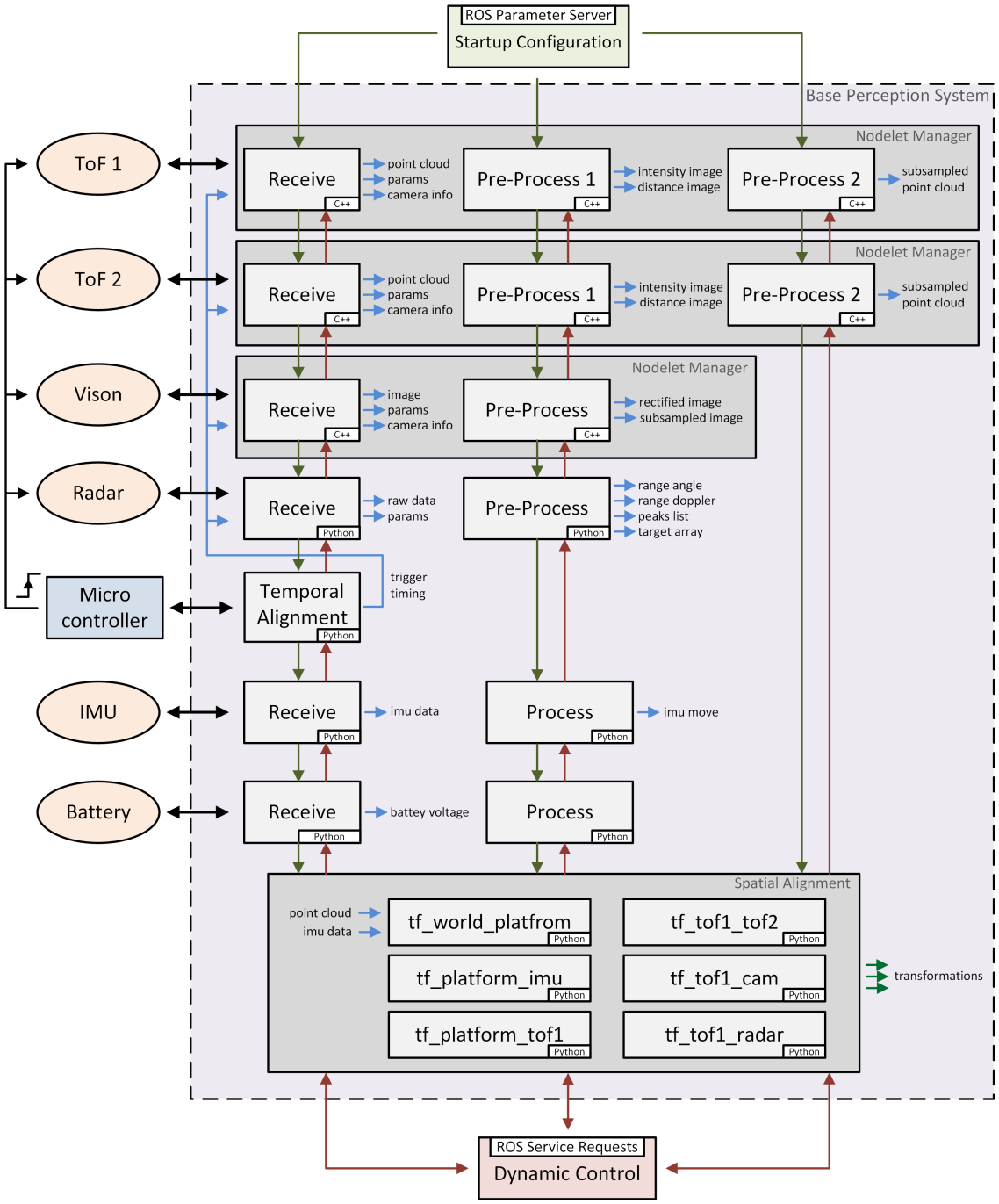


Figure 5.11: Simplified ROS architecture of the base perception system.

Table 5.1: Selected system parameters of the base perception system's radar nodes.

Radar Receive Node

Parameter	Startup Value
Start frequency	76 GHz
Stop frequency	78 GHz
Up-ramp duration	128 μs
Down-ramp duration	64 μs
Pulse time	200 μs
Measurement time	400 ms
Samples per chirp	1024
Number of chirps	128
Receive channels	16
Transmit channels	1
Frame delay	1000 ms

Radar Pre-Processing Node

Parameter	Startup Value
FFT bins stage 1	1024
FFT bins stage 2	256
FFT bins stage 3	256
CFAR reference length	40
CFAR guard length	4
CFAR offset	45
Minimum distance	0.25 m
Maximum distance	15.0 m

triggered measurements. The parameters of the pre-processing node control the behavior of the (pre-) processing algorithms (e.g., FFT calculation, CFAR detection). Since these algorithms have a high impact on the perception performance, the associated parameters have to be selected in compliance with the targeted application(s).

Dynamic Configuration

The base perception system's nodes allow the dynamic adaption of their parameters during runtime. In order to provide this functionality, the subsystem's nodes implement ROS services, able to be called during runtime. A call to these services includes a request message containing the proposed parameters for the corresponding node. In case of an incoming service request, the respective node gets triggered, examines the requested values, and attempts to update its parameters. After processing the service request, the node answers with a response, indicating whether the request was successful or not. Since various system parameters are not feasible for a dynamic adaption (e.g., name of input/output topics), only a subset of the system's parameters are made available for dynamic adaptation.

Some of the parameters can be changed on-the-fly, while others introduce an adaption delay. The re-configuration of certain sensor parameters requires a restart of the sensor (e.g., radar bandwidth or radar receive channels). During that adaption process, the corresponding sensor cannot provide measurement data, causing an interruption of the data stream. Parameters, which do not have to reconfigure the sensor, are immediately active without the introduction of an additional delay.

The dynamic parameter adaption is implemented with ROS services, since the utilization of the ROS parameter server introduces considerable drawbacks. In contrast to ROS services, the parameter server does not support direct feedback to the calling instance and does not include any timing information. With ROS services, a calling node can utilize the service's response message to gain information about the adapted parameters' validity period.

5.4 Use Cases

This section presents implementation-specific details of the platform's implemented use cases. Use cases subscribe to a subset of the base perception system's data streams in order to perform application-specific tasks. The use cases are decoupled from the base perception subsystem, enabling a distributed execution on multiple processing units.

5.4.1 Context-Aware Parameter Adaption

This use case subscribes to data streams of additional context sensors (IMU, battery sensor) and to the measurement data streams of the perception sensors (ToF, radar, vision). Multiple parameter handlers (ROS nodes) are deployed in order to dynamically adapt the parameters of the base perception system's nodes. Each parameter handler is realized as a single ROS node and is in charge of adapting one or more parameters. These parameter handlers utilize their input data streams to decide whether a certain parameter of a node shall be updated or not. If a parameter shall be updated, the parameter handler calls the respective node's ROS service and requests the adaptation of the considered parameter. Depending on the type of the requested parameter, the node will perform one of the following three actions to update it.

- Update a variable
  A processing-specific value (e.g., threshold, flag) is updated. The new value will be valid, beginning with the processing of the upcoming data stream.

- Update a sensor configuration value
  In order to change the sensor configuration (e.g., trigger mode, pixel binning, sample rate), the node has to reconfigure the sensor. In general, the adjustment is initiated by requesting a parameter update via a call to the respective sensor's API. Depending on the adapted parameter and sensor type, the update of the sensor's configuration may be valid immediately or after a temporary downtime of the sensor.

- Update of an external parameter
  To change an external parameter, a service call to another node is performed. An example is the adaption of the individual sensors' frame intervals, which are handled by the receive nodes but have to be adjusted in the temporal alignment node.

After a parameter adaption, the parameter handlers utilize the timestamp of the corresponding node's service response in order to estimate the timestamp of the resulting parameter update. The parameter handler can then use this timestamp to assess the impact of the updated parameter for iterative adaptation approaches.

5.4.2 Obstacle Detection

The obstacle detection use case subscribes to ToF and radar measurement data streams in order to generate a combined occupancy grid from the heterogeneous range data. Table 5.2 shows the input data streams of the obstacle detection use case. The subsystem synchronizes the different measurement data streams, utilizing the assigned timestamps. The use case incorporates a ROS node which merges the two subsampled ToF point clouds into a

Table 5.2: Obstacle detection use case: input data streams.

ROS Topic	Description
`/tof1/pc_voxel`	Subsampled point cloud of the first ToF camera
`/tof2/pc_voxel`	Subsampled point cloud of the second ToF camera
`/radar/peaks`	Peak list of the radar sensor
`/radar/ra`	Range-angle image of the radar sensor

common point cloud. Individual processing nodes are then used to convert the individual streams into separate grid structures, which are then fused into a common occupancy grid. The resulting occupancy grid is a 2D top-down grid of the area in front of the perception platform, representing the probability of occupancy for each grid cell. An additional ROS node subscribes to the occupancy grids and considers the temporal changes to create an occupancy grid map of the perceived field-of-view. Numerous parameters of the occupancy grid map (e.g., number of cells, number of frames) can be configured via the use case's node parameters. The nodes to merge the point clouds and to create the occupancy grid from the input streams are implemented in C++, since they require advanced point cloud processing techniques (implemented using the C++ PCL library). The mapping node is implemented in python since the high-level programming language enables a fast development workflow.

5.4.3 Environment Mapping

The environment mapping module utilizes chronologically acquired ToF point clouds to create a map of the environment. The iteratively extended map is stored in an overall point cloud, containing the correctly aligned points from previous measurements. Since the map is represented in a fixed-position world frame, the transformation between the ToF camera's frame (`ifx_tof1_frame`) and the global world frame (`world_frame`) has to be known. The mapping module utilizes the latest ToF point cloud's timestamp in order to fetch the corresponding transformation from the TF tree. Using this transformation, the mapping node can transform the point cloud into the global map's frame and merge it with the existing map. In order to keep the stored map within a feasible file size, the ToF camera's point cloud is converted into a 3D voxel grid before it is inserted into the overall map.

The environment mapping module highly depends on the quality of the transformation between the ToF camera's frame and the world frame. The most sensitive part of this transformation is the dynamic transformation between the environmental perception platform's base frame and the world frame. This transformation is determined and updated by the corresponding node of the base perception system's spatial alignment module. The `tf_world_platform` node applies an EKF to multiple sensor data streams to estimate the platform's absolute pose in the global world frame. The input data to the EKF are the relative pose updates from the IMU sensor and from consecutive ToF point clouds. In order to obtain a precise transformation, the parameters of the spatial alignment module (including the EKF configuration) have to be set under consideration of the targeted application.

Table 5.3: Pedestrian detection use case: input data streams.

ROS Topic	Description
`/tof1/pc_voxel`	Subsampled point cloud of the first ToF camera
`/radar/peaks`	Peak list of the radar sensor
`/camera/image_rect`	Rectified full resolution camera image
`/camera/camera_info`	Camera information (e.g., calibration values)

Although the implemented use case only utilizes the point clouds from one ToF camera, the approach can be easily extended to include the second ToF camera's point cloud. For this purpose, the transformation between the second ToF camera's frame (`ifx_tof2_frame`) and the global world frame (`world_frame`) has to be considered. The advantage of a second camera is the faster creation of the map due to the extended field-of-view.

5.4.4 Pedestrian Detection

The pedestrian detection use case combines the vision camera's image with the radar sensor's range data and one ToF camera in order to improve the ability to detect pedestrians. The module's inputs are the rectified 2D vision camera image, the subsampled point cloud from one ToF camera, and the target list from the radar sensor (see Table 5.3). The input data streams are synchronized using the timestamps, included in the data streams' headers.

First, the range data is cropped to the area of interest in front of the platform (e.g., excluding range data from objects over 2 m or at high distances). Due to its missing vertical information, the detected radar targets are converted into vertical lines, delimited by the sensor's field-of-view and the ground surface. The lines are identified by a start point and an end point in the 3D coordinate space. Using the vision camera's calibration values, these points are projected onto the image plane (using the OpenCV library) and the corresponding lines are drawn on the image. Next, the subsampled points of the ToF point cloud are projected onto the image plane and visualized as dots on the image. The resulting image of projected range data is then converted into a binary image, containing a positive value for areas containing range data.

The vision camera's rectified image is used to detect pedestrians via a HOG-based pedestrian classifier (included in the OpenCV library). This pedestrian classifier is applied at multiple scales across the image in order to detect potential areas containing pedestrians. The classifier is applied with a low threshold in order to increase the classifier's detection sensitivity. Since each positive classification provides a bounding box, this module's output is a list of bounding boxes distributed across the image.

The bounding boxes are then amplified by the binary range data image, resulting in a list of weighted bounding boxes. This list of bounding boxes is then converted into a single heatmap image. This heatmap image is thresholded and labeled in order to obtain the resulting bounding boxes, denoting the detected pedestrians. Many parameters of the use case can be customized, including threshold values, classification parameters, and the size of the projected range data.

5.4.5 Data Visualization

This use case is in charge of visualizing various data streams of the environmental perception platform. The data streams can originate from the base perception system and from other use cases. The subsystem utilizes the ROS framework's graphical visualization tool RViz. The tool is able to directly visualize several standard message types of the ROS (e.g., images, point clouds, or occupancy grids). In addition, RViz can display generic markers to support the visualization of custom messages.

While some of the environmental perception platform's data streams can be directly visualized in RViz (e.g., ToF point cloud, vision camera image), others cannot be visualized without further processing (e.g., radar peak list, radar range-angle image). The data visualization use case contains nodes to convert non-visualizable data streams into visualizable formats. For that purpose, the nodes subscribe to non-visualizable messages, convert them to a visualizable format (e.g., using markers, point clouds), and publish them in order to enable their visualization in RViz. Examples are the detected radar targets, which are converted into markers with custom colors and sizes, indicating each target's velocity and magnitude.

The data visualization use case can be manually started in addition to the remaining system if visualization of the sensor data is required (e.g., for visual evaluation). The desired configuration of the data visualization use case highly depends on the active use case(s) of the environmental perception system. Thus, individual predefined visualization configurations (including viewing settings) are available for each use case of the environmental perception system. Depending on the active use case (e.g., pedestrian detection), the corresponding conversion nodes for non-visualizable messages are started and the viewing settings of the RViz tool are adjusted accordingly. The visualization use case's system parameters can be used to further customize the behavior of the subsystem (e.g., disable certain conversion modules).

If the visualization use case is not started, the visualization modules are inactive (including the conversion nodes) and the full processing capabilities can be focused on the data processing. This can be beneficial in computationally expensive scenarios (e.g., the processing/recording of multiple data streams at high frame rates).

5.5 Platform Startup and Operating Modes

This section presents the implemented approach to simultaneously start and configure multiple ROS nodes via a single ROS launch file. The controlled startup procedure of the base perception system is described, including its contained nodes and utilized sensors. Additionally, the three operating modes (livestream, record, playback) are introduced. Eventually, the utilized method to startup use cases and configure their nodes is presented.

5.5.1 Launch Files

A top-level ROS launch file is used to start the modules and nodes of the base perception system. The *Extensible Markup Language* (XML)-based launch file utilizes Boolean arguments to enable or disable single modules (e.g., radar, IMU). The top-level launch file includes a configuration file (also stored as launch file), which holds the initial param-

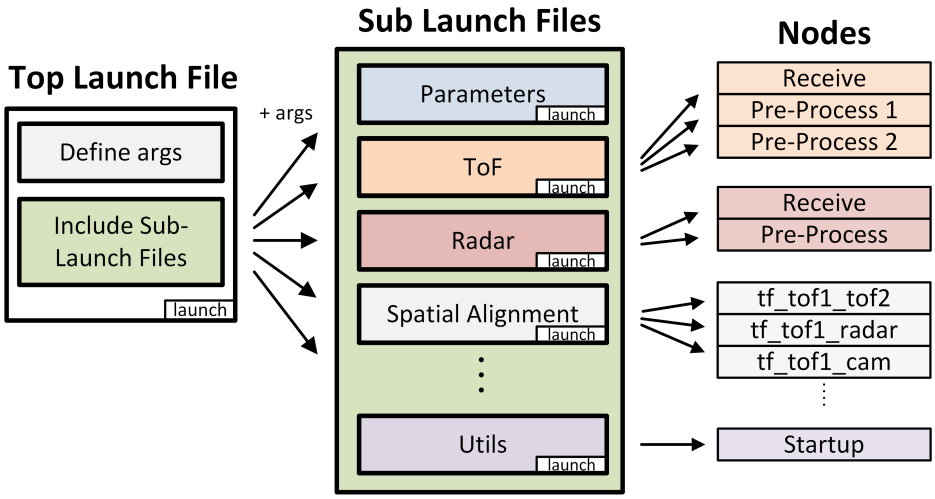


Figure 5.12: Composition of the top-level launch file.

eters of all base perception system's nodes. Upon startup, these initial parameters are transferred to the ROS parameter server and consecutively fetched by the corresponding ROS nodes. The enabled modules of the base perception system are included via sub-level launch files, which start the associated nodes according to the top-level launch file's arguments. Figure 5.12 shows a simplified illustration of the top-level launch file's composition and its linked components (e.g., sub-level launch files, nodes). As seen in the figure, the defined arguments of the top-level launch file are passed to the sub-level launch files in order to allow submodule-specific argument handling.

5.5.2 Sensor Platform Startup Procedure

The environmental perception platform utilizes a supervised startup procedure for its perception sensors. The base perception system requires all sensors to be ready for operation in order to provide its desired functionality. The parallel startup of multiple sensors can be problematic since all sensor nodes utilize the common USB interface. An unsupervised startup process can lead to non-deterministic behavior, i.e., cause some sensors to enter unresponsive states. Thus, a supervised startup routine is employed to start the sensors in a structured way.

After startup, the base perception system's receive nodes remain in an inactive state (per default). The receive nodes are in charge of the low-level communication with the physical sensors (e.g., starting the sensors and handling their data). A custom startup node is utilized to turn these nodes into running-state, assuring that the sensors are started in a controlled and deterministic way.

Figure 5.13 shows the flow diagram of the startup node. The node consecutively requests the transition of the base perception system's receive nodes into running-state. The receive nodes respond to the service request as soon as the respective sensor is started and the node is ready for operation. When all sensors are active and ready, the trigger node is turned into running-state to start the acquisition of sensor measurements.

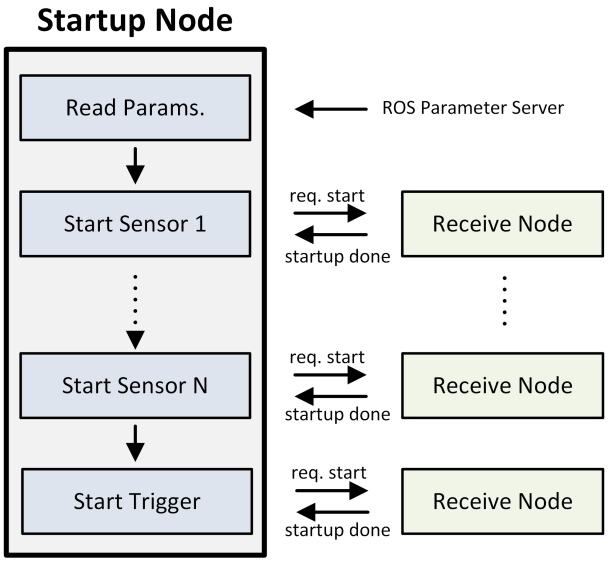


Figure 5.13: Starting the environmental perception system's sensors via a common startup node.

5.5.3 Operating Modes

The environmental perception platform supports multiple operating modes. In addition to its regular livestream operation, the environmental perception platform can be utilized to record individual datasets and playback earlier recorded datasets. Each operating mode employs a custom composition of launch files in order to start the associated components. The environmental perception platform's three operation modes of the are listed and described below.

- **Livestream mode**
  In livestream mode, the sensors acquire measurements and provide the obtained data to the base perception system. The subsystem receives the measurement data, performs various (pre-) processing steps, and provides multiple data streams at different abstraction levels to upcoming subsystems (e.g., use cases). As seen in Figure 5.14, the livestream launch file starts the base perception system's ROS nodes. The livestream launch file's arguments can be utilized to en/disable or individual modules, depending on the desired output.

- **Record mode**
  In record mode, selected data streams of the base perception system are recorded for offline evaluation. The record launch file is started in addition to the livestream launch file, as seen in Figure 5.15. The rosbag recording node subscribes to a number of data streams and efficiently stores them in a rosbag file. The subsect of the base perception system's data streams, which shall be recorded into a rosbag file, can be customized in the record launch file. The individual modules of the base perception system can be en/disable via the livestream launch file. The manual start of the recording launch file enables the custom selection of the recording's start time.

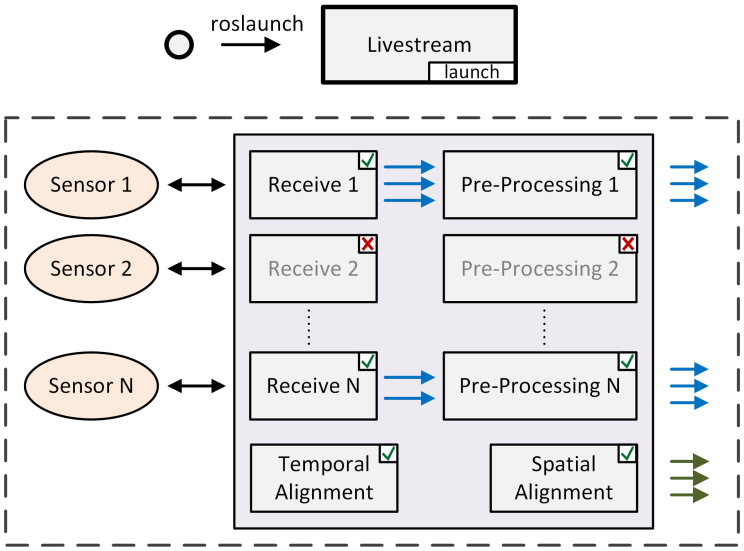


Figure 5.14: Livestream operating mode.

- **Playback mode**
  In playback mode, a recorded dataset is provided as input to the base perception system. In this operating mode, the playback launch file is started on top of the livestream launch file, as seen in Figure 5.16. The playback launch file starts the playback node, which recalls earlier recorded data streams (ROS messages) from a rosbag file. In addition, it starts the livestream launch file with disabled receive nodes, causing the physical perception sensors to stay inactive. The remaining processing modules are activated as defined in the livestream launch file.

The livestream mode is used to directly process the perceived measurement data without recording any data streams. This mode may be selected if the system's current environment shall be perceived and recording is not desired or feasible. An example is the platform's utilization to perform live perception tasks (e.g., pedestrian detection). Omitting the recording can be beneficial since it introduces an additional processing overhead, limiting the system's perception performance.

The record mode can be utilized to record dedicated datasets or to additionally store the measurement data during live processing. To record dedicated datasets, the base perception system's received measurement data streams are saved to the rosbag file. In order to record the data efficiently, all nodes of the base perception system except for the receive nodes may be deactivated (see Figure 5.15).

The playback mode is advantageous for the development of (pre-) processing modules and new use cases. Since recorded datasets are used, the performance of different implementations can be compared, based on similar input data. The playback mode enables a faster data evaluation than the livestream mode since the input data stream is directly available, and the physical sensors do not have to be started. Particular edge cases and key scenarios can be utilized as input data in order to evaluate the respective performance of different algorithms.

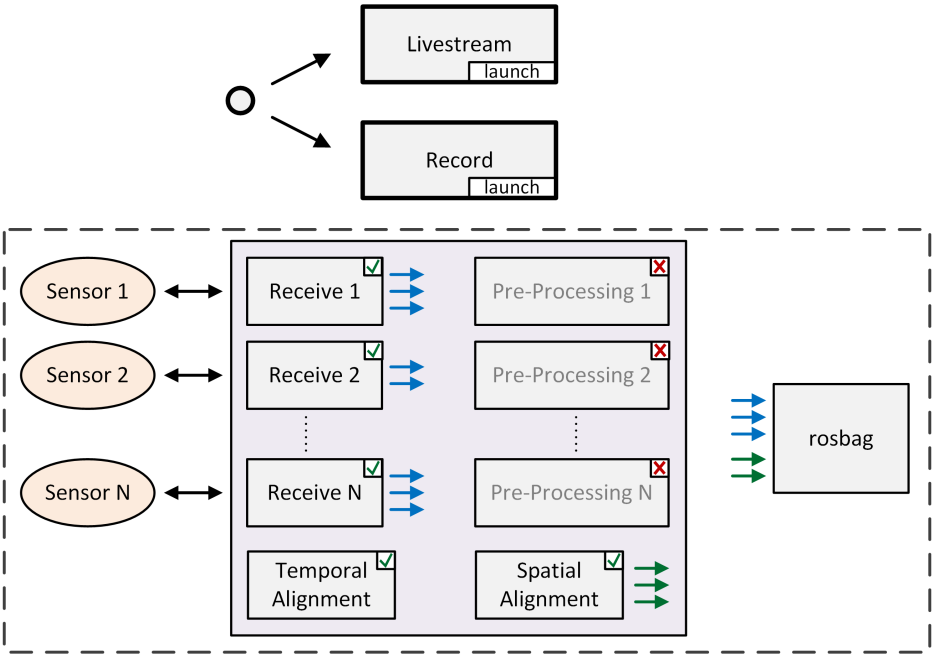


Figure 5.15: Record operating mode.

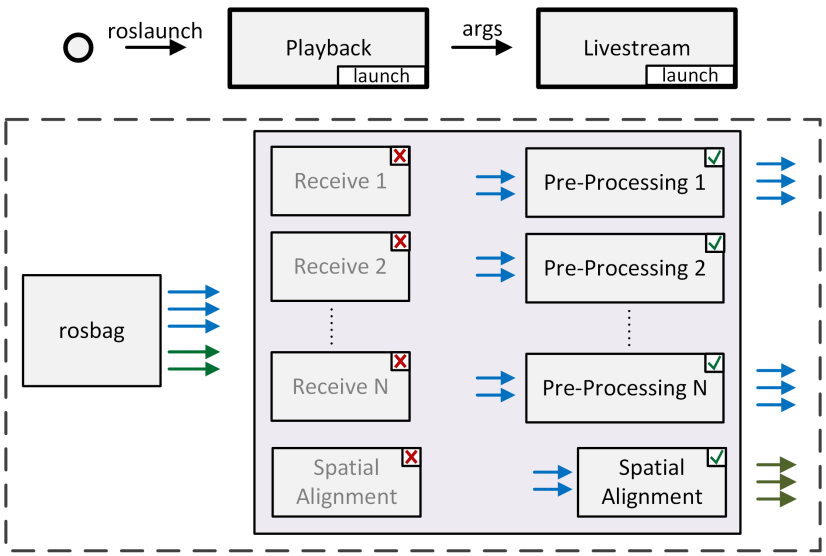


Figure 5.16: Playback operating mode.

5.5.4 Use Case Startup

Use cases can be manually started in addition to the base perception system. Each use case provides a standalone top-level launch file, defining the configuration of the corresponding subsystem. Similar to the livestream launch file, a use case's launch file defines the subsystem's parameters and starts the contained ROS nodes.

The launch file of an individual use case is typically compromised of the three main sections listed below.

- Launch file arguments
  The launch file arguments are defined in the initial section of the file and are used to enable or disable certain modules of the launch file. Examples are the activation of the obstacle detection use case's mapping module or the selection of the active parameter handlers of the context-aware parameter adaption use case.

- Node parameters
  The node parameters define the parameter values of the use case's contained ROS nodes. These parameters include the individual nodes' configurations (e.g., algorithm parameters, topic names, debug levels). In contrast to the base perception system, the use cases' parameters are directly defined in the corresponding top-level launch files.

- Node inclusions
  The last section of a use case's launch file includes the subsystem's nodes (e.g., specific processing nodes, fusion nodes). The individual modules (set of the nodes) can be enabled and disabled via the launch file arguments.

One or multiple use cases can be manually started in addition to the base perception system. The base perception system's presence is mandatory since the implemented use cases depend on the base perception system's output data streams. Also, the base perception system's operating mode (livestream, record, playback) does not affect the use cases' functionality and can be selected independently.

The data visualization use case provides an exception to the general structure of the use cases. Since the visualization configuration depends on the environmental perception system's active setting, multiple predefined launch files exist for the visualization use case. Individual visualization launch files exist for each use case (e.g., pedestrian detection, obstacle detection) and the base perception system. One or multiple of these launch files can be manually invoked to visualize the desired aspects of the environmental perception system.

Chapter 6

Results

This chapter provides an overview of the results obtained during the evaluation of this work's environment perception platform. First, the platform's final version is presented, and its attachment to different research robots/vehicles for real-world data acquisition is shown. The main part of the chapter presents the performance of the base perception system's modules and its capabilities in various scenarios. Additionally, example outputs of the implemented use cases are presented, and each use case's general performance is analyzed.

6.1 Environmental Perception Platform

As part of this thesis, multiple versions of an environmental perception platform were constructed. These perception platforms were attached to different vehicles in order to obtain real-world data in various scenarios. This section presents the final perception platform and shows this work's different vehicle setups to acquire real-world measurements.

6.1.1 Final Platform

The environmental perception platform's final version utilizes two ToF cameras, a vision camera, and a radar sensor as perception sensors. The platform is equipped with a battery sensor, an IMU, a processing unit, and a lithium battery, enabling independent and mobile operation. The platform can be split into two parts in order to enable flexible mounting options. Since the physical platform is based on aluminum profile beams, a simple attachment to various surfaces is possible. Figure 6.1 shows an image of the platform with the two distinct parts visible.

The platform can be employed in static and dynamic operating scenarios. In static scenarios, the platform is positioned on stationary, non-moving objects/surfaces. In that case, the perception platform's field-of-view does not change over time. The only dynamic behavior occurs if moving objects are present in the perceived scene. In dynamic operation scenarios, the system is mounted on a moving base (e.g., robot or vehicle). In this case, the platform's position, orientation, and the corresponding field-of-view change over time. The data acquisition in dynamic scenarios introduces real-world challenges, also faced by perception systems of automated vehicles and autonomous robots.

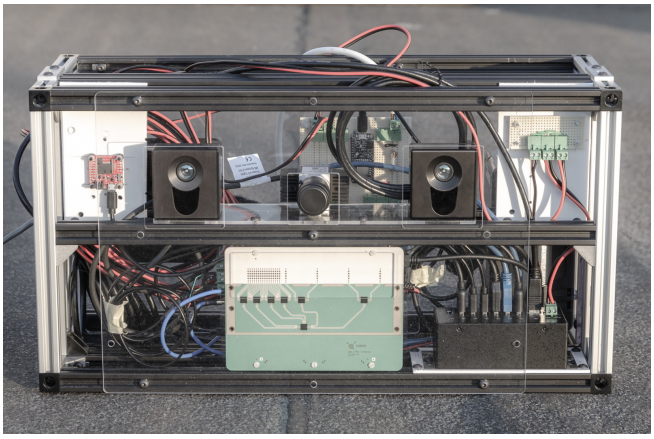


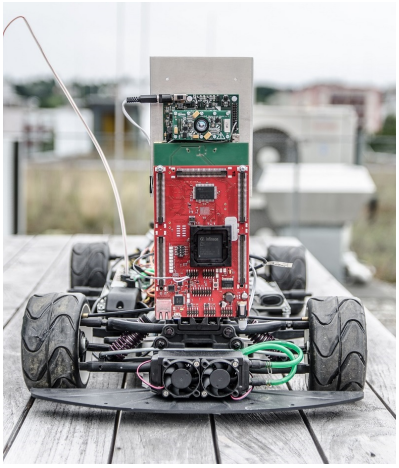
Figure 6.1: Final version of the environmental perception platform.

6.1.2 Attachment to Vehicles

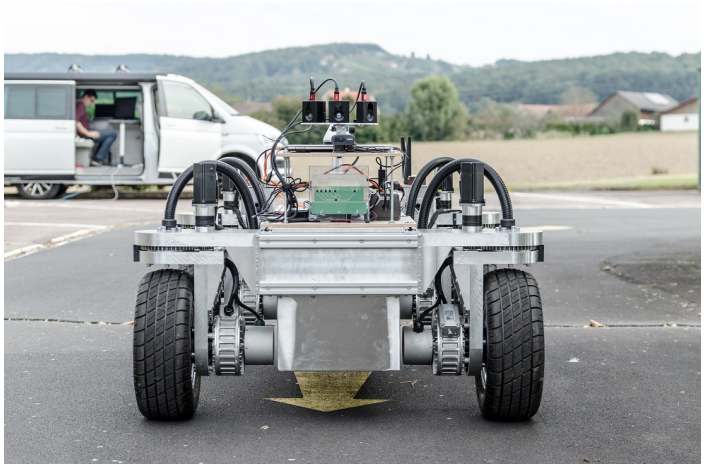
Due to a cooperation with the Virtual Vehicle Research Center in Graz, different releases of this work's environmental perception platform were mounted on research vehicles in order to obtain real-world measurement data in dynamic scenarios. The environmental perception platforms were connected to the respective vehicle's processing system, enabling the exchange of data between the subsystems (e.g., based on the ROS). The perception data was provided to the vehicle's processing system, enabling the incorporation into the vehicle's driving tasks. Additionally, the acquired real-world perception data was recorded in order to utilize it for offline evaluation of the processing pipeline in dynamic scenarios.

As the first step, the ToF-only perception platform (version I) was mounted on a scaled vehicle (see Figure 6.2a) to test the perception capabilities of ToF cameras in dynamic environments. The utilized scaled vehicle, provided by the Virtual Vehicle, is equipped with customized control electronics, allowing wireless remote control. The ToF measurements are directly processed by an automotive-qualified microcontroller (Infineon Aurix) in order to detect obstacles in the vehicle's path. The microcontroller is able to send commands to the vehicle's system controller in order to apply the vehicle's brakes or to modify its path. The work published in [106] shows a comprehensive overview of this approach and is included in this thesis (Chapter 8, Publication 1).

The updated version of the perception platform (version II) was mounted on an unmanned robot vehicle (see Figure 6.2b). The ROS-based robot was fully developed and constructed by the Virtual Vehicle as part of the ACTIVE research project. The second version of the perception platform provides data from ToF, radar, and vision sensors. A notebook is utilized as the processing unit, running a ROS-based perception software to process the sensor data and detect obstacles. The platform's output is forwarded to the vehicle's processing system to integrate the perceived surrounding data into its path-planning process. The research project's main focus was to perform accurate positioning, utilizing *vehicle-to-infrastructure* (V2I) communication. The perception platform was successfully utilized to detect obstacles and as a reference in order to evaluate the V2I localization accuracy. A more detailed description of this approach is published in [105] and is attached to this thesis (Chapter 8, Publication 7).



(a) Mount on a scaled vehicle [104].



(b) Mount on an unmanned ground vehicle [105].

(c) Mount on a passenger vehicle.

Figure 6.2: Attachment of the environmental perception platform to different vehicles.

The final version of the perception platform (version III) was mounted on a ROS-based research vehicle provided by the Virtual Vehicle Research Center (see Figure 6.2c). The modified passenger vehicle utilizes state-of-the-art automated driving components (e.g., Nvidia processing unit, drive to wire control, spinning rooftop lidar). The processing/powering part of the perception platform was placed in the vehicle's trunk. The sensing part of the perception platform was mounted at the vehicle's front bumper and was connected to the processing part via a USB 3 and a power cable. The perception platform can be either powered by the vehicle's power supply or by the perception platform's standalone battery. The environmental perception platform's provided measurement streams are forwarded to the vehicle's processing system and utilized as additional perception information. The setup was utilized to acquire real-world measurement data and evaluate the implemented perception tasks.

6.2 Base Perception System

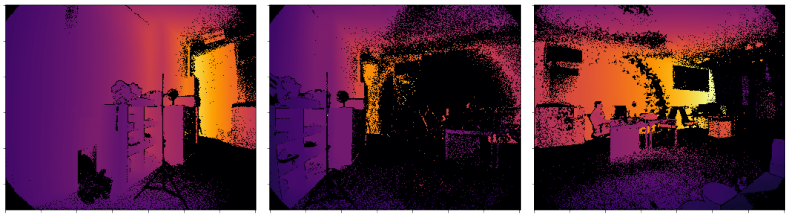
The base perception system describes the environmental perception platform's fundamental subsystem. The base perception system is in charge of the interactions with the sensors and the low-level data handling. This includes the temporal and spatial data alignment, the data (pre-) processing, and the configuration of the sensor/processing modules. The software architecture of this subsystem was designed and implemented, based on the ROS framework, enabling a data-driven processing flow using multiple independent modules. The hardware-specific structure of the base perception system's software architecture allowed only limited re-use of existing software modules, openly available to the research/robotics community. Thus, the majority of the modules were implemented from scratch, based on open-available research, or adapted from similar modules. This section presents results associated with this subsystem as well as performance metrics of the included modules.

6.2.1 Temporal Alignment

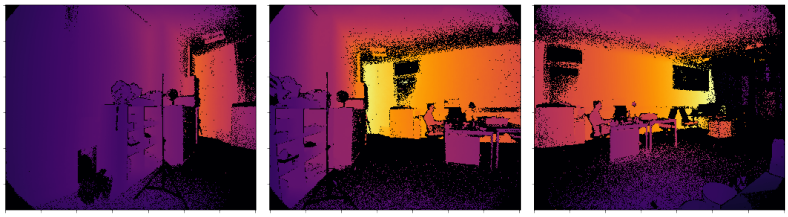
In order to obtain simultaneously acquired measurements, the different perception sensors are jointly triggered via external signals. This section presents the output of simultaneously triggered ToF cameras, showing the need for nested triggering to avoid interferences between the individual cameras. Additionally, the uncertainty of the data streams' estimated measurement timestamps is analyzed. A more detailed analysis of the perception system's temporal alignment is presented in the associated paper, published in [102] and included in this thesis (Chapter 8, Publication 9).

Nested Time-of-Flight Triggering

Simultaneous triggering of multiple sensors can lead to interferences. Figure 6.3a shows the distance images acquired from three simultaneously triggered ToF cameras (platform version II). The simultaneous illumination phases of the different cameras cause disturbances in the distance images due to interference effects. As seen in the figure, the center camera is most affected, resulting in invalid pixels (black) in the image's center.



(a) Simultaneous mode.



(b) Nested mode.

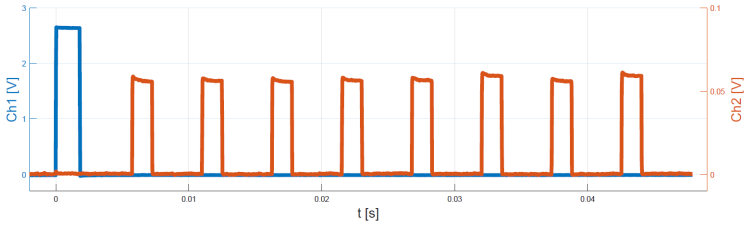
Figure 6.3: Influence of the external trigger strategy on the ToF measurement quality.

In order to avoid interferences, a nested illumination scheme is implemented. In nested illumination mode, the utilized ToF cameras are triggered with individual delays, causing small offsets between the measurement acquisitions. These individual trigger delays cause illuminations to be performed while the other cameras are in their readout phases. Figure 6.4a shows the illumination and readout phases of one measurement performed by a single ToF camera (eight-phase measurement). The time plot of three ToF cameras, performing eight-phase measurements in nested illumination mode, is depicted in Figure 6.4b. The signal course was recorded with an oscilloscope, using a photodiode in order to obtain a signal indicating the illumination phases. The plotted signals represent the trigger signal and the illumination intensity measured with the photodiode. In nested mode, the single illuminations do not influence each other and provide an interference-free output (see Figure 6.3b).

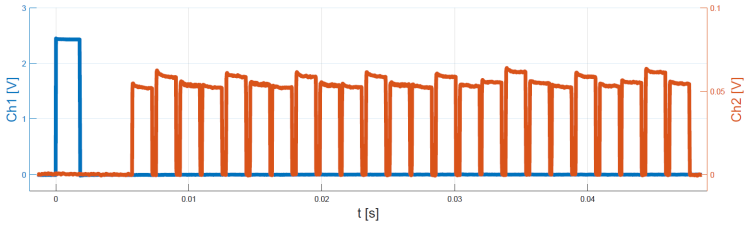
In the shown example, the offset between the individual frames is set to the duration of a single illumination (approximately 1.8 ms). Since a full readout of the ToF sensor takes about 3.8 ms, a maximum of three cameras can be operated in the nested mode without overlapping illumination phases. Consecutively triggering the ToF cameras (one full measurement after each other) would introduce a measurement offset of approximately 41 ms. In contrast, the offset introduced by nested triggering is small enough to be neglected for numerous applications, including this work's use cases.

Timestamp Accuracy

The sensors' measurement data streams are extended with an estimated measurement timestamp upon their reception by the base perception system's receive modules. The corresponding measurement timestamp is estimated utilizing a notification message from the trigger microcontroller. Since the timestamp is derived from the reception time of that message, the unknown transfer time causes an uncertainty in the resulting timestamp. The base perception system's temporal alignment module tries to compensate for that



(a) One camera.



(b) Nested mode with three cameras.

Figure 6.4: Temporal plot of a ToF camera's measurement in regular and nested mode.

delay utilizing both the estimated transfer time and the sensors' configured measurement rate. The mean round trip latency of the notification message (for 15 000 transfers) was determined during an experiment, resulting in an estimated mean transfer time of 0.825 ms. The additional incorporation of the frame rate helps to reject outliers and to keep the timestamp uncertainty in the range of a single millisecond during regular operation.

6.2.2 Spatial Alignment

The spatial alignment between the base perception system's individual coordinate frames is stored and maintained by the ROS framework's TF tree. The static transformations between the single perception sensors' frames are obtained during a separate calibration phase. The dynamic transformation between the platform and a location-fixed world frame is continuously updated during operation, utilizing the data from multiple sensors (IMU, ToF). Figure 6.5 shows the position and orientation of the platform's frames in 3D, graphically represented using the ROS framework's visualization tool RViz. The transformation tree, as stored by the ROS is depicted in Figure 6.6. As seen in this figure, each transformation includes a timestamp, allowing the system to provide temporally accurate transformations at the sensors' measurement times.

Precise transformations between the single perception sensors are crucial to enable the high-quality fusion of data from multiple sensors. The static transformations are determined during individual calibration procedures and provided to the TF tree at startup. Since a high quality of the transformations is required, the transformations obtained during the calibration procedures are manually re-evaluated after the calibration phase. In order to evaluate these transformations, multiple calibration targets are placed into the sensors' common field-of-view. The deviations between the single sensors' detected target positions are then utilized to indicate the corresponding transformation quality.

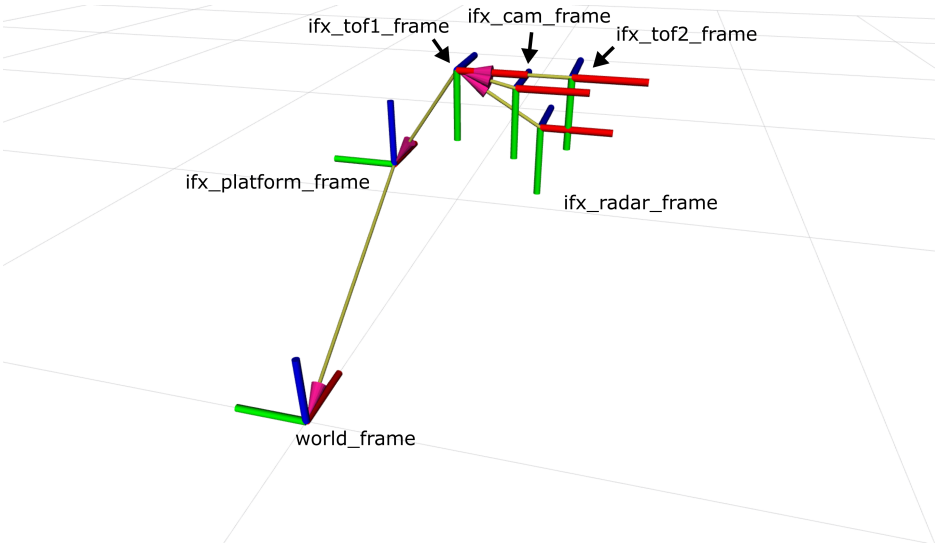


Figure 6.5: Spatial alignment of the platform's frames, visualized using the ROS tool RViz.

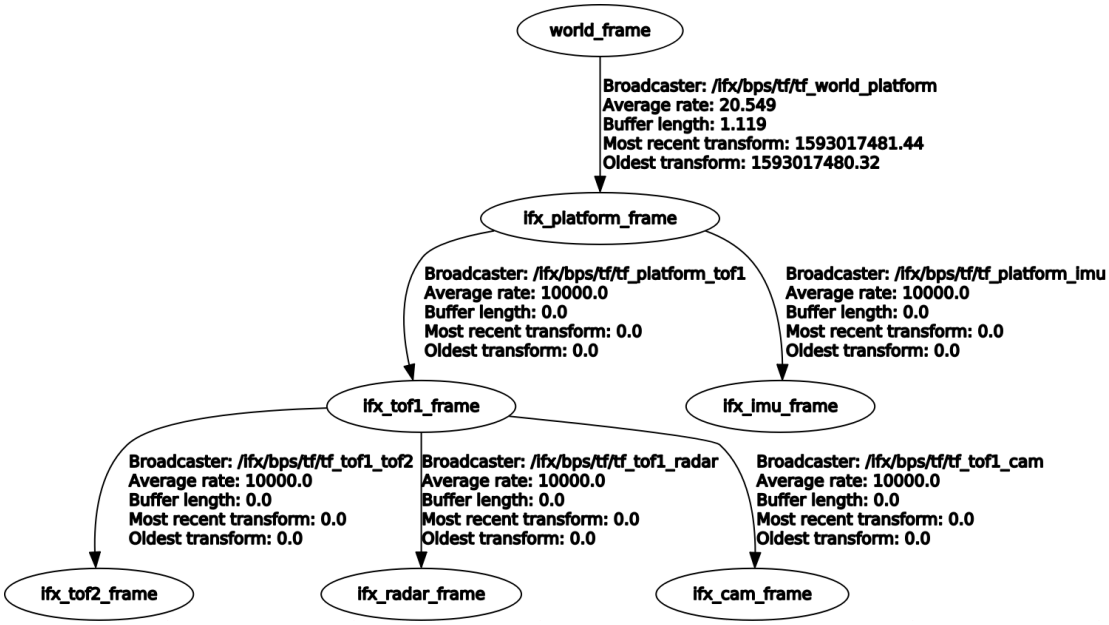


Figure 6.6: ROS transformation tree, visualized using a built-in visualization tool.

The transformation quality between two ToF point clouds is evaluated using the Euclidean distance between the detected target positions. For the transformation between radar and ToF, the 2D top-down distance between the different detected target points is considered. The transformation quality between the vision camera and the ToF camera is evaluated by projecting the ToF camera's detected target position onto the vision camera's image plane. The pixel distance between the target positions is utilized as a measure for the transformation quality.

For the proposed platform, the correctly executed calibrations resulted in a precise spatial alignment between the single sensors' frames. Since the transformation error is minimized during the calibration process, the resulting alignment error was in the range of the utilized sensors' measurement uncertainties. The determined transformation quality met the requirements of the implemented perception applications. However, if the perception platform is intended to be used for higher distances (e.g., over 20 m), additional calibration steps with targets at higher distances are required. The quality of the transformations has to be repeatedly evaluated since the sensor alignment can change over time (e.g., due to mechanical stress). If the transformation quality drops below a certain threshold, a re-calibration of the corresponding sensors' spatial alignment has to be performed.

The dynamic transformation between the world frame and the environmental perception platform drifts over time since solely relative pose changes are incorporated. However, the relative position change within a temporarily limited time-span is sufficiently accurate for various perception tasks (e.g., local mapping). The extent of the transformation's drift mainly depends on the pose estimation quality between consecutive ToF point clouds. The major influencing factors include the ToF camera's resolution, the number of overlapping points, and the pose estimation's rate. For typical system configurations, the transformation's deviation during the latest 30 seconds was sufficiently small to be neglected for the desired applications (e.g., the environment mapping use case).

6.2.3 Sensor Data Processing

The base perception system provides multiple measurement streams at different abstraction levels and metadata streams of supplementary measurement information. This subsection provides a complete list of the base perception system's data streams (final platform version), available to upcoming subsystems. Additionally, the base perception system's performance and data load are presented for typical configurations of the environmental perception system.

Provided Data Streams

All data streams published by the base perception system's modules are made available to upcoming subsystems (e.g., the implemented use cases) for further processing. Table 6.1 lists the base perception system's output data streams, eligible for subsequent processing. As seen in the table, the data streams are composed of measurement data streams (e.g., ToF point cloud, radar target list) and metadata streams (e.g., vision camera info, vision parameters). The data streams solely intended for internal use are omitted in the table since they are not further processed by any use case (e.g., the estimated measurement

Table 6.1: Output data streams of the base perception system.

Data Streams	Description
ToF point cloud	Full point cloud of the first/second ToF camera
ToF point cloud subsampled	Subsampled point cloud of the first/second ToF camera
ToF intensity image	Intensity image of the first/second ToF camera
ToF distance image	Distance image of the first/second ToF camera
ToF camera info	Camera metadata of the first/second ToF camera
ToF parameters	Parameter metadata of the first/second ToF camera
Vision image raw	Unprocessed camera image from the vision camera
Vision camera info	Camera information metadata of the vision camera
Vision parameters	Parameter metadata of the vision camera
Vision image undistorted	Undistorted camera image from the vision camera
Vision image subsampled	Subsampled camera image from the vision camera
Radar range-Doppler	Range-Doppler data from the radar sensor
Radar range-angle	Range-angle data from the radar sensor
Radar range-Doppler CFAR	CFAR thresholded radar range-Doppler data
Radar target list	List of detected radar targets
Radar parameters	Parameter metadata of the radar sensor
IMU data raw	Measurement data of the IMU sensor
IMU movement detection	Detected movements of the IMU sensor
Battery sensor value	Battery voltage from the battery monitoring module

timestamp from the trigger module). The output of individual data streams can be omitted by disabling them in the base perception system's parameters. In addition, entire modules and their associated processing branches can be disabled via launch file arguments (e.g., radar processing, ToF measurement acquisition).

Performance

The presented platform's performance highly depends on its utilized configuration. The base perception system's nodes, involved in the data (pre-) processing, introduce different latencies. The respective nodes' output data streams are published according to their associated processing latency. While the sensors' measurement data streams are made available almost immediately after their reception, the publication of high-level data streams can be significantly delayed (e.g., radar targets).

If the provided data rate is higher than a module's ability to process the data, frame drops will occur. Since a number of modules require synchronized input data streams, missing messages can lead to starvation effects. Frame drops at fundamental modules (e.g., within the base perception system) have to be avoided to provide the use cases with continuous output data streams. Thus, the processing load assigned to the base perception system's single modules defines the overall system's maximum feasible frame rate.

The maximum frame rate stated in this section describes the perception sensors' maximum measurement rate, allowing a stable operation (i.e., no frame drops). The deactivation of computationally expensive modules (e.g., point cloud manipulation) increases

the maximum frame rate but limits the selection of use cases. The system parameters can also be adapted in order to allow higher frame rates (e.g., reduce measurement data dimensions, reduce processing effort).

The final perception platform's performance was evaluated using its four perception sensors and the default configuration. If only the receive modules are enabled, the system can handle measurement rates of up to around 10 FPS. At higher rates, the USB connection is limiting the data transfer, causing frame drops to occur. If the raw measurement streams are additionally visualized, the maximum frame rate is further decreased.

Data Load

Similar to its performance, the base perception system's data load is also highly dependent on the system's active configuration. The provided data load from the employed sensors specifies the minimum data handling capabilities of the system (e.g., processing, recording). The data load of the platform's final version was examined for two different sensor configuration settings: high resolution and reduced resolution. Since the input data load of the base perception system is mainly determined by the perception sensors' measurement data streams (i.e., ToF, radar, vision), the remaining data streams were neglected for the following considerations (e.g., IMU data, battery status).

- Configuration 1: high resolution
  The ToF cameras were utilized at their full resolution (pixel binning=1). The radar sensor was configured to N=1024 samples, M=128 chirps, and L=16 antennas.

  One ToF point cloud of a single measurement results in 19.4 Mbit. A radar data cube amounts to 33.6 Mbit and an uncompressed Full HD image from the vision camera contains 55.3 Mbit. At a frame rate of 10 FPS, the combined sensor data results in a data rate of 159.4 MB/s.

- Configuration 2: reduced resolution
  In the second configuration scenario, the ToF cameras' resolutions were reduced (pixel binning=3). The radar sensor was configured to N=256 samples, M=64 chirps, and L=16 antennas.

  The resulting data loads of the individual sensor measurements amount to 2.1 Mbit for the ToF camera, 4.2 Mbit for the radar sensor, and 55.3 Mbit for the vision camera. At a frame rate of 10 FPS, the combined measurement data results in a total data rate of 79.7 MB/s. In order to further reduce the data load, the vision camera's output, a non-essential component for several perception tasks, was disabled. In this case, the resulting data rate dropped to 10.6 MB/s.

The nodes of the base perception system have to be able to cope with these data loads. Since an increased data load leads to an increased processing time, the system's maximum feasible frame rate is limited by the sensors' data load. In order to record datasets, the ROS has to be able to continually store these data streams into a rosbag file. Since a fast solid-state drive is used as the platform's data storage, the system was able to record the raw measurement streams for both examined configurations. In order to further increase the maximum data load for recording, the ROS framework's recording node provides the functionality to efficiently compress the data streams before they are written to the rosbag.

6.2.4 System Parameters

The base perception system includes a large set of parameters utilized to configure its contained nodes. The nodes can contain different types of parameters, including debug parameters (e.g., flags, logging level), data flow parameters (e.g., identifiers of the utilized topics/services/frames), processing parameters (e.g., algorithm threshold/resolution/flags), and sensor configurations (e.g., resolution, sample rate). The parameters are defined in the global configuration launch file, transferred to the ROS parameter server at startup, and fetched during the initialization of the system's nodes.

The base perception system's parameters influence the shape, structure, and quality of the provided data streams In addition, the selection of the parameters has a significant impact on the system's workload (e.g., processing load, latency, data rate). Thus, the selection of the system parameters is of paramount importance in order to provide a high perception quality. The parameters have to be defined according to the targeted application's requirements, the available processing capabilities, and the expected environment.

Figure 6.7 shows the visualization of the ToF and radar measurement data for two different system parameter sets. The data was acquired with the environmental perception platform's second version and shows an indoor office environment. The only differences between the two parameter sets are the radar processing node's FFT bins (range-angle FFT) and the ToF receive nodes' binning parameter. As seen in the figure, the variation of only these two parameters results in a high-resolution output (Figure 6.7a) and a low-resolution output (Figure 6.7b). Compared to the low-resolution parameter set, the high-resolution parameters result in a significantly increased data load and latency.

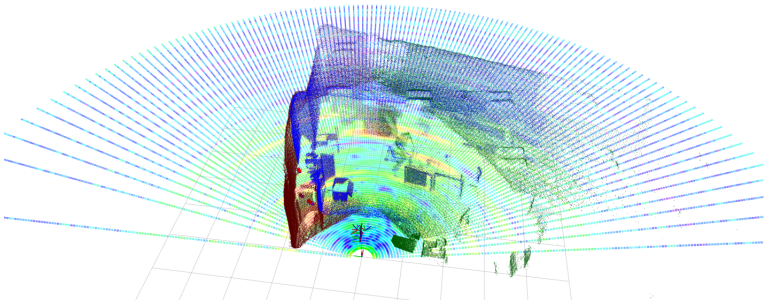
6.3 Use Cases

This section presents the obtained results during the evaluation of the implemented use cases. Since the main purpose of the use cases is to demonstrate the environmental perception platform's capabilities, common perception tasks were implemented (e.g., pedestrian detection, obstacle detection). The implemented perception tasks are based on well-established approaches and were adapted to the custom composition of this work's perception platform. Each use case's output is presented for one or multiple example scenarios and the use cases' general performance is evaluated.

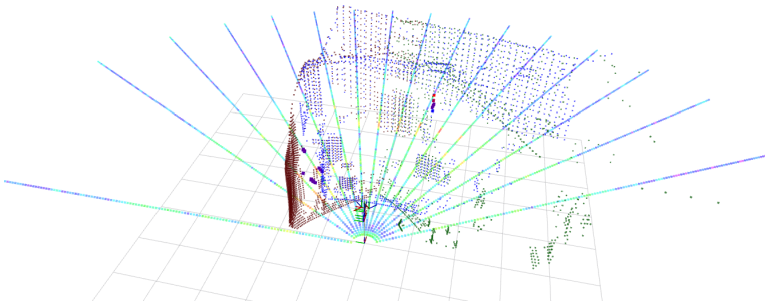
6.3.1 Context-Aware Parameter Adaption

The context-aware parameter adaption use case utilizes the system's perceived context state in order to adapt various system parameters to changing environments. The data from multiple context sensors, the data from the perception sensors, and the system's current state are considered to obtain information about the system's current context. This information is processed by the use case's parameter handlers, which propose the update of certain parameters in order to adapt the system to its current context.

The subsystem's capability is demonstrated by adapting the parameters of the obstacle detection use case's fusion module. The context-aware parameter adaption utilizes the available context information (e.g., the ambient light intensity) to determine a confidence value for each of the fusion module's input data streams. The corresponding parameter



(a) High-resolution parameters.



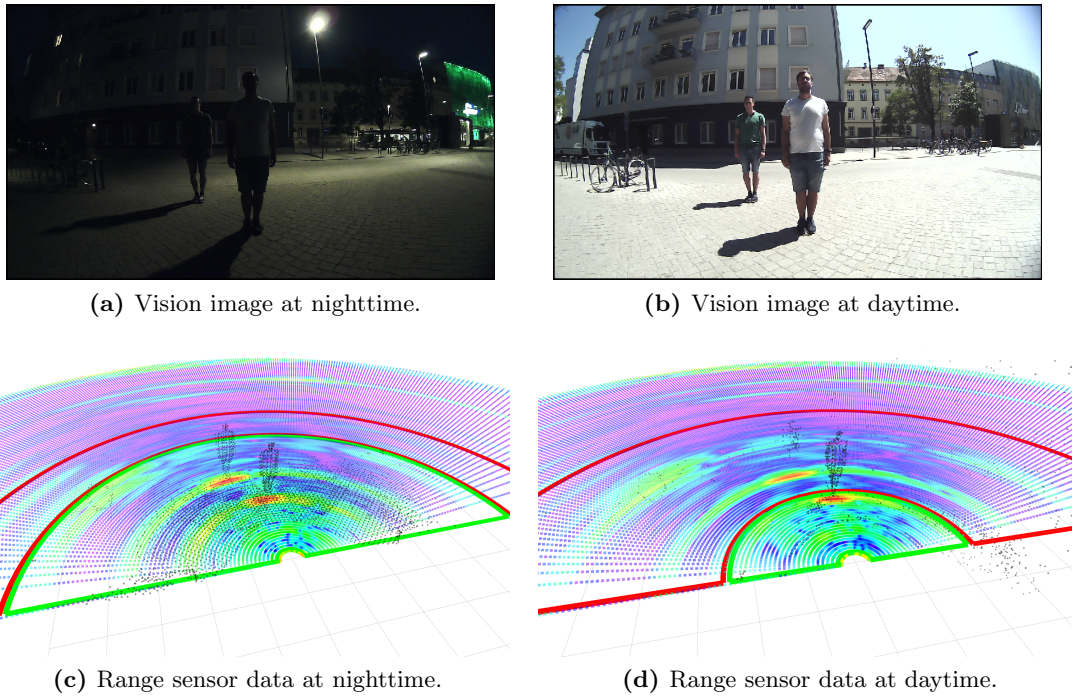
(b) Low-resolution parameters.

Figure 6.7: Impact of the system parameters on the resulting perception data. The ToF point cloud and the radar range-angle data are visualized for two different parameter sets [107].

handler initiates the adaption of the fusion module's confidence parameters (e.g., ToF confidence range, radar confidence range). The fusion module considers the sensor-specific confidence parameters for the fusion task in order to provide a reliable output stream. The adaptation of the platform's parameters to its current context state is crucial to preserve a high perception performance in changing environments.

Figure 6.8 shows an example scenario of the context-aware parameter adaption. The second version of the platform was placed on a stationary object and exposed to a similar scene at daytime and nighttime. As seen in the visualization of the corresponding ToF and radar range data, the ToF camera's perception quality is affected in the presence of bright sunlight. The context-aware parameter adaption subsystem recognizes this environmental state and degrades the ToF camera's data stream accordingly. The corresponding parameter handler determines the ToF camera's resulting confidence range and updates this value in the system's respective module. As seen in the figure, the associated confidence range is reduced, as indicated by the green area in front of the platform.

The use case also implements the context-aware adaption for additional system parameters of the base perception system (e.g., radar samples, ToF illumination time, processing thresholds). The obtained results show the potential of dynamic parameter adaption for environmental perception systems. The utilization of the current context state enables perception systems to significantly increase their robustness in changing environments. Our publication [107] presents a more comprehensive overview of the approach and is attached to this thesis (see Chapter 8, Publication 8).



(a) Vision image at nighttime.

(b) Vision image at daytime.

(c) Range sensor data at nighttime.

(d) Range sensor data at daytime.

Figure 6.8: Degradation of a range sensor affected by bright sunlight. A context-dependent confidence value is introduced in order to degrade affected data streams [107].

6.3.2 Obstacle Detection

The obstacle detection use case creates a local occupancy grid using multiple data streams from the ToF cameras and the radar sensor. The individual data streams are transformed into a common coordinate system before they are combined into an occupancy grid. The inclusion of the radar data in addition to the 3D ToF data enhances the confidence of detected obstacles and enables a common interpretation of the perception data. The combination of both sensors allows the platform to detect obstacles incapable of being trustworthily detected by the individual sensors' measurement data.

The use case is demonstrated in an outdoor scenario with multiple pedestrians in the perception sensors' common field-of-view. The data was recorded during a test drive with the second version of the environmental perception platform, mounted on the unmanned ground vehicle provided by the Virtual Vehicle. Figure 6.9b depicts an image of the described scenario, captured via the environmental perception platform's fisheye vision camera. There are four pedestrians present in the image, three standing still while one is walking away from the vehicle.

Figure 6.9a shows the use case's data flow to create the occupancy grid of this scenario. The input data streams consist of the merged and subsampled point cloud from three ToF cameras, the radar range-angle data, and the list of detected radar targets. As a first step, individual top-down grid structures are created for each of the input data streams. These individual grids provide sensor and data-specific representations of the perceived scenario. While the two closer pedestrians are clearly recognizable in the ToF grid, the

other two pedestrians are not distinctly detected anymore. The grid obtained from the radar peaks clearly detects the walking pedestrian due to its radial velocity, while the other persons result in weak peaks. The range-angle data stream acts as supplementary information, marking areas according to their radar reflectivity. During the fusion task, the individual grid structures are combined into a common occupancy grid. The fused occupancy grid utilizes the strengths of the individual sensors and results in a combined grid with enhanced expressiveness. As seen in the figure, the four pedestrians are clearly detected in the fused occupancy grid.

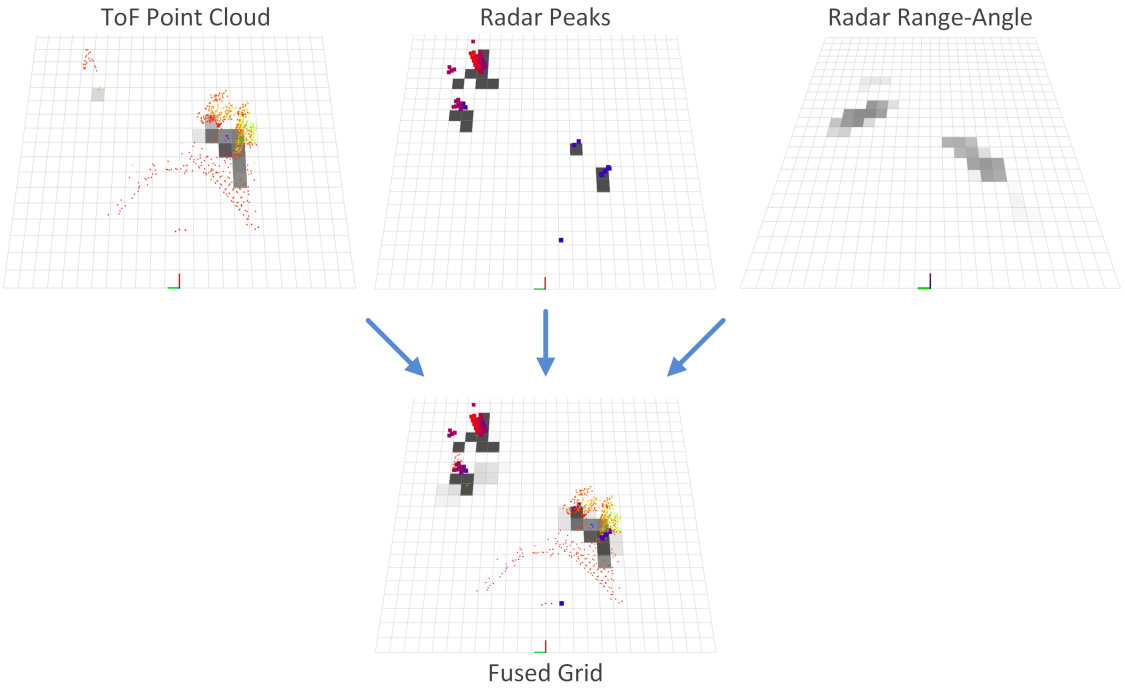
The resulting quality and performance of the obtained occupancy grid highly depend on the utilized system parameters. The most influential parameters include the sensor configurations (e.g., ToF resolution, radar sample number), the pre-processing parameters (e.g., cropped area, radar peak detection thresholds), and the use case's parameters (e.g., grid cell size, fusion parameters). These parameters have to be selected according to the requirements of the targeted application. In addition, the use case is able to create an occupancy grid map of the local environment. In that case, the last N occupancy grids are incorporated into a common map, utilizing the temporally accurate transformations between the individual measurements. More details about the use case are published in our conference paper [7], which is also included in this thesis (see Chapter 8, Publication 6).

6.3.3 Environment Mapping

The environment mapping use case creates a three-dimensional map of the environment utilizing the presented platform's perception data. For that purpose, the platform is moved through the desired environment (e.g., driven by a vehicle or manually moved) while the data from the perception sensors is accumulated. The ToF data streams are integrated into a common point cloud, representing a map of the environment.

In the use case's default configuration, the point cloud data stream from one ToF camera is transformed and added to a global point cloud, representing a map of the environment. The dynamic transformation between the platform's reference frame and the world frame has to be known and up-to-date in order to correctly align the data from different measurement times. The base perception system's spatial alignment module utilizes the pose change between consecutive ToF measurements and the measurement data from continuous IMU measurements in order to provide precise estimates of this transformation. The transformation and its change over time are managed by the ROS framework's TF tree and made available to all ROS nodes during runtime.

The environmental mapping use case is demonstrated using the final version of the developed environmental perception platform. In order to create a point cloud map, the platform was manually moved through an office environment over a time period of about 45 seconds. The utilized ToF camera was configured to a frame rate of 5 FPS, while the IMU sensor provides data at 100 Hz. The transformation between the world frame and the platform's frame is continuously estimated and utilized to align the consecutive point clouds into a common map. Since estimating the pose change between subsequent ToF point clouds is computationally expensive, the corresponding pose is provided at low rates. In the demonstrated scenario, these pose estimations were provided at rates between 0.5 and 1 FPS. The incorporation of the IMU data allows the estimation of the transformation during these pose updates.



(a) Occupancy grid data flow.

(b) Vision camera image.

Figure 6.9: Occupancy grid creation based on heterogeneous range data. The vision image is depicted as a reference but not included in the occupancy grid cration. Obtained with changes from [7].

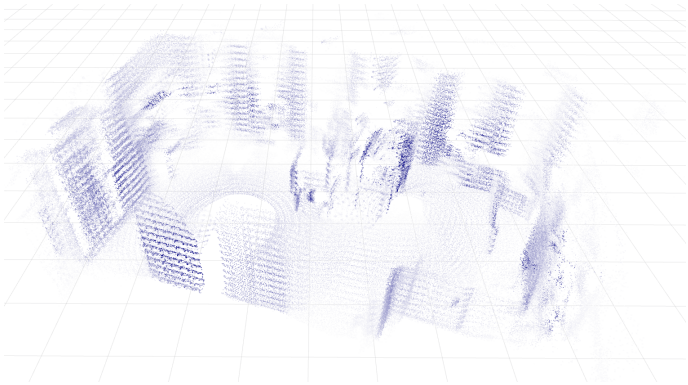


Figure 6.10: Map output of the environmental mapping use case.

Figure 6.10 shows the resulting output point cloud acquired during this process. The room's basic structure is clearly visible, containing multiple tables, shelves, and windows. As seen in the figure, the map incorporates a slight blur since the point clouds are not perfectly aligned. The reasons for that blur are inaccuracies of the estimated pose change, which lead to a drift of the transformation over time. The approach fully relies on the relative pose changes and does not utilize global measurements (e.g., GNSS) or feedback from the created map to compensate measurement drifts. Thus, the presented approach is limited to create maps for temporarily limited intervals (e.g., local mapping, occupancy grid maps). The perceived map's quality reflects the estimated transformation quality utilized for various perception tasks.

6.3.4 Pedestrian Detection

The pedestrian detection use case combines data from multiple perception sensors to detect pedestrians in the sensors' common field-of-view. The approach fuses the output of a HOG-based classifier and an augmented vision image in order to enhance the detection capabilities of the vision-only approach. The range measurements from the radar sensor and one ToF camera are projected onto the vision image and utilized to amplify regions, which potentially contain pedestrians.

The pedestrian detection use case is demonstrated using the platform's final version, placed on a stationary object. The example test scene shows a person walking towards the environmental perception platform. Figure 6.11 shows the data flow of the subsystem's modules for the example scenario. The upper-left image depicts the projected range data onto the vision camera's image. The radar targets are marked as blue vertical lines, while the ToF points are added as purple dots. Due to the simultaneous data acquisition and the synchronized data streams (via the timestamps), the range data is correctly aligned with the vision camera's image. The upper-right image shows the vision camera's image with marked bounding boxes, obtained from a moving window pedestrian detection using a HOG-based classifier. These two intermediate data streams are combined into a heatmap, utilizing the range data as an amplification value for the corresponding bounding boxes. The output of the heatmap is then thresholded in order to obtain an improved detection of the pedestrians in the scene. As seen in the output image, the walking pedestrian is correctly detected and marked with a bounding box.

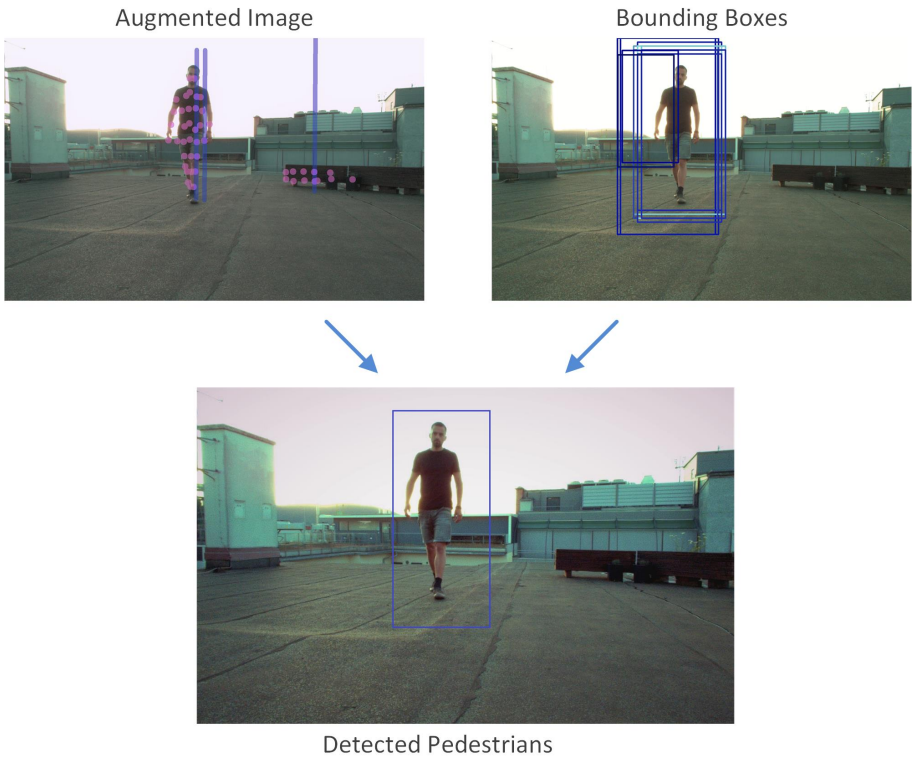
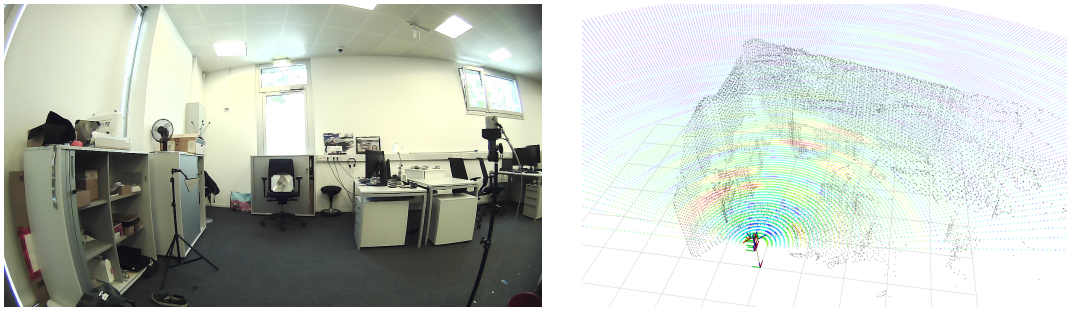


Figure 6.11: Pedestrian detection based on fused information from vision and range data [102].

Compared to the vision-only pedestrian detection, this approach detects pedestrians with increased confidence while keeping the false-positive rate low. Similar to the other use cases, the pedestrian detection use case can be customized via various parameters (e.g., size/shape of projected points, classification sensitivity, heatmap thresholds). In addition to the use case's parameters, the pedestrian detection's performance also depends on the base perception system's parameters (e.g., sensor configuration). In order to further increase the use case's performance, these parameters require additional fine-tuning according to the target environment. A more comprehensive description of the implemented pedestrian detection is available in our conference paper [102]. A reprint of that publication is attached to this thesis (see Chapter 8, Publication 9).

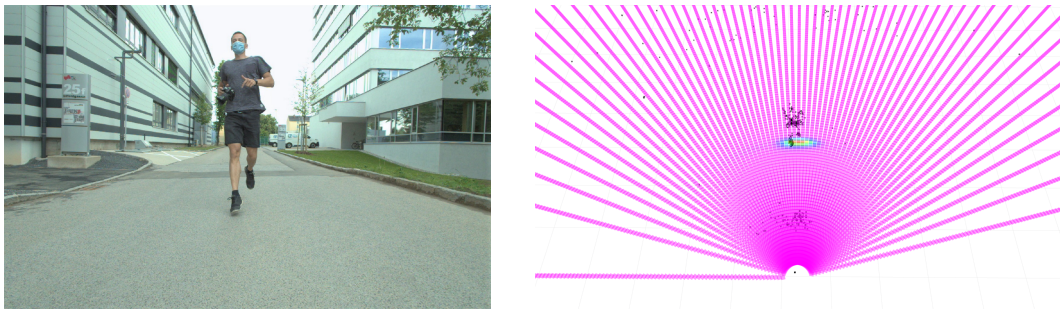
6.3.5 Data Visualization

The data visualization use case represents the environmental perception platform's data streams graphically. The ROS tool RViz is utilized to visualize different types of data streams provided by the base perception system or by other use cases. The visualization of the data streams is a valuable instrument to enable the visual evaluation of the system's perception performance. RViz utilizes the transformations provided by the base perception system's spatial alignment module in order to correctly align range data in the 3D coordinate space. In addition, the use case supports the conversion of not directly visualizable data streams to visualizable representations (e.g., detected radar targets).



(a) Fisheye camera image. **(b)** Range data.

Figure 6.12: Demonstration of the data visualization use case in an indoor scenario, recorded with the platform's second version.



(a) Vision camera image. **(b)** Range data.

Figure 6.13: Visualization of an outdoor scenario, recorded with the platform's final version.

The selection of visualized data streams and the corresponding display options in RViz differ significantly for the different use cases. Thus, there exist individual predefined launch files for each use case. These launch files include the corresponding use case's visualization configuration, including an associated display configuration for RViz.

The visualization use case is demonstrated in two different scenarios. The first scenario was recorded with the second version of the environmental perception platform, statically placed on a table. Figure 6.12a shows the acquired vision image of the corresponding office environment. The RViz visualization of the corresponding range data is depicted in Figure 6.12b. The tool visualizes the point clouds from the three ToF cameras and overlays the radar sensor's range-angle image. The second demonstrated scenario was acquired with the platform's final version, mounted on a passenger vehicle. The example scenario was recorded in an outdoor scenario and shows a pedestrian running towards the vehicle (see Figure 6.13a). The pre-processed range data of the corresponding measurement is presented in Figure 6.13b. The visualized ToF point cloud is a subsampled and filtered version of the full measurement. As seen in the figure, the radar range-angle data and the ToF camera's points of the pedestrian are correctly aligned.

Chapter 7

Conclusion and Future Work

This chapter highlights the main contributions of the thesis and states the work's influence on the associated research field. In addition, the research questions are answered, the major limitations of the platform are presented, and possible directions for future work are addressed.

7.1 Conclusion

Robust environmental perception is crucial to enable an autonomous operation of vehicles and robots in open environments. A robust perception ability is among the major safety requirements of potentially harmful machines and devices, which operate alongside humans. Any device deploying multiple sensors to perceive the environment can benefit from improved methods to enhance the provided output data quality. However, the research on novel sensor fusion concepts is restricted due to the limited availability of open datasets and open platforms capable of providing low-level sensor access.

This thesis presents the physical composition, the design, and the implementation of an environmental perception system with the ability to provide sensor data at various abstraction levels. Multiple state-of-the-art perception sensors were integrated into a standalone platform, capable of being mounted on various types of vehicles. The work addresses several key-challenges of perception systems, vital to enable the low-level fusion of heterogeneous sensor data. The presented approaches include the data alignment, the sensor synchronization, and the dynamic configuration of the system's parameters. Utilizing the provided sensor data, various sensor fusion concepts were implemented in order to enhance the performance of common perception tasks (e.g., object detection).

The constructed environmental perception platform evolved from a single-sensor solution to a multi-sensor perception platform. The final platform deploys a radar sensor, multiple ToF cameras, and a vision camera as perception sensors. The selected sensors allow the dynamic adjustment of their configurations, triggered data acquisition via an external signal, and are capable of providing low-level measurement data. Further, the platform includes additional context sensors (e.g., IMU, light sensor), a processing unit, and a lithium battery to allow the platform's independent operation. Since aluminum profiles are utilized as the base structure, the platform can be easily mounted on various platforms, including vehicles and robots.

The platform's fundamental component is the base perception system, a subsystem for data handling and early-stage processing. This subsystem implements several data-handling tasks and (pre-) processing steps, including the measurement data's spatial and temporal alignment The base perception system outputs structured and properly aligned data streams at different abstraction levels. Succeeding subsystems can utilize these data streams to perform application-specific processing tasks. In order to demonstrate the abilities of the platform, multiple use cases were implemented. These use cases include well-established perception tasks such as data visualization, context-aware parameter adaption, obstacle detection, and pedestrian detection.

Due to its rapid prototyping abilities and reputation in the research community, the ROS was selected as the base framework for the software implementation. Each subsystem's processing modules were implemented as ROS nodes. The data streams between these nodes are exchanged via ROS messages. Built-in tools of the ROS framework and community-provided modules were utilized to support and speed-up the implementation process. In addition, the framework allows straightforward integration of the environmental perception platform into other ROS-based systems (e.g., research vehicles).

The provided platform enables the conduction and evaluation of various sensor fusion concepts from low-level sensor fusion to high-level sensor fusion. Each of the base perception system's (pre-) processing modules performs a specified amount of processing before one or multiple output data streams are provided to succeeding modules. The perception platform supports the custom setting of the sensors' configuration parameters (e.g., exposure time, waveform type, sample rate). This enables the evaluation and exploration of the system's performance depending on the applied set of sensor configurations. Since the platform further allows the recording of datasets in addition to live processing, earlier recorded measurements can be used for later offline evaluation. The offline evaluation of recorded datasets enables the qualitative comparison of different processing algorithms (e.g., during the development phase).

As a practical demonstration, the proposed platform was mounted on different research vehicles to obtain data in real-world scenarios. Multiple datasets were recorded and utilized to evaluate the capabilities of the proposed environmental perception platform. For this purpose, the base perception system and the implemented use cases were operated with the recorded datasets and evaluated applying different types of parameter settings. The integration of low-level data has been shown to improve the system's perception performance for various use cases (e.g., obstacle detection or pedestrian detection). The fusion of data from multiple sensors enables the distinct detection of objects insufficiently perceived by any individual sensor. For example, the radar sensor's velocity information can be exploited to amplify the corresponding, low-confidence ToF points. The implemented context-aware parameter adaption use case utilizes the perceived context information to adapt the system's parameters (e.g., sensor configuration, algorithm parameters). The activation of this use case results in an improved perception performance in changing environments.

The proposed environmental perception platform is Infineon Technologies' first internally developed multi-sensor perception system, enabling simultaneous data acquisition with heterogeneous sensors. The perception system enables rapid evaluation of sensor capabilities in a multi-sensor system while preserving full control of the low-level sensor configurations. Since the platform can be mounted on various vehicles, the provided sen-

sor data can be utilized to perform state-of-the-art perception tasks in dynamic real-world scenarios. This enables the early identification of sensor weaknesses in the targeted environment, which is of high importance in product-to-system-oriented businesses. One limitation of the ToF camera, identified during an evaluation run (missing distance information in the area of saturated pixels), resulted in a patent application of the mitigation approach [108] (see Chapter 8, Patent 1).

This work's approach can be utilized as a reference for the development of multi-sensor perception platforms. The presented work provides solutions to numerous challenges arising during the development of low-level capable environmental perception platforms. Researchers can employ this knowledge in order to speed-up the design and implementation of similar platforms. This enables a faster advancement of research on low-level sensor fusion and further contributes to the development of more robust and reliable perception systems.

7.1.1 Answers to the Research Questions

In Chapter 1 of this thesis, three main research questions were formulated. The obtained results and findings during the work on this thesis can be utilized to answer these questions.

R1 *What is a feasible design and implementation of an environmental perception platform capable of performing a low-level fusion of heterogeneous sensors?*

This work's approach separates the platform into the base perception system and multiple use-case subsystems. The base perception system handles the low-level interaction with the sensors and provides aligned perception data at different abstraction levels to the use cases. The amount of (pre-) processing performed by the base perception system is individually adjusted to the active use case(s).

R2 *How can single components of a sensor fusion system be configured to obtain a satisfactory perception performance in changing environments?*

The context-aware parameter adaption use case presents an approach to adapt the system's parameters during runtime. Multiple parameter handlers utilize the current system state to determine new values for sensor configuration parameters and system parameters (e.g., sample rate, illumination time). The parameters are dynamically assessed using system knowledge, heuristics, and iterative feedback-based methods.

R3 *How can low-level fusion of radar and ToF data contribute to enhance the quality of state-of-the-art environmental perception systems for robotic/automotive applications?*

As seen in the implemented use cases, the fusion of radar data with high-resolution distance data can be beneficial for various applications. Radar sensors are capable of operating in harsh weather conditions, whereas light-based sensors typically provide higher resolutions. The complementary strengths of the heterogeneous sensors can be utilized to increase the robustness of the fused data. The obtained results indicate that the additional data from radar sensors can enhance the performance of a variety of perception tasks.

7.1.2 Limitations

The platform's main restriction is the limited selection of perception sensors, as sensor fusion strategies can only be evaluated for the provided types of sensor data (ToF, radar, vision). However, the modular design of the base perception system allows fast integration of additional perception sensors.

The utilized radar sensor cannot perform a parameter adaption on-the-fly, as the development board has to be restarted after each adaptation of the sensor's configuration. The introduced delay limits the platform's adaptation speed and causes radar frames to be dropped during that transition. The deployed ToF cameras provide a similar output as the commonly used lidar sensors. However, the operating range of ToF cameras is generally inferior to the range of lidar sensors.

The presented time synchronization approach relies on an estimation of a notification message's delay and an expected frame delay. Thus, the precision of the obtained timestamps is limited by the deviation of the message's delay from the estimated delay. In addition, the calibration approach of this thesis requires the manual placement of multiple artificial targets and the manual engagement into the procedure.

Multiple factors limit the maximum frame rate of the platform. Although the single sensors' measurement durations (for standard configurations) allow frame rates of more than 20 FPS, these rates are not achievable in practice. The major limiting factors are the system's processing performance and the bandwidth to the sensors. For this reason, the base perception system is incapable of handling high frame rates, especially if the majority of processing modules are enabled. The maximum feasible rate highly depends on the utilized system parameters (e.g., sensor resolution, algorithm parameters).

7.2 Directions for Future Work

This section presents possible directions for future work based on the outcome of this thesis. The proposed future work is separated into three main categories: the research on perception applications, the platform's optimization, and the recording of datasets.

7.2.1 Research on Perception Applications

The most straightforward direction for future work is the platform's intended usage: to perform research on environmental perception systems and their applications. A first step is the adaption of additional well-established perception tasks (e.g., localization, obstacle recognition) to the presented platform. The platform's low-level access to the sensors enables research on all layers of the perception processing chain. Possible research areas range from low-level tasks, such as data handling or self-adaption, to high-level tasks, such as localization or object detection. Additionally, the low-level data streams can be utilized to implement and evaluate novel sensor fusion concepts.

The availability of spatially and temporally synchronized perception data also enables the utilization of machine learning-based methods on different data abstraction levels. However, since the learning-based approaches require a high amount of training data, a large dataset of labeled data is vital to deploy this approach successfully.

Additional representations of the sensors' raw data could be integrated in order to evaluate their benefit for various perception tasks. This includes the radar range-Doppler signature of targets and the variation of the corresponding radar response over time. The so-called *micro-Doppler* data (as described in [109]) can be used for object recognition in addition to the data gathered from the ToF and the vision camera.

7.2.2 Platform Optimization

Future work can also assess further improvement of the environmental perception platform's hardware and software capabilities. This subsection introduces several approaches to optimize the proposed platform.

Hardware

Additional perception sensors of different types (e.g., lidar, thermal camera, ultrasonic) could be added to the platform in order to provide more heterogeneous data streams. The platform could be equipped with additional context sensors to enhance the obtained context information and improve its self-adaptation capabilities. However, the inclusion of additional sensors also results in an increased demand for interfaces and processing power.

The existing sensors could be exchanged with more powerful sensors of the same type to extend the platform's target applications. For example, the radar sensor could be exchanged with an alternative model capable of additionally providing the azimuth angle and adapting the sensor configuration on-the-fly. The used ToF cameras could be upgraded to models with a more powerful illumination unit, capable of operating at distances of up to 50 m. The vision camera could be replaced by a version, which already performs fundamental processing steps directly on the camera, like compressing the image data.

Hardware modifications could also be utilized to improve the processing performance of the platform. The processing unit (Intel NUC) could be interchanged with powerful CPU/GPU combinations or an automotive supercomputer (e.g., Nvidia Drive platform) to allow computationally expensive processing (e.g., deep learning). FPGAs could be added to the platform to perform certain low-level pre-processing in hardware (e.g., radar FFT or ToF point cloud calculation).

From a safety viewpoint, the system could be extended with a safety-focused microcontroller to monitor non-deterministic processing tasks. The additional supervision enables the system to detect failures and react accordingly, improving the its overall robustness. Depending on the targeted applications, the microcontroller could also handle the system's critical real-time tasks.

Software

The proposed perception system utilizes the *Kinetic Kame* distribution release of the ROS framework. In order to benefit from its most recent advancements and functionalities, an update to the latest version is recommended in the future (support of Kinetic Kame ends 2021). Alternatively, the software could be ported to the ROS 2 framework, a major advancement of the original ROS framework, addressing the initial framework's limitations (e.g., real-time performance, python 3 support).

In addition, the software performance of the platform's base perception system could be further optimized in the future. One possibility is the utilization of ROS nodelets, a concept that allows a zero-overhead copy of data between ROS nodes (shared pointer). Since nodelets are only supported for C++ interfaces, all python-based nodes have to be ported beforehand.

7.2.3 Dataset Recording

The environmental perception platform could be utilized to record more versatile datasets in various environments. These datasets could then be used to support the development of algorithms and to evaluate their performance in reference scenarios. Additionally, the recorded data could be labeled with ground truth information to enable the utilization of the datasets to train learning-based algorithms. A well-structured dataset can furthermore be published online with the main goal to provide the research community with the low-level perception data obtained from the platform.

Datasets can be recorded in various environments to support the development of robust perception algorithms. The platform can be either mounted on different types of vehicles or statically placed in the environment. Additionally, the datasets can be recorded in different environmental conditions (e.g., highway driving, city, fog, snow, sunshine). These datasets can then be utilized to determine a perception platform's limits and to identify edge cases. Identifying and mastering edge cases is of major importance to move the technology forward and to improve the robustness of future systems.

Chapter 8

Publications

The work conducted during this thesis has resulted in multiple publications to the scientific research community. Eight peer-reviewed conference papers, one book chapter, and one patent application are attached to the thesis. Figure 8.1 illustrates the association of the single papers to the contributions and the objectives, defined in Chapter 1. Identifiers for the publications (1-9) and the patent application (P1) are put into the circles next to one or multiple associated contributions. Due to the continuous advancement of the perception platform, certain contributions were addressed in multiple publications. This chapter briefly outlines each publication in order to provide an overview of the attached publications and their connections to each other. In addition, an authors' contribution statement is provided for each work, describing how the workload of the publications was distributed among the associated authors. Subsequently, reprints of the single publications are attached to this chapter.

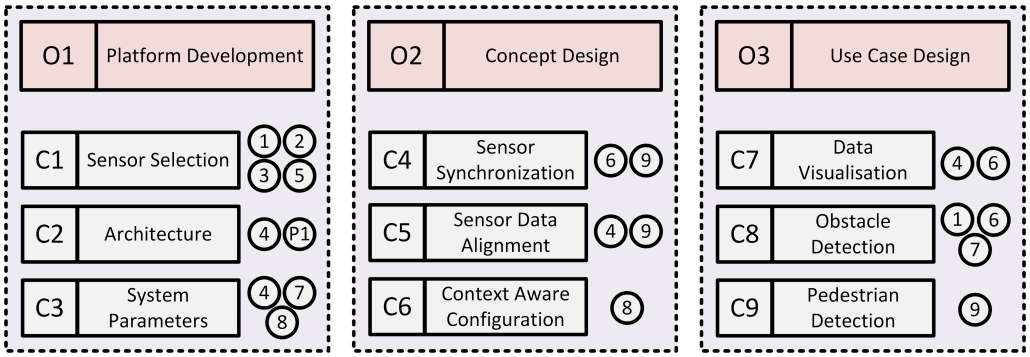


Figure 8.1: Assignment of the publications (1-9, P1) to the contributions (C1-C9) and the objectives (O1-O3) of this thesis.

A practical evaluation of a ToF camera for environmental perception, implemented on an automotive safety-focused microcontroller, is described in **Publication 1**. The paper presented in **Publication 2** covers the state-of-the-art of automotive radar sensors and highlights the importance of these sensors in perception systems due to their ability to work in harsh weather conditions. **Publication 3** provides a general overview of ToF cameras and presents a way to utilize ToF cameras to perform localization. In **Publication 4**, a platform is presented, capable of combining the data from radar and ToF. The

utilized architecture can be used as a base to fuse radar and ToF data. The feasibility of ToF cameras for parking assistance and collision avoidance is presented in **Publication 5**. The work published in **Publication 6** shows an approach to fuse ToF and radar data into a common occupancy grid. **Publication 7** shows the incorporation of the environmental perception platform into a robot research vehicle. The robot utilizes the mounted perception platform in order to detect obstacles and to evaluate its localization precision. The approach published in **Publication 8** outlines the context-aware adaptation of the environmental perception platform's parameters during runtime. **Publication 9** presents a novel method to timestamp triggered measurements and demonstrates a use case, performing pedestrian detection. A novel approach to detect and filter erroneous ToF pixels next to overexposed areas caused by highly reflective objects is described in **Patent 1**.

List of Included Publications

Publication 1: J. Steinbaeck, A. Tengg, G. Holweg, and N. Druml, *A 3D Time-of-Flight Mixed-Criticality System for Environment Perception*, 2017 Euromicro Conference on Digital System Design (DSD), 30 Aug.-1 Sept. 2017, Vienna, Austria.

Author contributions: conceptualization JS, AT, and ND; state-of-the-art research JS; development of methodology JS; hardware/software design JS; conduction of experiments JS; data analysis and interpretation JS; drafting the manuscript JS; critical revision AT and ND; final approval GH.

Publication 2: J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, *Next Generation Radar Sensors in Automotive Sensor Fusion Systems*, 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF), 10-12 Oct. 2017, Bonn, Germany.

Author contributions: conceptualization JS; state-of-the-art research JS; drafting the manuscript JS; safety considerations subsection: ND; critical revision CS and ND; final approval GH.

Publication 3: H. Plank, J. Steinbaeck, N. Druml, C. Steger, and G. Holweg, *Localization and Context Determination for Cyber-Physical Systems Based on 3D Imaging*, Solutions for Cyber-Physical Systems Ubiquity, IGI Global, 2018.

Author contributions: conceptualization HP and ND; state-of-the-art research HP and JS; development of methodology HP; hardware/software design HP; conduction of experiments HP; data analysis and interpretation HP; drafting the manuscript HP and JS; critical revision HP, JS, ND, and CS; final approval GH.

Publication 4: J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, *Design of a Low-Level Radar and Time-of-Flight Sensor Fusion Framework*, 2018 21st Euromicro Conference on Digital System Design (DSD), 29-31 Aug. 2018, Prague, Czech Republic.

Author contributions: conceptualization JS, CS, and ND; state-of-the-art research JS; development of methodology JS; hardware/software design JS; conduction of experiments JS; data analysis and interpretation JS; drafting the manuscript JS; critical revision CS and ND; final approval GH.

Publication 5: J. Steinbaeck, N. Druml, A. Tengg, C. Steger, and B. Hillbrand, *Time-of-Flight Cameras for Parking Assistance: A Feasibility Study*, 2018 12th International Conference on Advanced Semiconductor Devices and Microsystems (ASDAM), 21-24 Oct. 2018, Smolenice, Slovakia.

Author contributions: conceptualization JS, ND, and BH; state-of-the-art research JS; development of methodology AT and BH; hardware/software design AT and BH; conduction of experiments AT and BH; data analysis and interpretation JS; drafting the manuscript JS; critical revision ND, CS, and BH; final approval BH.

Publication 6: J. Steinbaeck, C. Steger, E. Brenner, G. Holweg, and N. Druml, *Occupancy Grid Fusion of Low-Level Radar and Time-of-Flight Sensor Data*, 2019 22nd Euromicro Conference on Digital System Design (DSD), 28-30 Aug. 2019, Kallithea, Greece.

Author contributions: conceptualization JS; state-of-the-art research JS; development and implementation of methodology JS; conduction of experiments JS; data analysis and interpretation JS; drafting the manuscript JS; critical revision CS, EB, and ND; final approval GH.

Publication 7: J. Steinbaeck, N. Druml, T. Herndl, S. Loigge, N. Marko, M. Postl, G. Kail, R. Hladik, G. Hechenberger, H. Fuereder, C. Steger, E. Brenner, and C. Schwarzl, *ACTIVE - Autonomous Car to Infrastructure Communication Mastering Adverse Environments*, 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), 15-17 Oct. 2019, Bonn, Germany.

Author contributions: conceptualization TH, ND, HF, and C Schwarzl; development and construction of robot SL, NM, MP, and C Schwarzl; design and implementation of road side unit GK, RH, GH, and HF; design and implementation of environmental perception platform JS; conduction of experiments JS, SL, NM, MP, RH, and CS; localization accuracy analysis and interpretation GK, RH, GH, and HF; perception data analysis and interpretation JS; drafting the manuscript JS; critical revision ND, RH, C Steger, EB, and C Schwarzl; final approval C Schwarzl.

Publication 8: J. Steinbaeck, A. Strasser, C. Steger, E. Brenner, G. Holweg, and N. Druml, *Context-Aware Sensor Adaption of a Radar and Time-of-Flight Based Perception Platform*, 2020 IEEE Sensors Applications Symposium (SAS), 9-11 Mar. 2020, Kuala Lumpur, Malaysia.

Author contributions: parameter adaption concept JS; sensor degradation concept AS; state-of-the-art research JS; hardware/software design JS; conduction of experiments JS and AS; data analysis and interpretation JS and ND; drafting the manuscript JS; critical revision AS, CS, EB, and ND; final approval GH.

Publication 9: J. Steinbaeck, C. Steger, E. Brenner, and N. Druml, *A Hybrid Timestamping Approach for Multi-Sensor Perception Systems*, 2020 23rd Euromicro Conference on Digital System Design (DSD), 26-28 Aug. 2020, Kranj, Slovenia.

Author contributions: conceptualization JS, CS, and ND; state-of-the-art research JS; development and implementation of methodology JS; conduction of experiments JS; data analysis and interpretation JS; drafting the manuscript JS; critical revision CS, EB, and ND; final approval ND.

Patent 1: J. Steinbaeck, H. Plank, and A. Schoenlieb, *Time of Flight Sensor Module, Method, Apparatus and Computer Program for Determining Distance Information Based on Time of Flight Sensor Data*, European Patent Application, filed 2018.

Author contributions: identification of problem JS; conceptualization HP and AS; design of methodology JS, HP, and AS; data collection JS, AS, and HP; data analysis and interpretation JS, AS, and HP; drafting the methodology for the patent application JS, HP, and AS.

A 3D Time-of-Flight Mixed-Criticality System for Environment Perception

Josef Steinbaeck*, Allan Tengg†, Gerald Holweg*, and Norbert Druml*
*Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, gerald.holweg, norbert.druml}@infineon.com
†Virtual Vehicle Research Center, Graz, Austria
allan.tengg@v2c2.at

***Abstract*—Automated driving systems have to operate at the highest level of robustness and safety. Thus, redundancy and diversity of the deployed systems are inevitable in order to guarantee the functionality in any possible scenario. Today, the most used sensor technologies for environment perception are color cameras, radar, light detection and ranging (LIDAR), and ultrasonic sensors. This work evaluates the feasibility of a 3D Time-of-Flight (ToF) camera to be used as environmental perception sensor in robotics and automated/assisted driving.**

To examine the performance of the sensor in the field, a ToF processing platform is attached to a 1:5 scaled remote control vehicle. An algorithm, which detects and reacts to obstacles in real-time, is designed and implemented on an AURIX automotive safety-microcontroller as major part of a mixed-criticality application. The embedded system highly benefits from the low computational effort required by ToF imaging (in contrast to stereo vision). Utilizing all three cores of the AURIX, the system achieves frame-rates of up to 30 frames per second (FPS).

***Index Terms*—Time-of-Flight, 3D imaging, infrared image sensors, accident prevention, robotics, unmanned vehicles**

I. Introduction

Environmental perception sensors are one of the key technologies in the robotic/automotive industry to enable autonomous systems. Multiple sensor technologies are used in parallel to wipe out weaknesses of single sensors, targeting a very robust digital representation of the environment in any possible scenario. The most valuable environmental perception sensors as of today are color cameras, radar, LIDAR, and ultrasonic sensors. Multi-sensor data fusion is crucial to mitigate individual weaknesses and accomplish the desired safety standards required for automated driving functionality. Thus, the industry is permanently aiming to discover and exploit new sensor technologies in order to add value and robustness.

ToF cameras are an uprising technology in this sector. In contrast to stereo-vision, they provide high frame-rate 3D data with minimal computational overhead. Additionally, ToF cameras are very compact and inexpensive. As the name indicates, the travel time of light is measured to obtain distance information of a scene. In the indirect ToF approach, the scene is illuminated with continuously modulated infrared light and the phase difference between the transmitted and the reflected light is measured [1]. Utilizing the speed of light, the distance can then be efficiently determined. A practical approach is to place an array of photonic mixing device (PMD) pixels behind an optical lens. The obtained sensor data is then used to calculate a distance image and an intensity image of the scene. A graphical illustration of the ToF principle is shown in Fig. 1. The work presented in [2] includes further details of the ToF principle.

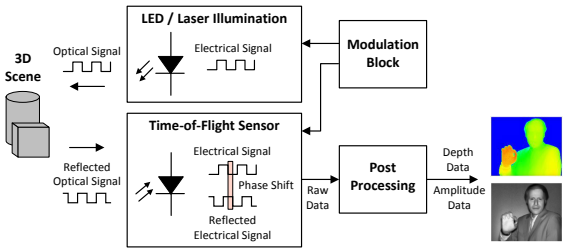


Fig. 1. Principle of ToF 3D imaging using continuously modulated infrared light [2].

In this work, we evaluate the feasibility of a 3D ToF camera in order to be used in robotics as well as automated/assisted driving systems. For this purpose, an automotive safety microcontroller is used to process the data from a state-of-the-art ToF camera. The successful implementation of a basic object detection algorithm is demonstrated by mounting the platform on a 1:5 scaled remote controlled vehicle. To sum up, the contributions of this paper to scientific research are:

- The design of a lightweight object detection algorithm based on raw data from a ToF camera.
- An implementation of the algorithm on a resource-constrained automotive safety controller as part of a mixed-criticality application using a pipelining approach to maximize the throughput.
- The practical evaluation of a ToF sensor for automotive environment perception with a scaled vehicle as demonstrator.

Section II introduces related approaches based on distance data and ToF cameras. The design and implementation of the image processing algorithm on an automotive microcontroller is described in Section III. The Sections IV and V show the performance of the ToF processing platform mounted on a scaled vehicle and discuss the potential of ToF sensors in the robotic/automotive industry.

II. Related Work

This section gives an overview of some techniques on how to compensate imperfections of ToF cameras. Additionally, some related applications considering object detection are presented.

A. Time-of-Flight Cameras

Computing the distance image from raw ToF sensor data is a straightforward task and requires low computational effort. However, the image values come with systematic and random errors, such as temperature and distance dependent errors, global offset, and fixed pattern noise. Calibration and mathematical models can be used to compensate the well-known systematic errors. The authors of [3] give an in-detail overview of the different errors and methods for their correction. However, every error compensation adds an extra computation time to the distance data calculation and reduces the maximum frame-rate of the system.

There exist approaches to speed-up the error compensation by implementing parts of the distance data computation and error compensation in hardware. The authors of [4] present an approach to implement the 3D distance computation and various time-consuming error compensation steps on an FPGA.

Interfering infrared light, transmitted at the same wavelength as by the ToF camera, decreases the sensor's signal-to-noise-ratio (SNR) and can cause pixels to saturate. Thus, ToF cameras can become impractical when direct sunlight is present. The authors of [5] show a method to minimize the influence of ambient light by using a method called suppression of background illumination (SBI). ToF cameras with SBI can be used in situations where ambient light is present and are not limited to indoor usage.

One possibility to enhance the expressiveness of the distance image is by fusing it with a color image of the same scene, as shown in [6]. This can however be challenging for dynamic scenes, since it requires the ToF camera and the color camera to be precisely synchronized.

B. Object Detection with 3D Cameras

Since the distance image obtained from ToF cameras is similar to 3D images from stereo cameras, many existing algorithms initially designed for stereo are also applicable to ToF images. This is beneficial since certain applications can profit from the characteristics of ToF imaging. Advantages of ToF include the lowlight performance due to the active illumination and the relatively low computational overhead. Thus, also novel methods specializing on ToF and utilizing these advantages have been introduced.

The work published in [7] shows an efficient way to segment a scene and to classify pedestrians using 3D data. Existing approaches of ToF cameras in the automotive industry target driver monitoring [8] and human-machine-interaction (HMI) [9]. A ToF camera as part of a front-view application is presented in [10], where a measurement range of over 35 m could be achieved. The authors of [11] show an approach to utilize ToF cameras for optical vehicle-to-vehicle (V2V) communication. General applications of ToF cameras include pose tracking [12], indoor navigation [13] and augmented reality.

To the best of our knowledge, there is a gap in literature concerning real-time 3D environment perception using safety-focused microcontrollers. This paper provides an innovative contribution to the ongoing discussion in this important field of research [2] [9].

III. Time-of-Flight based Environmental Perception System

This section describes the components of the setup built in this work and shows the structure of the implemented algorithm on the mobile platform.

A. Time-of-Flight Camera

As ToF camera, an evaluation board from Infineon Technologies was selected. It houses the IRS1020C, one of the first generation 3D image sensors developed and produced by Infineon Technologies AG in cooperation with PMD Technologies AG. The sensor has a resolution of 352 x 188 pixels, comes with SBI to enable outdoor applications and supports frame-rates of over 100 FPS. The raw sensor data can be accessed via a parallel interface (PIF) or a serial interface (CSI-2). To increase the readout speed, 2 x 2 binning and the selection of a centered region of interest are enabled on the imaging chip. This reduces the transmitted resolution to 160 x 120 pixels and improves the SNR.

A dynamic on-the-fly reconfiguration is possible via I^2C. The most influential configuration parameters and their impacts are:

- the modulation frequency, influencing the unambiguous range and accuracy,
- the exposure time, affecting the SNR and saturation effects, and
- the frame-rate, influencing the time between two ToF images are captured.

The evaluation board also consists of an illumination board, equipped with two infrared LEDs. With this relatively weak, but eye-safe illumination unit, a minimum effective working range of 4 m can be expected. However, existing work has shown that using a stronger illumination unit can extend the measurement range to more than 35 m [10].

B. Automotive Microcontroller

As core component, an evaluation kit equipped with the TC299TF automotive microcontroller from Infineon Technologies AG was chosen. This 32-bit microcontroller represents the high-end segment of the AURIX family and offers real-time capabilities and state-of-the-art safety features. The ISO 26262 compliant AURIX architecture supports safety requirements up to Automotive Safety Integrity Level D (ASIL-D). The chip consists of three independent TriCore CPUs, each based on the combination of a RISC/DSP/MCU core. The AURIX chip is equipped with a camera interface (CIF), directly connected to the extended memory of the microcontroller. Hence,

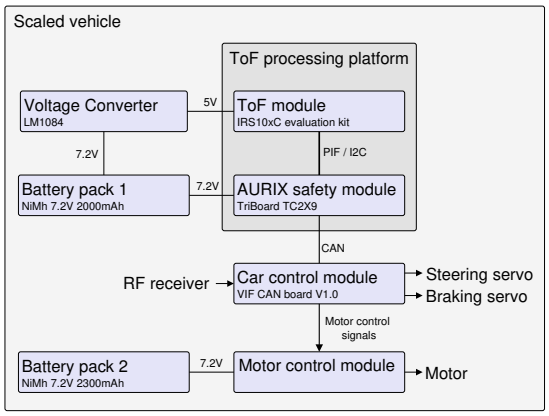


Fig. 2. Block diagram of the scaled vehicle's main electronic components and their connections.

received image data via the PIF is directly accessible by the controller. The evaluation kit implements several interfaces to communicate with other systems, for example Ethernet and controller area network (CAN).

C. Scaled Vehicle

A 1:5 scaled, electrical vehicle provided by the Virtual Vehicle Research Center in Graz is used to demonstrate the implemented algorithm with practical sensor data in dynamic scenes. The car can be controlled using a radio frequency (RF) remote and reaches a maximum speed of 25 km/h.

The vehicle houses a car control module, which is in charge of steering, braking, and controlling the engine. The module can be configured to periodically transmit the current state of the car via its implemented CAN bus interface. The current car state includes the actual speed, the remote signals, the motor voltage, and temperatures measured at significant points of the vehicle. The car can also be driven by wire. In this case, the control signals from the remote are overwritten by values received via the CAN interface.

To attach the bulky ToF processing platform to the vehicle, a flexible camera arm is used. The connection between the AURIX microcontroller and the ToF camera is established via the PIF and I^2C using an adapter board. Additionally, the AURIX microcontroller is connected to the car control module via a CAN bus. The contained blocks of the connected platform are illustrated in Fig. 2. The vehicle requires two separate power supplies in order to avoid power line variations of the electronic modules during high load situations of the engine. A front-view picture of the scaled vehicle with the ToF processing platform mounted in forward direction is depicted in Fig. 3.

D. Algorithm Design

Due to the limited image processing capabilities of the automotive safety microcontroller, a fast and low-complexity algorithm is designed in order to enable a high execution-rate

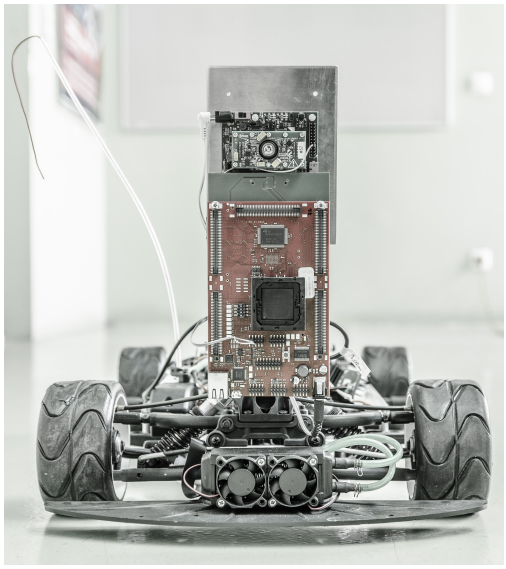


Fig. 3. Front view of the scaled vehicle. The ToF processing platform, consisting of separate boards for the automotive microcontroller and the ToF camera, is mounted on top of the vehicle.

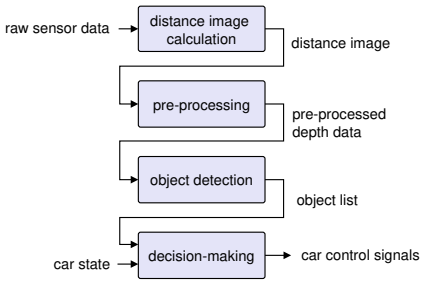


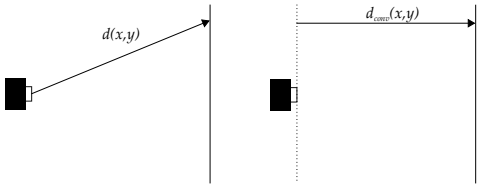
Fig. 4. The main image processing tasks with their input and output data.

and a low latency. The algorithm consists of the four major steps shown in Fig. 4.

1) Distance image calculation: The first processing step takes the raw sensor data as input and calculates the distance image as well as the amplitude image. To obtain a phase difference from the raw sensor data, the arctangent function has to be calculated (as shown by the authors of [1]). The phase difference value $\Delta\varphi$ is then used together with the modulation frequency f_{mod} and the speed of light c_0 to determine the distance $d(x, y)$ for every pixel (see Equation 1).

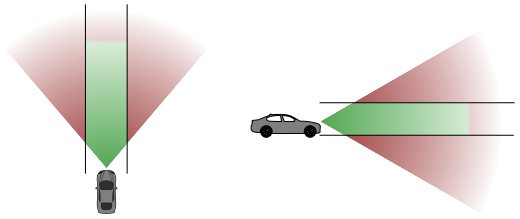
$$d(x,y) = \Delta\varphi \cdot \frac{1}{2} \cdot \frac{c_0}{2\pi \cdot f_{mod}} \tag{1}$$

Eventually, the global offset (one of the systematic errors described in [1]) is compensated and followed by an unambiguous range shift in order to keep the distance values within the unambiguous phase-interval.



(a) Before. (b) After.

Fig. 5. Distance conversion to extract the z-component of the distance points.



(a) Top view. (b) Side view.

Fig. 6. The region directly in front of the vehicle (filled greed) corresponds to the area of interest for the object detection. Distance values outside this region are discarded and not considered for object detection (filled red).

The arctangent is a high-cost function in software and consumes the majority of the computation time within this processing step. Thus, it is crucial to select the best fitting arctangent algorithm for the target architecture in order to minimize the introduced computation delay. We selected a lookup table with linear interpolation as arctangent algorithm. Compared to implementations of the CORDIC algorithm [14] and series expansion, our implementation of a lookup table with linear interpolation resulted in a better performance on the AURIX microcontroller.

Since additional systematic error compensation adds a significant processing delay, only the very essential global offset compensation is performed. This design decision is mandatory in order to keep the processing time on the target platform low and enable high frame-rates.

2) Pre-processing: Several pre-processing steps are performed to enhance the image representation with the goal to improve the quality of the object detection.

This includes a conversion of the distance values to only their z-component which simplifies further object detection steps. As shown in Equation 2, the new distance values $d_{conv}(x, y)$ are calculated using the horizontal pixel angle β and the vertical pixel angle α. Knowing the field-of-view, the pixel angles α and β are approximated with an angle of 0.5° for every pixel from the center. A distance vector before and after the conversion step is shown in Fig. 5.

$$d_{conv}(x, y) = d(x, y) \cdot cos(\alpha) \cdot cos(\beta) \tag{2}$$

Common neighborhood thresholding according to the method presented in [15] is performed in order to detect and discard erroneous pixels. The processed pixel is only considered for further processing if enough neighboring pixels are located in the same distance range.

Amplitude thresholding is applied to discard non-confident pixels with a low amplitude value. This is done by comparing the values to a reference image of a plain white surface taken at a distance of 1 m. The processing step discards amplitude values $A(x, y)$ smaller than the reference value $A_{ref}(x, y)$ scaled with a constant scaling factor c_{th1} (see Equation 3).

$$A(x, y) < A_{ref} \cdot c_{th1} \tag{3}$$

An additional thresholding operation to avoid phase-wrapping based on the work in [16] is applied. Phase-wrapping occurs when objects exceeding the unambiguous range of the system are detected. Since the amplitude value decreases with the square of the distance, every amplitude value is scaled with the square of its responding distance value $d(x, y)$. If that value is smaller than a certain threshold c_{th2}, the pixel is considered not confidential and is discarded (see Equation 4).

$$A(x, y) \cdot d(x, y)^2 < A_{ref} \cdot c_{th2} \tag{4}$$

Finally, median filtering is performed in order to get a smoother version of the image and to reduce the noise from single pixels.

3) Object detection: To achieve a fast, low-effort object detection, only a certain area of interest (AOI) directly in front of the camera is considered (see Fig. 6). The first step of the object detection is to discard pixels outside the AOI. This processing step is executed by comparing every pixel's distance value to a precomputed limit for that pixel position. For efficiency reasons, these limits are stored in a horizontal and a vertical lookup-table. Additionally, only pixels within the robust measurement range of the camera are considered. Pixels outside that range are discarded as well.

Afterwards, an efficient histogram-segmentation based on the work in [7] is performed. The difference of our approach is that the histogram is smoothened after it is created. Then the first derivative is calculated in order to detect the local minima, resulting in possible segmentation borders. If the histogram values between two local minima meet different criteria (for example the amount of pixels or the relation to neighboring minima), the included pixels are classified as an object. Finally, a list of detected objects is created, containing an area, distance, and position for each element.

4) Decision-making: The final step of the algorithm is to decide whether it is required for the system to take over control of the scaled-vehicle in order to avoid a collision. The decision-making algorithm is triggered whenever a new object list or a new car state is available. If the calculated stopping distance at the current speed is within a certain range to the closest detected object, the algorithm takes control and immediately applies the brakes until the car is fully stopped. Afterwards, the control is returned and the vehicle can be controlled via the RF remote again.

Equation 5 shows the calculation of the stopping distance s_{stop} on the embedded system during runtime. The stopping

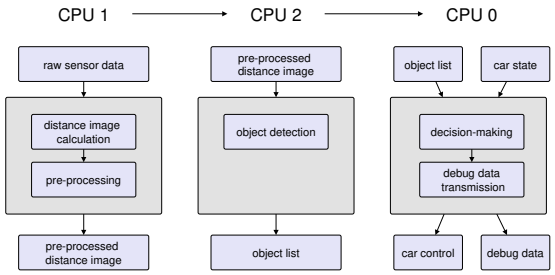


Fig. 7. Partitioning of the processing tasks on the three AURIX CPUs.

distance is the sum of the reaction distance s_{react} and the braking distance s_{brake}. The current speed of the car v is contained in the car state while the reaction time T_{react} is the time since the distance image was captured. The symbol μ represents the friction factor and g the gravitational acceleration.

$$\begin{aligned} s_{stop} &= s_{react} + s_{brake} \\ &= v \cdot T_{react} + \frac{v^2}{2\mu g} \end{aligned} \tag{5}$$

E. Implementation of the Processing Algorithm

The processing algorithm was designed to be equally distributed on the three AURIX CPU cores. Compared to a single-core solution, this reduces the workload per CPU and enables pipelined processing. Fig. 7 shows how the single data processing tasks are partitioned onto the three CPUs.

CPU 1 starts processing after new ToF sensor data was fully transmitted via the PIF. After processing, the pre-processed image is transferred to CPU 2 which performs the object detection and forwards an object-list to CPU 0. This CPU handles the decision-making which is triggered either if a new object list is available or a new car state is received via the CAN bus. Depending on the result of the decision-making task, control signals to takeover are sent to the scaled vehicle via CAN. The data-processing on the CPUs is done on each core's isolated high-speed memory, while data between the CPUs is exchanged via a globally accessible shared memory. This simple method is favored over more complex approaches since the data transfers between the CPUs only make up a minor portion of the overall processing time. The chosen approach grants each core its full local memory and keeps the synchronization overhead low.

Additionally, the CPU 0 also transmits debug information to other platforms (for example a PC) using the Ethernet interface of the AURIX. The actual transferred debug data can be configured, but typically includes intermediate image data and the car state. The lightweight IP (lwIP) stack handles the transmission of UDP packages via Ethernet.

A port of the real-time operating system FreeRTOS is used on the microcontroller to utilize independent tasks, running at defined priorities on each of the three AURIX CPUs. Low-priority tasks (for example sending debug data via Ethernet) are interrupted if high-priority tasks (for example processing received data via CAN) are ready for execution.

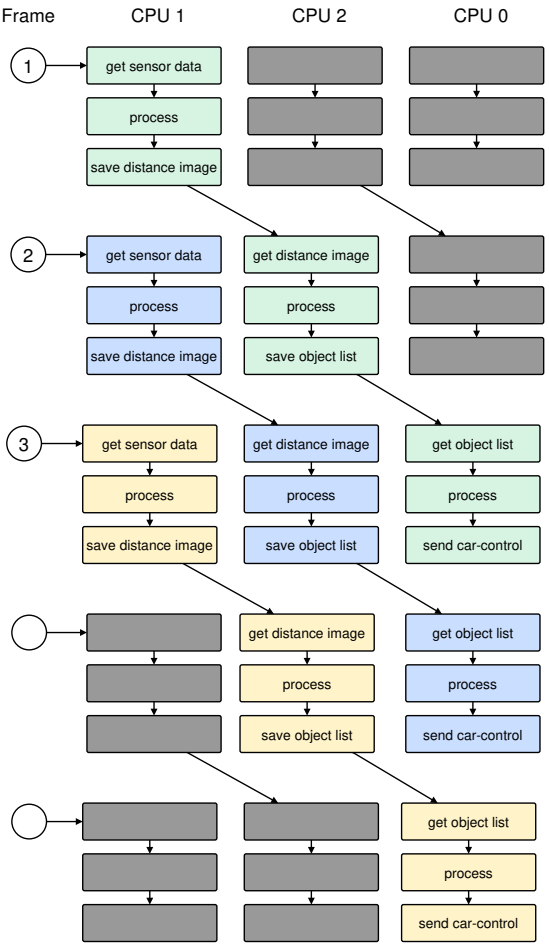
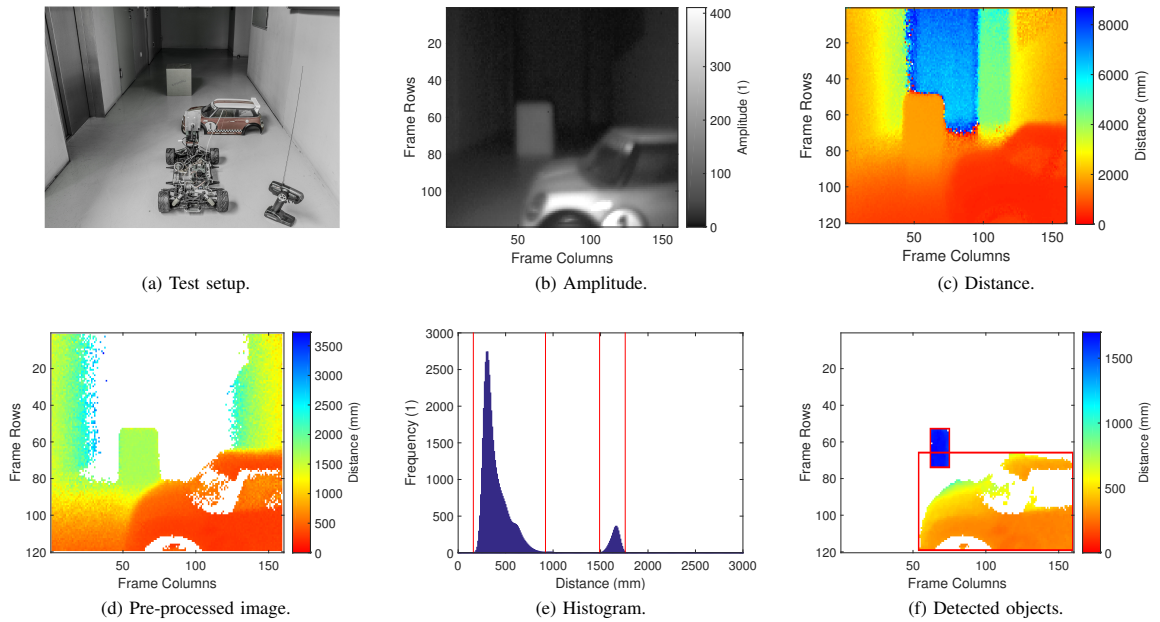


Fig. 8. Pipelining concept implemented on the three AURIX CPUs.

The processing algorithm is distributed on the three cores in order to enable pipelined data processing with the goal to significantly increase the throughput. Fig. 8 illustrates a flow diagram of the pipelining approach. Compared to a single-core solution, the additional data transfers between the CPUs cause the latency to increase. But since the transfers only consume a minor part of the overall execution time, the multi-core approach is beneficial for the targeted application.

F. Latency

The overall system's latency T_{lat} is the time delay from a scene change (for example an appearing object) until the system decides whether it is required to intervene or not. This time shall be minimized and shall outperform the reaction time of an attentive human driver (starting with 0.7 s [17]). It can be calculated as stated in Equation 6.



(a) Test setup. (b) Amplitude. (c) Distance.

(d) Pre-processed image. (e) Histogram. (f) Detected objects.

Fig. 9. Test setup and the data at different processing stages of the algorithm.

$$T_{lat} = T_{pre} + T_{capt} + T_{proc} \tag{6}$$

T_{pre} is the elapsed time from a newly appearing object, until the ToF camera starts to capture the following frame. In the worst case, this can be the entire time between two captured frames. Thus, a higher frame-rate causes this delay to decrease. T_{capt} defines the time it takes the ToF camera to capture a frame and to transfer it to the AURIX. This introduced delay depends on the pixel-clock of the PIF, while the illumination time is constant during runtime. For a pixel-clock of 66.6 MHz and an exposure time of 2 ms, the added delay T_{capt} is about 13 ms. T_{proc} is the time the implemented algorithm takes until the decision on whether to enable emergency braking or not is drawn. Since the algorithm is partitioned on the three AURIX cores, the introduced delay is the sum of the processing time on each CPU core.

IV. Results

An example for intermediate data, captured during the single processing steps, is shown in Fig. 9. As seen in Fig. 9a, the demonstrator was placed in a static test scene containing different obstacles. The ToF camera was configured to run at a frame-rate of 30 FPS, a modulation frequency f_{mod} of 17 MHz, and an exposure time T_{exp} of 2 ms. The distance image calculation task results in an amplitude image (see Fig. 9b) and a distance image (see Fig. 9c). These images were obtained, by performing only the lightweight distance image processing steps described in this paper. Computationally expensive image enhancement techniques are omitted in order to enable high frame-rates. During pre-processing, non-confidential pixels are discarded and basic smoothing is applied resulting in the image shown in Fig. 9d. The generated histogram only considering pixels within the AOI is shown in Fig. 9e. The histogram-segmentation task results in the detection of two objects, with the segmentation borders marked by red lines. Fig. 9f shows the distance image of the remaining pixels within the AOI. The two detected objects are highlighted with red rectangles.

Fig. 10 shows a timing diagram of the three AURIX CPU cores. The processing tasks on the AURIX cores meet their deadlines with the ToF camera running at 30 FPS. The utilization of CPU 1 is almost 90% and has no input-data dependency. Since the execution time of the object detection depends on the input, the utilization of CPU 2 can vary between about 10% and almost 100%. The brief decision-making task on CPU 0 (narrow spikes in timing diagram) is triggered immediately after new input data is available. The longer processing pulses on CPU 0 are related to the low-priority transfer of debug data via Ethernet. This task does not interfere with the high-priority tasks, since it is interrupted by the preemptive RTOS scheduler whenever any tasks of higher priority are ready.

At a frame-rate of 30 FPS and an exposure time of 2 ms, the introduced delay until an object is captured T_{pre} is less than 33 ms. The execution times on CPU 1 and CPU 2 have to be below 33 ms, while the processing time on CPU 0 can be neglected. This results in a maximum processing time T_{proc} of 66 ms. Assuming a capturing delay T_{cap} of 13 ms, the

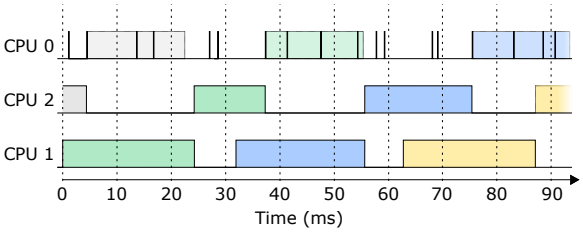


Fig. 10. Timing diagram of the three AURIX CPU cores during operation. Active processing is indicated by the colored pulses on each core.

latency T_{lat} of the system is guaranteed to be less than 112 ms (according to Equation 6). This is significantly faster than the average reaction time of an attentive human driver.

Benefiting from the high frame-rate and the low latency, the simple and efficient algorithm performs well in practice. The automotive microcontroller can take over control of the car via the CAN bus in order to avoid collisions. After the car is in a safe state, the control of the scaled vehicle is returned to the remote control. In order to test the emergency braking algorithm, the scaled vehicle was driven on a dry street with several obstacles.When used in this setting and a maximum speed of 25 km/h, the system detects obstacles early enough to fully stop the vehicle before a collision occurs.

V. Conclusion

A high frame-rate and a low latency are crucial requirements for environmental perception sensors in dynamic applications. This is even more important for sensors with a short range or a low resolution. Considering the range-restrictions of current ToF sensors, this work proposes an efficient method to use a ToF imaging sensor together with a safety-focused automotive microcontroller for close proximity sensing.

We implemented a ToF processing algorithm on the resource constrained state-of-the-art safety controller as part of a mixed-criticality application. A pipelining approach is used on three independent CPU cores to increase the throughput and the frame-rate of the system. To practically demonstrate the functionality of the proposed system, a scaled vehicle featuring a simplified version of emergency braking was deployed. The implemented algorithm enables frame-rates of up to 30 FPS and a maximum latency of 112 ms. Intelligent re-partitioning and excluding the transmission of debug data can further increase the throughput of the system and enable frame-rates of over 50 FPS.

Possible applications of ToF sensors are as successor or as redundant system to ultrasonic sensors. The key benefit of ToF sensors are their fine-grained data, capable to be visually interpreted and the low computational effort to obtain 3D images. In contrast to other range imaging techniques like stereo-vision, ToF cameras enable real-time range imaging on safety-focused embedded systems. We conclude that, automotive applications such as parking, back-driving assistance or close proximity perception could highly profit from ToF sensing data.

Acknowledgments

The authors of this paper would like to thank the Austrian Research Promotion Agency (FFG) and the ARTEMIS Joint Undertaking for funding the projects EMC^2 and the ACTIVE with the project numbers 621429 and 855010.

References

[1] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, March 2001.

[2] N. Druml, G. Fleischmann, C. Heidenreich, A. Leitner, H. Martin, T. Herndl, and G. Holweg, "Time-of-flight 3d imaging for mixed-critical systems," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, July 2015, pp. 1432–1437.

[3] S. Foix, G. Alenya, and C. Torras, "Lock-in Time-of-Flight (ToF) Cameras: A Survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, September 2011.

[4] N. Druml, C. Ehrenhoefer, W. Bell, C. Gailer, H. Plank, T. Herndl, and G. Holweg, "A fast and flexible HW/SW co-processing framework for Time-of-Flight 3D imaging," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, apr 2017, pp. 165–170.

[5] T. Möller, H. Kraft, J. Frey, M. Albrecht, and R. Lange, "Robust 3D Measurement with PMD Sensors," *Range Imaging Day, Zürich*, 2005.

[6] H. Plank, G. Holweg, T. Herndl, and N. Druml, "High performance time-of-flight and color sensor fusion with image-guided depth super resolution," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1213–1218.

[7] X. Wei, S. L. Phung, and A. Bouzerdoum, "Object segmentation and classification using 3-D range camera," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 74–85, 2014.

[8] N. Ziraknejad, P. D. Lawrence, and D. P. Romilly, "The effect of Time-of-Flight camera integration time on vehicle driver head pose tracking accuracy," in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, July 2012, pp. 247–254.

[9] T. Kopinski, S. Geisler, L. C. Caron, A. Gepperth, and U. Handmann, "A real-time applicable 3D gesture recognition system for automobile HMI," *2014 17th IEEE International Conference on Intelligent Transportation Systems (ITSC 2014)*, pp. 2616–2622, 2014.

[10] T. Ringbeck, T. Möller, and B. Hagebeuker, "Multidimensional measurement by using 3-D PMD sensors," *Advances In Radio Science*, vol. 5, pp. 135–146, 2007.

[11] H. Plank, G. Holweg, C. Steger, and N. Druml, "Time-of-flight based optical communication for safety-critical applications in autonomous driving," in *Computer Safety, Reliability, and Security: SAFECOMP 2016 Workshops, ASSURE, DECSoS, SASSUR, and TIPS, Trondheim, Norway, September 20, 2016, Proceedings*. Springer International Publishing, 2016, pp. 183–194.

[12] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Estimating human 3D pose from Time-of-Flight images based on geodesic distances and optical flow," in *2011 IEEE International Conference on Automatic Face Gesture Recognition and Workshops (FG 2011)*, March 2011, pp. 700–706.

[13] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, November 2012, pp. 692–697.

[14] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, September 1959.

[15] Y. Dalbah, S. Rohr, and F. M. Wahl, "Detection of dynamic objects for environment mapping by time-of-flight cameras," in *2014 IEEE International Conference on Image Processing (ICIP)*, October 2014, pp. 971–975.

[16] J. Poppinga and A. Birk, *A Novel Approach to Efficient Error Correction for the SwissRanger Time-of-Flight 3D Camera*. Springer Berlin Heidelberg, 2009, pp. 247–258.

[17] M. Green, ""How Long Does It Take to Stop?" Methodological Analysis of Driver Perception-Brake Times," *Transportation Human Factors*, vol. 2, no. 3, pp. 195–216, 2000.

Next Generation Radar Sensors in Automotive Sensor Fusion Systems

Josef Steinbaeck*, Christian Steger†, Gerald Holweg*, and Norbert Druml*
*Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, gerald.holweg, norbert.druml}@infineon.com
†Graz University of Technology, Graz, Austria, Graz, Austria
steger@tugraz.at

***Abstract*—During the last decades, radar sensors have become an established component in assisted driving systems. Light detection and ranging (LIDAR) sensors are commonly perceived as the key technology to enable fully autonomous driving, but are still expensive. Recent advances in radar technology, enable modern radar sensors to sense the environment in higher detail, thus adding significant value to the environment perception quality of a sensor fusion system. Due to their performance, reliability, and cost-efficiency, modern radar sensors will be a crucial element in future autonomous vehicles.**

In this paper we give an overview of the current state-of-the-art of automotive radar sensors and their potential in future vehicles and data fusion systems. We want to show that modern radar sensors add significant value to automotive environment perception in order to enable fully automated driving. Additionally, challenges arising with the integration of new radar sensors and remaining issues are discussed.

***Index Terms*—radar, automotive sensors, accident prevention, automated driving, autonomous vehicles**

I. Introduction

Many automakers and tech companies are striving to deploy fully automated driving to the roads within the next couple of years. Fully automated vehicles will greatly rely on sensor data to safely navigate through the streets. Today's most valuable environmental perception sensors are vision, radar, LIDAR, and ultrasonic. In contrast to assisted driving, fully automated driving platforms do not act as a backup systems for a human driver. Fully autonomous vehicles have to stay functional in any scenario, what increases the sensor requirements tremendously.

Most prototype vehicles targeting fully automated driving highly rely on a spinning LIDAR sensor mounted on top of the vehicle (e.g. [1]). This single-sensor solution is very popular among prototyping vehicles, since it directly provides a highly accurate point cloud of the environment. The spinning LIDAR provides an out-of-the-box platform to implement higher-level functions for autonomous vehicles (for example path planning or object detection) without worrying about sensor data fusing and processing.

Unfortunately, commonly used LIDAR systems also come with disadvantages like moving mechanical components, reduced performance in difficult light conditions, and the high prices (year 2017 [2]). First prototypes of low-priced solid-state LIDAR systems have already been introduced, and patents were granted (e.g. [3], [4]). However, these products might also come with further limitations (e.g. smaller field-of-view) and performance restrictions compared to expensive spinning LIDAR systems. To sum up, LIDAR alone does not meet the high sensor requirements to be used in future mass-produced fully autonomous vehicles.

TABLE I
Comparison of environment perception sensor capabilities.

Sensor	Radar	LIDAR	Vision
Range	++	+	++
Range resolution	+	++	o
Angular resolution	o	++	+
Works in bad weather	++	o	-
Works in dark	++	++	--
Works in bright	++	+	+
Color/contrast	--	--	++
Radial velocity	++	o	-

In order to mitigate the weaknesses of individual sensors, multiple sensor technologies have to be used in a redundant and diverse way. This comes with the requirement to perform the non-trivial task to fuse the data of multiple inhomogeneous sensors. Table I shows the strengths and weaknesses of todays most popular automotive environmental perception sensors. A more detailed performance comparison of currently available environmental perception sensors is given in [2]. As seen in the table, none of the individual sensors provides sufficient robustness to provide an always working solution. The big strengths of radar sensors are their ability to work in bad weather conditions, as well as the possibility to directly measure the relative angular velocity of the detected targets (Doppler effect).

This work focuses on the recent advances in automotive radar technology. Modern radar sensors can now provide high-resolution environment data, which can be very beneficial for sensor fusion systems. The robust measurement of radar sensors is especially valuable, since it also works in difficult conditions where other sensors (LIDAR, vision) are likely to fail. This work gives an overview of the characteristics of modern radar systems, including the antennas, modulation techniques, sensor data processing and fabrication technologies. To sum up, the contributions of this paper to scientific research are:

- An overview of state-of-the-art automotive radar solutions

and an outlook into the future.

- Possible opportunities for automotive radar sensors in order to gain importance with respect to future automated road vehicles.
- Architecture requirements for future automotive multi-sensor fusion systems.

Section II gives an overview of the state-of-the-art of automotive radar technology. Different approaches of integrating modern radar sensors into sensor fusion systems as well as related safety considerations are presented in section III. The Section IV summarizes the potential of modern radar sensors in the robotic/automotive industry and gives an outlook to future systems.

II. Radar Sensors for Automotive Vehicles

Radar sensors have been around in the automotive industry since a few decades already. Driving assistance systems can be divided into comfort functions and the more demanding safety applications. First systems were mostly used for comfort applications in the premium car segment, like adaptive cruise control (ACC). ACC requires the vehicle to know the position and velocities of the other vehicles in the vicinity. Since radar technology continuously improved, recent radar sensors are also used in safety applications like autonomous emergency braking (AEB). The 2013 Mercedes S-Class was already equipped with seven cameras, six radars and twelve ultrasonic sensors in order to enable advanced driving assistance functions [5], [6]. Since 2014, AEB is part of the European new car assessment program (Euro NCAP) car safety rating system. Thus vehicles with AEB can achieve a better NCAP safety rating. The work published in [7] gives a well-structured overview of the history of radar systems in an automotive context.

Radar sensors have some beneficial characteristics for automotive environment perception. Compared to other environmental perception sensors, radar also works in foggy, dusty, snowy and badly lighted environments [8], [9]. Additionally, the price for mass-produced radar sensors is decreasing, making them feasible for the installation in mid-range cars. Table II shows a comparison of different radar sensors already used in today's vehicles. The sensors are capable of measuring distances to objects more than 200 m away and provide high range accuracy, but they are restricted by their limited angular resolution. Newer generations of automotive radar sensors implement advanced methods to overcome this limitation. Thus next generation of radar sensors will be capable to achieve a significantly higher resolution (azimuth as well es elevation) and will be able to perceive the environment in greater detail.

A. Basic Principle

Radar sensors transmit high-frequency electromagnetic waves and receive the reflection of that wave in order to estimate the position and velocity of present objects. The distance to a target is determined using the temporal delay between the transmitted and a received signal. Due to the Doppler effect, the radial velocity of moving targets can be calculated using the occurring frequency shift of reflected waves.

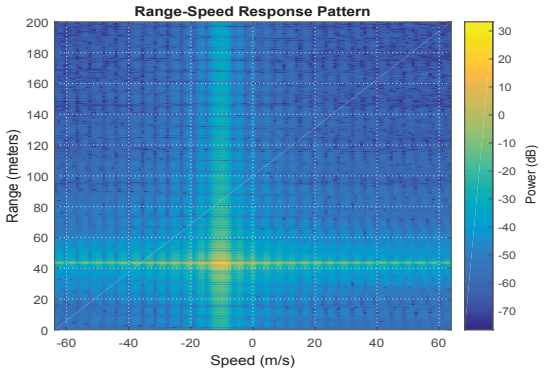


Fig. 1. Radar range-velocity measurement of a target in the range of 43 m and a speed of 10 m/s. Obtained from [13].

With the simplest form, the transmission of a continuous wave (CW), only the speed of a target can be measured utilizing the Doppler effect. Transmitting a frequency modulated continuous wave (FMCW) is one possibility to also determine the distance to a target as well as the velocity. The received signals are digitized and processed by a 2D FFT in order to determine the range-velocity matrix. The output matrix after range-doppler processing of a single target is shown in Fig. 1. Using that output, there exist a number of algorithms to detect obstacles using radar only [10], [11] as well as fusing it with data from other environment perception sensors [12].

According to [14], the range resolution ΔR of an ramp-sequence based FMCW radar system [15] depends on the frequency-sweep bandwidth B, while the velocity resolution Δv depends on the time-duration T of the sweep ramp and the number of ramps N (Equations 1 and 2).

$$\Delta R = \frac{c_0}{2B} \tag{1}$$

$$\Delta v = \frac{\lambda}{2NT} \tag{2}$$

B. Frequency Bands

Initially, the 77 GHz band was planned to be used for long range radar (LRR) applications, while 24 GHz band was planned for high-resolution short-range radar (SRR) and mid-range radar (MRR) [16]. For long range comfort applications like ACC, high precision is not mandatory. Thus, the bandwidth of the 77 GHz band (76-77 GHz) is sufficient and the frequency range is still used for LRR systems.

The situation for SRR in the 24 GHz range is more complex. The high bandwidth of the 24 GHz UWB ranging from 22.0 GHz to 26.625 GHz allows a high resolution. However, this frequency band has a very limited transmission power and suffers from interferences since it is also used by several other radio services (in Europe). Thus, only short range operations

TABLE II
CHARACTERISTICS OF CURRENT AUTOMOTIVE RADAR SENSORS, OBTAINED FROM [2].

Sensor	Frequency (GHz)	Bandwidth (GHz)	Range (m)	Azimuth Angle (°)	Accuracy	Cycle (ms)
Bosch LRR3	77	1	250	±15.0	0.1 m	80
Delphi ESR	77	-	174	±10.0	1.8 m	50
Continental ARS30x	77	1	250	±8.5	1.5 %	66
SMS UMRR Type 40	24	0.25	250	±18.0	2.5 %	79
TRW AC100	24	0.1	150	±8.0	-	-

are possible, like blind-spot detection or parking assistance. The European Union has introduced this frequency band to allow a fast market introduction of early SRR systems, but restricted its usage until January 2018. The 24 GHz industrial, scientific, and medical (ISM) band from 24.05 GHz to 24.25 GHz is allocated permanently and can be used for longer ranges since it allows a higher transmission power. But since the bandwidth of 200 MHz is too low for detecting details, this frequency band is rather used for industrial applications than automotive.

To solve this problem, the European Union has allocated the 79 GHz UWB (77-81 GHz) as successor to the temporary 24 GHz UWB. This frequency band is capable to be used for LRR as well as SRR systems [17]. Due to the high resolution, the 79 GHz band is capable for the usage in active safety applications like collision avoidance. The shorter wavelength compared to the 24 GHz systems allows more compact housing and thus lower-weight systems and easier installation. One disadvantage of higher frequencies in radar systems is a significant signal attenuation caused by the front bumper of the vehicle [17]. However, there already exist approaches to mitigate this effect in real-time [18].

C. Technology

Initial automotive radar systems used discrete circuit elements while later systems consisted of multiple Monolithic Microwave Integrated Circuits (MMIC) based on gallium arsenide (GaAs). Infineon presented an automotive qualified SiGe based radar system in 2008 [19], replacing the higher-priced earlier GaAs solutions. The work presented in [20] shows the circuit design of a multichannel 77 GHz automotive radar transmitter. Nowadays, almost all new systems are based on silicon-germanium (SiGe) and more and more functionality is integrated into single chips. In order to further reduce prices and and increase the integration, BiCMOS technology, the combination of bipolar and complementary metal-oxide-semiconductor (CMOS) technology, is used to efficiently add digital parts into a single chip [21]. Fig. 2 shows an overview on the frequency capabilities of different semiconductor materials. According to [22], there are currently six million radar systems used in vehicles and a rise to 25 million units is predicted. 70% of the existing radar solutions are still manufactured in GaAs, but due to the introduction of SiGe, this is very likely to change in the future.

SiGe bipolar and SiGe BiCMOS processes are currently leading, but the industry is working on manufacturing a

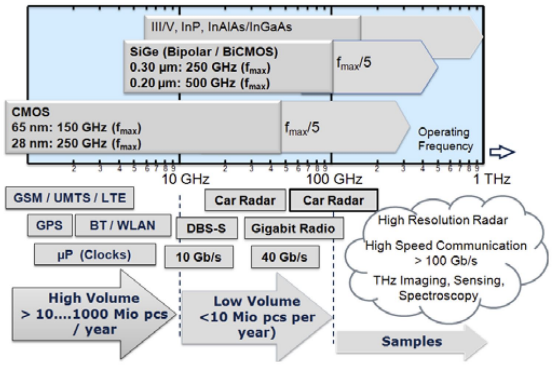


Fig. 2. Favorable semiconductor technologies depending on the operation frequency [20].

79 GHz single-chip radar solution in full CMOS. First prototypes of CMOS transceiver chips have already been presented (e.g. [23]). SiGe transistors have significant advantages compared to CMOS (e.g., noise performance), but CMOS allows a higher integration (enabling, e.g., in-sensor processing) and has the potential to further reduce the price of radar systems. Thus, CMOS is very attractive for future systems that exploit more antennas and thus require higher chip integration. However, developing a CMOS solution requires very advanced circuit design and numerous shortcomings have to be solved in order to make the process feasible. The authors of [24] describe challenges of 77 GHz CMOS radar systems and the work presented in [20] points out the advantages of SiGe compared to CMOS. Since CMOS requires more complex circuitries than SiGe solutions, full CMOS might only reduce the cost for solutions with a relatively large digital part. So, even if a automotive qualified CMOS radar is available to the market, BiCMOS solutions are very likely to co-exist for a certain amount of time.

There has also been great progress regarding the packaging of the radar MMIOs. Embedded wafer level BGA (eWLB) is now the leading packaging technology for radar transceiver chips. More details on packaging technology can be found in [21].

D. Beamforming

Beamforming can be used to narrow the transmit beam to a limited field-of-view (FOV) as well as to detect the direction

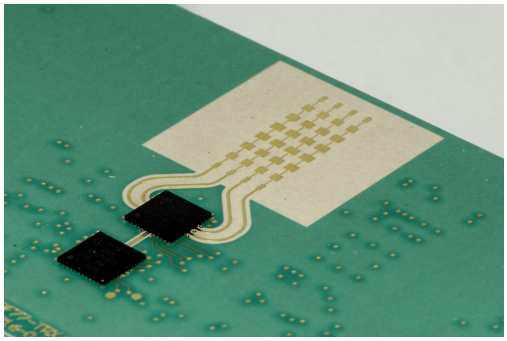


Fig. 3. Four planar TRX (transmit and receive) patch antennas, connected to a MIMO transceiver IC.

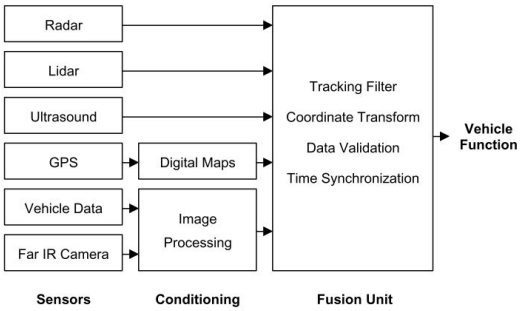


Fig. 4. Sensor fusion architecture, obtained from [35].

of existing targets in order to improve the efficiency of the radar system. There exist different approaches the achieve beamforming:

- Mechanical solutions are the most straightforward way to implement beamforming, but come with the disadvantage of a restricted scanning speed and the wear of mechanical parts [25].
- In early stages of automotive radar, switched multi-beam antennas with a narrow beam-width were used in order to determine radar obstacles [25].
- Optical beamforming methods use lenses to steer the beam into a limited field-of-view (FOV) [26]. The authors of [27] present a lens-based radar system, which can be beneficial for long range radar systems.
- Analog beamforming utilizes a high number of transmitters working similar to phased array antennas. Using this method, it is possible to electronically steer the direction of the transmitted beam. The authors of [28] show an approach to achieve analog beamforming.
- Digital beamforming performs the detection of the direction of arrival (DOA) at the receiver. The angle of the detected objects is typically measured using an array of antennas and comparing the amplitude and phase of the detected signals.

The majority of modern automotive radar systems employ planar patch antennas. An example of a simple planar patch antenna is shown in Fig. 3. These antenna arrays can be used for beamforming, with each channel directly connected to a multiple-input multiple-output (MIMO) transceiver channel. In practice, radar systems also use combinations of the above mentioned beamforming approaches. There exist different approaches to estimate the DOA in order to achieve a high-resolution angle (e.g., [29], [30], [31]). However, only the real-time capable algorithms are feasible for embedded automotive systems.

E. Outlook

The decreasing price of radar sensors enables carmakers the possibility to integrate several radar sensors into future cars. According to [32], the market penetration of 77 GHz and 79 GHz radar sensors is about to rise during the next decade. However, future highly automated driving applications are in need of a higher range-, angle- and velocity-resolution [33]. As stated in [31], the range accuracy is mostly limited by the bandwidth while the angular accuracy is limited by the antenna aperture and the number of receive channels. Due to ongoing advances in circuitry design, using higher frequencies can significantly improve the performance of radar systems. The authors of [34] predict that doubling the frequency can double the angular resolution of an antenna of the same size.

According to [35] it takes several (4-6) years from the development of a new sensor system to the mass-market introduction of this feature. Thus, current advances in the radar technology will take several years until the components reach automotive qualification and appear in new vehicles.

III. Multi-Sensor Data Fusion

A single environment perception sensor is not sufficient to cover the high safety requirements of future vehicles. Thus, multiple sensors of different technologies have to be combined in order to accomplish redundancy and diversity. The sensor fusion system mitigates the weaknesses of the individual sensors and outputs a robust environment model in any scenario. The individual sensors work independently, causing the measurements from the sensors to arrive asynchronously. Additionally, since the sensors are typically mounted on different positions of the vehicle, each sensor has a individual point of view. In order to perform sensor fusion, the sensors require a common understanding of time and space and a standardized data interface [35]. A block diagram of a typical sensor fusion system and its main components is shown in Fig. 4.

A. Sensor Fusion Architectures

There exist three basic types of sensor fusion architectures [36]:

- Low-level sensor fusion
- Feature-level sensor fusion
- High-level sensor fusion

In the low-level sensor fusion approach, the fusion platform processes the raw data from all individual sensors. Disadvantages are the high-data rate and the high complexity to process

the vast amount of raw data. The advantage is the availability of raw measurement data on fusion level, avoiding any loss of information through to pre-processing. Additionally, one centralized processing platform eases time-synchronization and is usually more cost-efficient compared to a distributed system. The authors of [37] show a low-level sensor fusion approach of a far infrared camera, a laser scanner and several radars.

For sensor fusion at feature-level, pre-processing is applied at sensor level to extract certain features. These extracted features are then fused together by the sensor fusion system. This architecture is a trade-off between the two other approaches.

High-level sensors fusion performs the whole processing for each sensor individually. The object lists are then fused together by the sensor fusion system. Advantages are a lower communication overhead and the modularity of the system. An example for high-level sensor fusion is presented in [38], where a fusion of radar and LIDAR data is performed.

B. Data Alignment

In order to perform sensor fusion, the data from the different sensors have to be aligned both temporally and spatially [36]. The temporal alignment of sensor data relies on accurate timestamps added to each measurement of the single sensors. To achieve this, all involved units have to work with a perfectly synchronized global time base.

For spatial alignment, the measurements from the different sensors have to be transferred into a common coordinate system.

C. Safety Considerations

The authors of [39] predict that certain future applications of automated driving will require update-rates of up to 50 Hz. Thus, high-performance processing units as well as high-bandwidth communication have to be integrated into a vehicle's E/E infrastructure.

The high-priority tasks of the safety critical systems are required to guarantee a deterministic response within a certain time-frame. Buses like controller area network (CAN) or Flexray are getting replaced by Ethernet in order to increase the bandwidth. Since standard Ethernet has a non deterministic behavior, it is not applicable to safety critical tasks. Thus, special standards for Ethernet in vehicles are required in order to enable real-time reaction to sensor data. A central clock for the network and known latencies between the nodes are required for deterministic Ethernet [40].

Real-time operating systems have to be used for the processing of these high-priority tasks. Safety critical tasks have to run on safety-certificated hardware and operating systems. Non-safety critical computations can be outsourced to operating systems with multi-tasking abilities running on high-performance hardware.

According to the functional safety standard ISO26262, each function of the autonomous vehicle is assigned an automotive safety integrity rating (ASIL). The highest rating is ASIL D, which is assigned to the most safety critical tasks within a car. In order to guarantee the normal execution of these safety critical tasks, they have to run on redundant safety-certificated hardware and be separated from non-safety functions.

Examining the E/E market for automated driving, semiconductor vendors offer system-on-chip solutions dedicated to sensor fusion applications, which are commonly based on multiple advanced RISC machine (ARM) cores to deploy high computational performance plus dedicated hardware accelerators (e.g., for artificial intelligence) to enable vision functions. Such fusion systems can support at maximum ASIL B or in some cases ASIL C levels. On the other hand, microcontrollers that have been recently developed for functional safety (e.g. transmission control) are based on one or more lockstep cores. These devices feature extended error checks and corrections on internal bus and memories, analog and digital built-in self tests, and several internal monitors (voltage, temperature, clock integrity), enabling them to support ASIL D. These devices, however, are constrained in terms of computational performance. Therefore, major efforts are currently spent by the industry and academia in order to come up with sensor fusion systems integrating number crunching AI solutions and supporting ASIL D at the same time.

The authors of [41] give an in-depth overview of the challenges to develop safety-critical systems for future vehicles.

IV. Conclusion

Although the radar technology has significantly advanced during the last years, radar sensors alone are not sufficient to provide a car with reliable environment information in order to navigate autonomously. Since radar waves can propagate through several non-translucent materials, like thin walls, fog, rain, or snow, they can provide superhuman perception abilities to a vehicle. Therefore, radar sensors are very valuable to provide an environmental perception system with extra information about the cars environment, which are not provided by other sensors. To go even one step further, utilizing the strengths of radar might be essential for automated vehicles in order to meet the high requirements to sensor robustness in future automated vehicles.

However, utilizing the full potential of radar sensors is still a big challenge of today. The obtained data from the sensor is significantly more challenging to interpret than data from visual sensors (such as cameras or LIDAR).

Due to the vast amount of sensor data, high-performance processing units are used in prototype vehicles for automated driving. There is a lack of safety classified, deterministic sensor processing units in order to guarantee operation also in case of faults. Automotive qualified components have to be utilized as safety critical nodes in future vehicles.

Acknowledgments

The authors of this paper would like to thank the Austrian Research Promotion Agency (FFG) for funding the Autonomous CarTo Infrastructure communication mastering adVerse Environments (ACTIVE) project with the number 855010.

REFERENCES

[1] P. Navarro, C. Fernández, R. Borraz, and D. Alonso, "A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data," *Sensors*, vol. 17, no. 1, p. 18, 2016.

[2] F. de Ponte Müller, "Survey on ranging sensors and cooperative techniques for relative positioning of vehicles," *Sensors (Switzerland)*, vol. 17, no. 2, pp. 1–27, 2017.

[3] J. Heck, J. K. Doylend, D. N. Hutchison, H. Rong, and J. B. Sendowski, "Solid state LIDAR circuit with waveguides tunable to separate phase offsets," US Patent US9 575 341 B2, 2017.

[4] K. T. Krastev, H. van Lierop, H. Soemers, R. H. M. Sanders, and A. J. M. Nellissen, "Mems scanning micromirror," US Patent US20 100 296 146 A1, 2010.

[5] W. Fleming, "Forty-Year Review of Automotive Electronics: A Unique Source of Historical Information on Automotive Electronics," *IEEE Vehicular Technology Magazine*, vol. 10, no. 3, pp. 80–90, September 2015.

[6] J. Dickmann, N. Appenrodt, J. Klappstein, H. L. Bloecher, M. Muntzinger, A. Sailer, M. Hahn, and C. Brenk, "Making bertha see even more: Radar contribution," *IEEE Access*, vol. 3, pp. 1233–1247, 2015.

[7] H. H. Meinel, "Evolving automotive radar - from the very beginnings into the future," in *2014 8th European Conference on Antennas and Propagation (EuCAP)*, 2014, pp. 3107–3114.

[8] T. Peynot, J. Underwood, and S. Scheding, "Towards reliable perception for Unmanned Ground Vehicles in challenging conditions," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2009, pp. 1170–1176.

[9] G. Brooker, R. Hennessey, C. Lobsey, M. Bishop, and E. Widzyk-Capehart, "Seeing through dust and water vapor: Millimeter wave radar sensors for mining applications," *Journal of Field Robotics*, vol. 24, no. 7, pp. 527–557, July 2007.

[10] G. Reina, D. Johnson, and J. Underwood, "Radar sensing for intelligent vehicles in urban environments," *Sensors (Switzerland)*, vol. 15, no. 6, pp. 14 661–14 678, 2015.

[11] K. Kaliyaperumal, S. Lakshmanan, and K. Kluge, "An algorithm for detecting roads and obstacles in radar images," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 1, pp. 170–182, 2001.

[12] Shunguang Wu, S. Decker, Peng Chang, T. Camus, and J. Eledath, "Collision Sensing by Stereo Vision and Radar Sensor Fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 606–614, December 2009.

[13] Eugin Hyun, Woojin Oh, and Jong-Hun Lee, "Multi-target detection algorithm for FMCW radar," in *2012 IEEE Radar Conference*, May 2012, pp. 0338–0341.

[14] G. R. Curry, *Radar system performance modeling*. Artech House, 2005.

[15] V. Winkler, "Range Doppler detection for automotive FMCW radars," in *2007 European Radar Conference*, October 2007, pp. 166–169.

[16] M. Schneider, "Automotive Radar Status and Trends," *Microwave Techniques*, pp. 3–6, 2005.

[17] H.-L. Bloecher, A. Sailer, G. Rollmann, and J. Dickmann, "79 GHz UWB automotive short range radar Spectrum allocation and technology trends," *Advances in Radio Science*, vol. 7, pp. 61–65, May 2009.

[18] A. Melzer, F. Starzer, H. Jager, and M. Huemer, "Real-Time Mitigation of Short-Range Leakage in Automotive FMCW Radar Transceivers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 7, pp. 847–851, July 2017.

[19] H. P. Forstner, H. Knapp, H. Jager, E. Kolmhofer, J. Platz, F. Starzer, M. Treml, A. Schinko, G. Birschkus, J. Bock, K. Aufinger, R. Lachner, T. Meister, H. Schafer, D. Lukashevich, S. Boguth, A. Fischer, F. Reininger, L. Maurer, J. Minichshofer, and D. Steinbuch, "A 77GHz 4-channel automotive radar transceiver in SiGe," in *2008 IEEE Radio Frequency Integrated Circuits Symposium*, June 2008, pp. 233–236.

[20] F. Dielacher, M. Tiebout, R. Lachner, H. Knapp, K. Aufinger, and W. Sansen, "SiGe BiCMOS technology and circuits for active safety systems," in *Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*, April 2014, pp. 1–4.

[21] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and C. Waldschmidt, "Millimeter-Wave Technology for Automotive Radar Sensors in the 77 GHz Frequency Band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 845–860, March 2012.

[22] R. Lachner, "Industrialization of mmWave SiGe technologies: Status, future requirements and challenges," in *2013 IEEE 13th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, January 2013, pp. 105–107.

[23] Y.-A. Li, M.-H. Hung, S.-J. Huang, and J. Lee, "A fully integrated 77GHz FMCW radar system in 65nm CMOS," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, February 2010, pp. 216–217.

[24] J. P. John, J. Kirchgessner, R. Ma, D. Morgan, I. To, and V. P. Trivedi, "Si-Based Technologies for mmWave Automotive Radar," in *2016 IEEE Compound Semiconductor Integrated Circuit Symposium (CSICS)*, October 2016, pp. 1–4.

[25] W. Menzel and A. Moebius, "Antenna Concepts for Millimeter-Wave Automotive Radar Sensors," *Proceedings of the IEEE*, vol. 100, no. 7, pp. 2372–2379, July 2012.

[26] T. Binzer, M. Klar, and V. GroB, "Development of 77 GHz Radar Lens Antennas for Automotive Applications Based on Given Requirements," in *2007 2nd International ITG Conference on Antennas*, March 2007, pp. 205–209.

[27] S. Lutz, C. Erhart, T. Walter, and R. Weigel, "8 channel MIMO long range radar concept for angular estimation in multi target scenarios," in *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, April 2015, pp. 1–4.

[28] C. Pfeffer, R. Feger, C. Wagner, and A. Stelzer, "FMCW MIMO Radar System for Frequency-Division Multiple TX-Beamforming," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 12, pp. 4262–4274, December 2013.

[29] P. Wenig, M. Schoor, O. Gunther, B. Yang, and R. Weigel, "System Design of a 77 GHz Automotive Radar Sensor with Superresolution DOA Estimation," in *2007 International Symposium on Signals, Systems and Electronics*, July 2007, pp. 537–540.

[30] C. Fischer, F. Ruf, H. L. Bloecher, J. Dickmann, and W. Menze, "Evaluation of different super-resolution techniques for automotive applications," in *IET International Conference on Radar Systems (Radar 2012)*. Institution of Engineering and Technology, 2012, pp. 1–6.

[31] F. Engels, P. Heidenreich, A. M. Zoubir, F. K. Jondral, and M. Wintermantel, "Advances in Automotive Radar: A framework on computationally efficient high-resolution frequency estimation," *IEEE Signal Processing Magazine*, vol. 34, no. 2, pp. 36–46, March 2017.

[32] M. Kunert, H. Meinel, C. Fischer, and M. Ahrholdt, "Report on interference density increase by market penetration forecast, MOre Safety for All by Radar Interference Mitigation (MOSARIM)," European Commission FP7 Project, Tech. Rep. 0, 2010.

[33] J. Dickmann, J. Klappstein, M. Hahn, M. Muntzinger, N. Appenrodt, C. Brenk, and A. Sailer, "Present research activities and future requirements on automotive radar from a car manufacturer's point of view," in *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, April 2015, pp. 1–4.

[34] C. Waldschmidt and H. Meinel, "Future trends and directions in radar concerning the application for autonomous driving," in *2014 44th European Microwave Conference*, October 2014, pp. 1719–1722.

[35] R. H. Rasshofer and K. Gresser, "Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions," *Advances in Radio Science*, vol. 3, pp. 205–209, May 2005.

[36] M. Aeberhard and N. Kaempchen, "High-level sensor data fusion architecture for vehicle surround environment perception," in *Proc. 8th Int. Workshop Intell. Transp*, 2011.

[37] L. Walchshäusl, R. Lindl, K. Vogel, and T. Tatschke, "Detection of Road Users in Fused Sensor Data Streams for Collision Mitigation," in *Advanced Microsystems for Automotive Applications 2006*, Berlin/Heidelberg, 2006, pp. 53–65.

[38] D. Gohring, M. Wang, M. Schnurmacher, and T. Ganjineh, "Radar/Lidar sensor fusion for car-following on highways," in *The 5th International Conference on Automation, Robotics and Applications*, December 2011, pp. 407–412.

[39] D. Caveney, "Cooperative Vehicular Safety Applications," *IEEE Control Systems Magazine*, vol. 30, no. 4, pp. 38–53, August 2010.

[40] A. Patterson, "Die Evolution von Embedded-Architekturen für die nächste Generation von Fahrzeugen," *ATZelektronik*, vol. 12, no. 2, pp. 26–33, April 2017.

[41] G. Macher, M. Stolz, E. Armengaud, and C. Kreiner, "Filling the gap between automotive systems, safety, and software engineering," *e & i Elektrotechnik und Informationstechnik*, vol. 132, no. 3, pp. 142–148, June 2015.

Localization and Context Determination for Cyber-physical Systems based on 3D Imaging

Hannes Plank
Infineon Technologies Austria AG, Austria
Josef Steinbaeck
Infineon Technologies Austria AG, Austria
Norbert Druml
Infineon Technologies Austria AG, Austria
Christian Steger
Graz, University of Technology, Austria
Gerald Holweg
Infineon Technologies Austria AG, Austria

ABSTRACT

In recent years, consumer electronics are becoming increasingly location and context-aware. Novel applications, such as augmented and virtual reality, have high demands in precision, latency and update rate in their tracking solutions.
3D imaging systems have seen a rapid development in the past years. By enabling a manifold of systems to become location and context-aware, 3D imaging has the potential to become a part of everyone's daily life. In this chapter, we discuss 3D imaging technologies and their applications in localization, tracking and 3D context determination. Current technologies and key concepts are depicted and open issues are investigated.
The novel concept of location-aware optical communication based on Time-of-Flight depth sensors is introduced. This communication method might close the gap between high performance tracking and localization.
The chapter finally provides an outlook on future concepts and work-in progress technologies, which might introduce a new set of paradigms for location-aware cyber-physical systems in the Internet of Things.

Keywords: 3D imaging, depth sensing, internet-of-things, location-awareness, optical communication, localization, tracking, augmented reality, tracking

INTRODUCTION

3D imaging technologies have seen rapid developments in the past years. The introduction of the Microsoft Kinect depth sensor to the consumer market in 2010 triggered a massive research interest and effort. In 2016, the first mass-produced smartphone appeared, featuring depth sensing based on Time-of-Flight. The availability of such ubiquitous and miniaturized depth sensing solutions can tremendously help any kind of electronic device to sense and understand its environment.
A crucial part of operation for certain devices is localization. While depth sensors provide geometric information about the immediate surrounding, determining location and orientation within a certain coordinate system is a challenge of its own. This chapter explores the opportunities depth sensing systems provide to localization. A focus is set on applications in fields such as consumer electronics, internet of things and autonomous robots.

Localization and tracking of electronic devices has a long history and has seen the use of a variety of different principles. This work focuses on the fields of high performance localization based on optical and sensor fusion solutions. Localization principles in general can be categorized into passive, guided and cooperative solutions.
A passive system is able to determine its position in a local or global coordinate system without external help. An increasing number of applications also require information about the orientation of the device. A combination of position and rotation sensing is often referred to as pose determination. A pose has of six degrees of freedom (dof) and completely describes the static position and orientation of an entity in 3D space. Each axle in 3D space presents one degree of freedom for the position and one degree for rotation around the axle. Passive 6-dof localization is often used in computer vision based positioning systems, where features are matched with prerecorded databases. Early examples are cruise missiles using terrain contours for navigation.
A well-known example for guided localization is GPS, where devices use the aid of satellites to determine their position. Cooperative localization solutions use a communication channel, which is often used for active identification and position forwarding. Optical tracking, using image sensors and active markers is an example for cooperative tracking. In such system, an external base-station with an image sensor can sense a tracked device equipped with active LED markers, and has the ability to toggle the LEDs for identification. Another example are beacon based systems, where active beacons forward information about their location.

When classifying the location-awareness of cyber-physical systems, it is important to distinguish between localization and tracking. While these terms are sometimes used ambiguously, tracking is commonly used in a relative context, where the registration of movements is important. Tracking the pose of a device does not always lead to the determination of a position within a meaningful coordinate system. However relative position and rotation changes can be detected. For certain applications, this is sufficient and no broader localization is required. Examples for such systems are instruments measuring rotations, such as gyroscopes or compasses, some 3D scanning solutions or human interface devices.
Localization is often associated with position determination without focus on detecting relative pose changes. A combination of tracking and localization is used in a lot of location-aware systems and leads to localization at a high update rate. Tracking and localization are often performed by different sensors, because localization solutions often lack of the desired accuracy to track relative pose changes.
While localization can provide the position and orientation within a greater context, tracking sensors provide the accuracy and update-rate required for the application.
A great example of sensor fusion for localization and tracking is Wikitude (2016). This smartphone application provides augmented reality on smartphones. It annotates the video stream of the internal camera with meaningful information about the environment and displays it on the screen. GPS or WIFI is used for positioning. The absolute orientation is determined by the gravity and compass sensors. The gyro sensors are used to track movements to enable a high update rate of the rotation. This enables to robustly attach information to certain landmarks and directions in the smartphone video stream.

Arising technologies such as virtual and augmented reality, autonomous driving and indoor localization demand precise pose determination at very high update rates. These demands are tackled in state-of-the-art systems with a sensor fusion approach, combining optical and inertial sensors. Optical sensors are used in the form of 2D and 3D cameras, LiDAR (light detection and ranging) and optical triangulation.
Data fusion with inertial sensors can compensate the flaws of optical sensors. Most optical positioning sensors require a line of sight connection and sometimes feature a slow update rate and too much latency as well as visual artifacts such as motion blur. Inertial sensors are commonly miniaturized, using MEMS technology and feature high update rates. These sensors are already well-established in mobile devices. Inertial sensors however base their measurements on differential movements and rotations. In order to measure absolute movements and rotation, the measurements need to be integrated. This introduces an integration error which introduces drift (Woodman, 2007). This drift can be compensated by fusing these measurements with non drifting data, as optical systems can produce.

Using cameras for localization is traditionally accomplished by sophisticated computer vision methods, which are often solely based on 2D images. A common approach is simultaneous localization and mapping (SLAM), where a 3D representation of the environment is created during localization. Depth sensors are capable to improve the performance of such monocular SLAM systems (Steinbrücker, Sturm, & Cremers, 2011). A prominent example is Google Tango, which uses a number of sensors, including a Time-of-Flight depth camera for SLAM based localization on mobile devices. Depth sensing systems are devices capable to directly gather geometric information about the immediate environment. A measurement typically consists of a coordinate in three dimensions, usually relative to the depth sensor itself. In a depth camera, every pixel produces such measurement. If the shutter of the camera triggers all pixels at the same time, every measurement is captured simultaneously. Since all available depth imaging systems are limited in range and show systematic measurement errors, some systems also attach a confidence measure to each measurement.

One of the reasons that the vast research effort on optical localization systems are based on 2D cameras, is that the field of depth sensors is much younger than 2D sensors and they are not yet part of common vision systems. This might change in the smartphone and tablet domain, since as of 2016 the first smartphone featuring a Time-of-Flight depth camera appeared on the consumer market. Measuring depth based on the Time-of-Flight principle is the most miniaturized solution available and has the flexibility to be used in a manifold of applications.

DEPTH IMAGING FOR 6-DOF LOCALIZATION

This section introduces current depth imaging solutions with focus on Time-of-Flight technology. Depth sensors alone are usually not directly associated with localization, although they provide 3D-context awareness of the immediate surrounding. A sensor fusion approach, incorporating depth sensors however offers advantages in SLAM based systems (Steinbrücker, Sturm, & Cremers, 2011), and can improve tracking precision in general. In this work, we focus on depth sensing based on Time-of-Flight, since it is the most miniaturized solution, and is the only depth sensor, which can be found in mass produced smartphones. We also present a concept in this chapter, where Time-of-Flight sensors are directly used for location-aware optical communication, closing the gap between depth sensing and localization.

The principle of Time-of-Flight depth sensing

Time-of-Flight imaging is based on measuring how long light takes to travel from the sensing system to the scene and back to a photosensitive sensor. These systems can be distinguished by their operating principle, as illustrated in Fig. 1. Direct Time-of-Flight systems emit short light pulses and measure the time between emission and reception. Each pulse corresponds with one measurement. A prominent example is LiDAR, where bundled light pulses are emitted by a laser and detected by photodiodes. The angle of these pulses is modulated to receive a spatial image of the surroundings. Indirect ToF imaging sensors relay on a continuous wave approach.

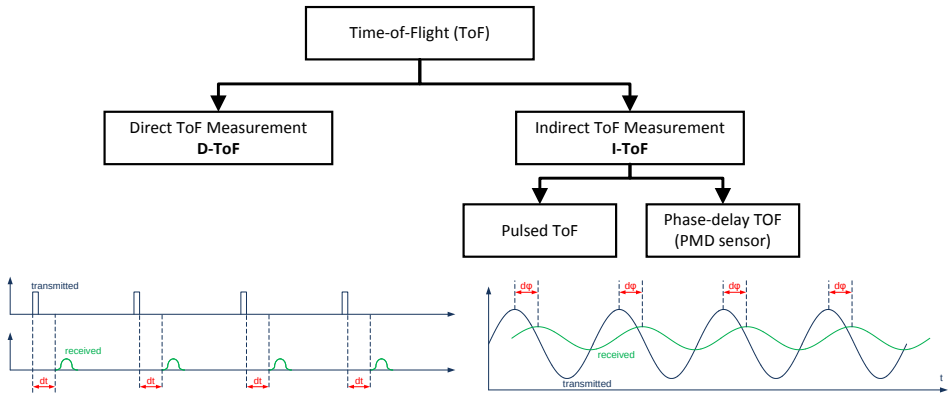


Fig. 1 The different principles of Time-of-Flight based depth measuring

The operating principle of this indirect approach is illustrated in Fig. 2 and works by emitting pulsed infrared light at a wide radiation angle. An image sensor receives the reflected light and is able to measure the phase-shift between incoming and outgoing signal, which is proportional to the distance. The emitted light pulses usually have a frequency of up to 100 MHz. They originate from an active illumination unit, which typically consists of an infrared LED or vertical cavity surface emitting laser (VCSEL). The pulses travel to the sensed object and back to the image sensor. The lens projects the light onto a ToF image sensor.

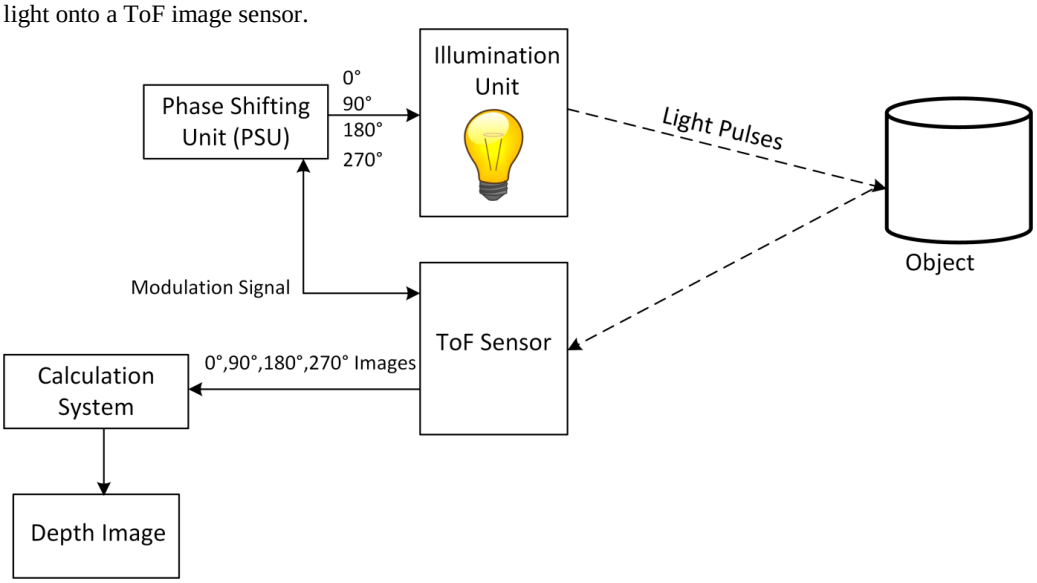


Fig. 2 The working principle of continuous wave indirect Time-of-Flight 3D imaging

Each pixel of the ToF sensor contains a photonic mixer device, which produces a signal corresponding to the phase-shift between the outgoing and incoming light (Tobias, 2005). The incoming photons generate charge-hole pairs in the silicon, which are moved into either of two charge buckets A and B. This is decided by the logical state of the modulation signal. This signal is generated on the ToF sensor and is also used to emit light pulses. In order to control the PMD devices, the signal is directly supplied to each pixel. The charge difference between bucket A and B is the output of the ToF sensor and is related to the phase-shift of the incoming and outgoing signals.
This phase output value however also depends on the amount of incoming light. Reflection properties of the sensed material and as well the distance influence this value. In order to determine the exact phase-shift, the most common way is to conduct four or more different measurements. In these

measurements, the phase-shift offset between the outgoing light and the modulation signal is changed, producing four or more different phase images.
A well-established procedure is to take four measurements I with four equidistant phase offsets (e.g. 0°, 90°, 180°, 270°) and calculate the phase-offset by the following relation:

$$p = \text{atan}(\frac{I_0 - I_{180}}{I_{90} - I_{270}})$$

Since p is proportional to the phase-shift of a pulsed light signal, the phase values wrap and start again at zero, if the distance is too long. Time-of-Flight sensors are capable to change their modulation frequency in order to produce another different set of four phase images to unwrap these ambiguous phases. The final measurement for a depth image consists then of eight phase-images, which results in a larger range, while maintaining precision. The tradeoff is a decreased frame-rate and potential motion artifacts. The eight phase mode is commonly used in SLAM applications, which prefer depth quality over frame-rate.

The drawback of Time-of-Flight sensors is limited resolution of the depth image, since each pixel contains the circuit of the photonic mixer device. The limited photo-sensitive area on the silicon is compensated partly by using micro-lenses directly applied above the silicon to focus light to the photosensitive areas of a pixel. Since continuous wave Time-of-Flight sensing is the most miniaturized depth sensing system available, increasing the sensor size to enhance resolution or photosensitivity is often not feasible. Another drawback is the sensitivity to incoming background light. Despite countermeasures such as background illumination suppression (BSI) and infrared filters, very bright light sources such as reflected sunlight can reduce the signal to noise ratio, but do not directly influence the measurement.

Alternative Depth Imaging Approaches

Each available depth imaging system has its own trade-offs and no system is yet predominant. Compared to Time-of-Flight, all image-sensor based approaches need a certain size to conduct measurements based on triangulation.
Stereo depth sensors usually consist of two cameras which are mounted in parallel with a certain distance. This baseline between these cameras is necessary, because depth is measured by the pixel offset between features of two images. A short baseline impairs depth quality and range. However progression in research and increased sensor resolution made rather small baselines feasible, like they are found in form of dual camera systems in smartphones. Stereo cameras usually produce sparse depth measurements, since it is not possible to measure feature disparities of homogenous surfaces. The depth image's x/y resolution however is superior to most other depth sensing principles, because high resolution image sensors can be utilized for stereo. A sensor fusion approach, combining ToF and stereo depth sensors (Zhu, Wang, Yang, Davis, & Pan, 2011), is a promising solution, when high quality depth images are required.

A variation of stereo sensors is structured light (D. Scharstein, 2003). Structured light utilizes a projector in order to project a pattern onto the scene. An infrared camera senses the projected pattern. Since the projected pattern is predefined, the operating principle is similar to a stereo camera setup, as both systems are based on extracting the feature disparity caused by the distance to the camera. The famous Microsoft Kinect sensor is based on this technology. Unlike stereo, dense depth images cannot be gathered, as the projected pattern is also reflected on homogenous surfaces.
Due to the active illumination principle, structured light based systems can also operate in low light conditions. The drawback of this system is that the active illumination requires more energy, like ToF, and the system is also impaired by ambient light. Unlike ToF, such systems also require a certain distance between projector and camera.

Depth imaging based on LiDAR is most commonly used in automotive applications. Unlike image sensor based approaches, the angular variation of LiDAR measurements does not stem from optical

projection onto an imaging sensor, but from mechanical scanning devices. This mitigates multi-path interference, and allows the usage of highly sensitive photo elements, such as single photon avalanche diodes. This increases the effective range at the cost of more complex sensing systems.

Direct Localization with Depth Sensors

Depth imaging sensors currently are not commonly used in the field of localization of electronic devices. One of the reasons is that most of the research on image based localization is focused on the far more established and ubiquitous color cameras. 3D data does not have as much variation and distinctive features as 2D images. Geometry is more generic and repetitive than reflected light. Due to the measurement principles, 3D imaging also suffers from resolution and limited range.
Geometry however is more consistent, since it is not influenced by different illumination. With depth sensors it is possible build dense 3D models of the environment, as demonstrated by Newcombe et al. (2011) in their Kinect Fusion approach. Such 3D models can be used for re-localization and mapping the environment for systems such as autonomous robots. Another application for such high-quality depth maps is augmented reality. Due to the integration of depth data to a dense model, it is possible to embed virtual objects, using the high quality depth data.
Biswas and Veloso (2012) present an indoor navigation system based on depth information using a plane filtered pointcloud and Monte Carlo localization. Since solely depth sensors are used, the robots can also dynamically avoid obstacles, which are not in the reference map.
While these approaches might work in a smaller context, with distinctive geometry, sensor fusion approaches are more favorable for localization. Later in this chapter, we explore the concept of using Time-of-Flight depth sensors in a novel way to establish location-aware optical communication links to embedded devices. If these devices forward positional information, highly accurate cooperative localization is possible by solely using miniaturized depth sensors.

Depth and Color Sensor Fusion

Depth and color cameras can be combined to create a unified RGB-D sensor, capable to capture images containing both depth and color information. When a SLAM or visual localization system is provided with depth information, the system can either build better 3D maps, but also benefit in robustness from the immediately available distance data (Steinbrücker et al., 2011). The process of 3D and 2D data fusion can also improve depth image resolution, if the required processing power is available. This is due to the principle that depth edges often correlate with edges in color and intensity images. Color edges without any depth discontinuity can be simply ignored, when the lower resolution depth image does not show any variation. Since depth imaging systems usually lack high resolution, color edge information can be used to interpolate and upscale depth images in a meaningful way.

Research has produced a large number of image-guided depth upscaling algorithms (Chetverikov, Eichhardt, & Jankó, 2015). The input for most methods is a high resolution color image and sparse depth measurements which are mapped into the color image space. This sparse RGB-D image is produced by 3D measurements which are mapped to the 2D image. The requirement for such mapping is knowledge about the intrinsic camera parameters of both cameras, which include distortion coefficients, focal length and pixel offsets of the optical centers. These parameters are usually derived by checkerboard calibration, which is also possible for Time-of-Flight depth cameras, since they are able to produce a grey-scale intensity image. The extrinsic parameters of a dual camera system are in this case a translation vector T and a rotation matrix R, which describe the transformation from the depth camera's coordinate system to the color camera. These parameters can also be gathered by capturing images of a checkerboard pattern, and using a toolchain, such as Camera Calibration toolbox (Bouguet, 2016). If intrinsic and extrinsic camera parameters are known, depth measurements with depth $d_{i,j}$ and pixel position $x_{i,j}$ can be mapped to 3D space of the color camera, using the pseudoinverse P of the intrinsic depth camera matrix:

$$X_{i,j} = T + Rd_{i,j}\frac{Px_{i,j}}{\left\|Px_{i,j}\right\|}$$

The 3D measurements $X_{i,j}$ can be projected to 2D image space coordinates $v_{i,j}$ by multiplication with the intrinsic matrix I of the color camera.

$$v_{i,j} = I\, X_{i,j}$$

Methods to interpolate these projected depth images involve various principles, such as energy minimization (Ferstl, Reinbacher, Ranftl, Ruether, & Bischof, 2013), graph based methods (Dai, Zhang, Mei, & Zhang, 2015) or edge aware interpolation (Plank, Holweg, Herndl, & Druml, 2016). Most methods are designed with focus on depth image quality and not for efficiency. In location-aware cyber-physical systems, interactive framerates are desired. When the depth images are used to build 3D maps in a SLAM system, several depth images per second are desired. Methods to create high resolution depth images on restricted hardware are relatively rare. While there are approaches, which emphasize on low computational complexity (Dai et al., 2015), there are not many implementations which can be executed on parallel processing systems such as GPUs. The joint bilateral filter, developed by Kopf et al. (Kopf, Cohen, & Lischinski, 2007) can be executed in parallel and works by weighting image filter kernels on color similarity. This works well with relatively small upscaling factors. If however more than just a few pixels between depth values need to be interpolated, depth values influence pixels despite edges between the interpolated pixel and the original depth value. We therefore developed an edge-aware interpolation approach which is optimized for GPUs on mobile devices (Plank, Holweg, Herndl, & Druml, 2016). In this approach, sparse depth values are mapped to high-resolution color images. Each depth value propagates its influence among circular paths to the surrounding pixels. If edges are crossed, the influence drastically decays. If no edges are crossed, the influence is evenly distributed, suppressing sensor noise. Our prototype implementation is capable of producing 13 frames per second, when executed on GPUs on mobile devices.

Beside the capability of SLAM systems to create better geometric models of the environment, the availability of high resolution 1-1 mapped depth and color images, enables better context awareness, since combined depth and color data benefits 2D object detection algorithms (Yu & Zhao, 2012).

A rather unexplored issue is synchronization among camera systems. Most academic work assumes that color and depth sensors operate in synchronous mode, since the technical solution to a synchronous camera system seems trivial. In practical applications however, synchronization is often not feasible as it usually requires tight cooperation across multiple hardware vendors. The vast majority of image sensors are developed to be integrated into monocular systems, not offering any option for hardware synchronization. For RGB-D SLAM systems however, synchronization can be avoided by using additional sensors. Such system is usually in motion while the environment remains static. If color and depth cameras gather information at different times, this can be corrected by using motion tracking data from an inertial sensor. Inertial sensors can operate at a high update-rate. If all measurements, including the depth and color images are accurately timestamped, the actual pose difference between depth image and color image can be calculated by transforming the depth data by the relative pose derived by the motion tracking system. By using physical models of the tracking system, the relative pose can be even more refined.
Ovren et al. (2013) introduce this approach in their attempt to correct the effect of rolling shutter image sensors. A rolling shutter is caused by the pixel readout process, and means, that not all color pixels are captured at the same time. Inertial based depth image alignment is only possible with static scenes, because they are only able to compensate the motion of the camera system.

LOCALIZATION WITH 2D IMAGING SENSORS

Feature Based Localization

Color information alone can be directly used for localization by matching visual input data against databases (Vedaldi & Soatto, 2008). This works by finding features, which are regions or points in input images which are significant and distinctive. Feature descriptors are an abstract representation of these regions with the goal of being able comparable to other features, while being resilient against pose and intensity variations. These feature descriptions can be stored in a visual dictionary,

associating these features with localization information. These databases are either generated systematically via 3D scanning systems, or using topologic information. If these features are recognized by a vision system, the pose of the system can be calculated by triangulation. This kind of visual localization can also be used to initialize a visual SLAM system (Lepetit, Arth, Pirchheim, Ventura, & Schmalstieg, 2015). SLAM can provide more accurate localization, by creating a 3D data representation of the sensed environment.

Visible Light Localization

The motivation of visible light localization (VLC) is caused by the increasingly ubiquitous LED lights. Due to the fast switching speed, it is possible to transmit information from repurposed existing illumination systems. With appropriate modulation, it is possible to use lights for communication without perceivable flickering. Visible light based localization has different applications with different demands in precision. Current products, such as Qualcomm Lumicast (Jovicic, 2016), repurpose image sensors for visible light localization. The distinction between vision based methods with active targets and visible light localization is that base-stations transmit information to help with localization. This can be either IDs or positional information of the base station.
Do and Yoo (2016) provide an extensive survey on methods and implementations of visible light based positioning systems. Such systems usually consist of base-stations, which emit encoded light to electronic devices, equipped with photo-detection sensors. The base stations are either re-used light sources, such as traffic lights or lamps, or dedicated infrared beacons. The simplest solution is based on proximity detection. Such systems can be implemented with just a photodiode as receiver. It is however only possible to detect the base-station itself, so only a very coarse localization with an uncertainty of several meters (Campo-Jimenez, Martin Perandones, & Lopez-Hernandez, 2013) is possible.
Finger printing based methods can achieve more precise positioning, but require pre-recorded maps for localization. Time difference of arrival (TDOA) is another method, which works by receiving light signals from multiple base-stations. For 3D localization, at least three base-stations need to be in the direct field of view. The tracked device directly measures the distance to the base-station in this method. This is accomplished by measuring the time it takes the light pulses of each base station to travel to the device. The position is then determined by trilateration. Such localization systems require a good synchronization of the base stations and the localized device. A single photodiode can be used to receive the signals, and a method to separate the received signals of the base-stations has to be employed. This is possible by using time or frequency division multiplexing (Liu, et al., 2014). TDOA localization is not limited to the optical domain. It is possible to also use radio or sound waves, however multi-path effects need to be considered. Image sensors might be capable to measure time-differences, and also offer light-source separation due to the projection via lenses. The position on the pixel can be used to detect the angles between the base-stations. Due to the pixel readout process, image sensors usually cannot be sampled at the required rates. It either requires dedicated image sensors, featuring customized electronic shutter units or direct pixel readout. Dynamic vision sensors (Censi, Strubel, Brandli, Delbruck, & Scaramuzza, 2013) are a promising development, and might be able to conduct such measurements. An active vision sensor does not produce images, but events which describe local pixel value changes. The difficult synchronization between device and base-stations might deem such image sensor based approaches unfeasible, because it is also possible to determine the position on triangulation alone. Time-of-Flight 3D imaging sensors might be capable to support TDOA based localization, but to our knowledge, this has not yet been investigated and is subject for future research.
Another method to determine the distance between optical receivers and LEDs is to measure the received signal strength (RSS). Calculating the distance to light sources is based on modelling the light source and its propagation path to photo sensors. After calibration, distances can be associated with the output of optical receivers. When receiving the signal from multiple base-stations, the signal differences between the base-stations can be used to calculate the distance. This mitigates the influence of background illumination. RSS based positioning has the potential to be simply implemented and widely adopted, since no synchronization is necessary. The problem however is that the received signal strength depends on the orientation of the photo detector relative to the LEDs as

well. The light strength also depends on the orientation of the LEDs, since light is radiated inhomogeneously. It is however possible to combine RSS with angle based localization methods (Mauro Biagi, 2015).
Localization based on triangulation requires systems, which are able to measure the angle of arrival (AOA). This can be either accomplished by an array of photodiodes (Lee & Jung, 2012) or by using image sensors. AOA systems are in general more complicated, but do not require synchronization. With the help of a 3D camera, it is possible to localize the relative positions of the base stations, and combine trilateration and triangulation in order to improve the localization accuracy. It is also possible to avoid determining the positions of the base-stations beforehand. If only relative movements need to be detected, the base stations can be supplied with their relative locations from the 3D camera system via optical communication. In the next section, we present our OptiSec3D approach, which enables these concepts by combining Time-of-Flight depth sensing with optical communication.

LOCATION-AWARE OPTICAL COMMUNICATION BASED ON TIME-OF-FLIGHT SENSORS

The operating principle of Time-of-Flight depth sensors requires an image sensor, capable of demodulating phase differences of pulsed light. In this section, we present our effort to create a novel location-aware optical communication system. We further go into detail, how it might benefit future localization and tracking systems in the fields of IoT and cyber physical systems.

The most significant feature of image sensor based optical communication is the directional awareness of the communication partner. If depth imaging sensors are used for optical communication, it is even possible to track communication partners in 3D. While there exist a manifold of image sensor based optical communication systems, Time-of-Flight sensors have not yet been widely explored for optical communication.
A first attempt was made by Yuan et al. (2014), who establish a one-way communication link between a Time-of-Flight camera and an array of modulated LED lights. The sending device avoids the required synchronization by recovering the Time-of-Flight sensor's modulation signal with a photodiode. The emitting LEDs are supplied with a phase-modulated modulation signal and manipulate depth measurements of the ToF sensor. These depth measurements are analyzed and the received information is extracted. Since the Time-of-Flight sensor is operated in normal depth sensing mode at relatively low frame-rates in this approach, multiple LEDs are used to transmit information in parallel. Such multiple input approaches are limited in range, since the pixel array cannot resolve individual LEDs when a certain distance is exceeded.
If a system however is capable of configuring and controlling Time-of-Flight 3D imaging systems with a direct connection and a real-time system with low level configuration access, optical communication parameters can be changed to increase readout speeds and it is also possible to use just single modulated LEDs to send information. In our OptiSec3D approach, we utilize Time-of-Flight sensors as optical transceivers, which are also capable to incorporate depth measurements into the communication protocol. Our approach has the potential to reach a throughput of several kilobits per second.

Operation Principle of the OptiSec3D approach

Indirect Time-of-Flight sensing works by emitting pulsed infrared light. The active illumination unit of such system can be used as transmitter, since it is designed to emit pulsed infrared light at different phase-shifts. The pixels of the receiving Time-of-Flight image sensor are capable to demodulate the phase-shifted signal. This allows optical communication based on pulsed light phase-shift keying (PLPSK).
The vast advantage of PLPSK is that multiple bits can be encoded in one image. In most image sensor based approaches, simple binary modulation schemes, such as on/off keying (Roberts, 2013) or pulse position keying are used. They support the transmission of one bit per frame at best. PLPSK takes advantage of the photonic mixer device (PMD), located on each pixel of a Time-of-Flight sensor. The PMD decodes phase differences of incoming light pulses, by sorting the incoming charges into charge

storage buckets A and B on the pixels. After the readout process, the voltage difference of these buckets is proportional to the phase difference between the own modulation signal and the incoming light pulses. If these light pulses are phase-modulated, the output of the ToF sensor contains the decoded phase offset signal. Since at least four equidistant phases are used during communication, it is possible to decode phase differences by just using one frame, instead of at least four frames used during depth measurement.

Channel Characteristics

Due to the measurement principle of the PMD on each pixel, the sensor is sensitive to pulsed light within a certain frequency range. Non-pulsed background light does not have a direct influence on the measurement. The photons of continuous light arrive during both switching states of the PMD with near equal intensity. This fills both charge buckets equally, leading to increased noise but no measurement bias. Due to this principle, extensive image processing is unnecessary, as it can be assumed, that all detected signals originates from potential communication partners.
Another side-effect of the PMD pixels is that sensors can choose modulation frequencies from a large spectrum to communicate. Stray light from different connections from different systems do not directly influence the measurement, as long the light pulse frequency is just several thousand Hertz apart. There exist no experimental evaluations so far, but a viable spectrum of 16 to 26 MHz can potentially yield to 1000 different channels, with a rather large distance of 10000 Hz. Using fast-switching VCELs, instead of LEDs, the upper boundary of the spectrum can be extended to over 100 MHz.

Image sensor based optical communication usually suffers from the low frame-rates of the sensors. While there exist experimental dedicated communication image sensors (I. Takai, 2013), the necessary digitization of complete frames has been holding back high speed image sensor based communication links so far.
Time-of-Flight sensors need to capture up to eight phase images in order to create one depth image. Therefore the readout and analog-digital conversion circuitry is often optimized for fast readout and digitization.
In order to demodulate a line-of-sight communication signal, just a very short exposure time is required. These characteristics enable high frame-rates during optical communication. We manage to operate our Time-of-Flight sensor at 500 frames per second with full sensor readout. This however can be massively increased, if the sensor is only read-out partially. By configuring the readout region to a minimum of 16x32 pixels, we are able to reach 7300 frames per second. This can lead to a transmission throughput of 14600 bits per second, when using 4-PLPSK. This high framerate leads to fast light source tracking capabilities, supported by fast adoption of the readout window, to accommodate moving targets.

Sensor Synchronization

A technical challenge throughout many communication systems is synchronization. In Time-of-Flight based communication, it is important that the frequency of the modulation signal of a ToF camera matches the frequency of its communication partner. If the modulation frequency of a ToF sensor is different from the frequency of the incoming light pulses, the measured phase values start to drift. Figure 3 shows sampled phase values, when the sender continuously emits pulsed infrared light without phase-shift. If this signal is measured and digitized, the frequency of this signal is the absolute modulation frequency difference between both communication partners.
Synchronization can be accomplished by sampling the incoming light pulses, and calculating the frequency difference of the communication partner. The modulation frequency of Time-of-Flight cameras is usually configurable to accommodate different use-cases. In the case of OptiSec3D, the frequency is adapted by configuring a phase locked loop (PLL) on the sensor. Synchronization can be reached, if the PLL is adapted by the measured frequency difference.

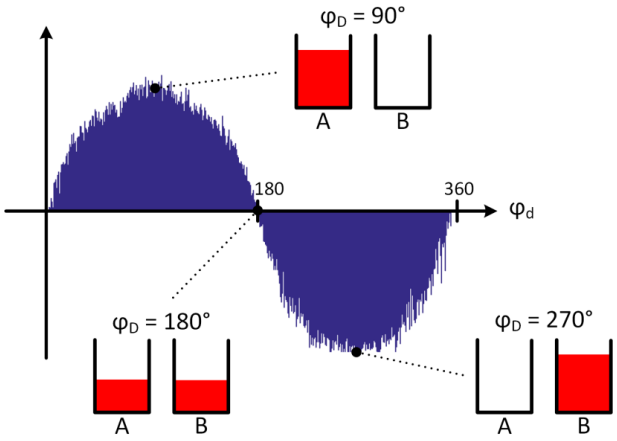


Fig. 3 The observed phase with corresponding charge bucket contents, if sender and receiver are not synchronized

Implementation of OptiSec3D

The core of each OptiSec3D communication partner is an Infineon Real3™ 3D imaging sensor, based on Time-of-Flight technology of pmd technologies. The Xilinx Zynq 7000 platform is used in our platform to operate the sensor with software executed on its integrated ARM processors, while the FPGA is used as glue logic and for imaging data transmission. The software uses an I2C bus to configure the ToF imaging sensor. This live-configuration of the sensor allows changing the internal workings of a normal depth sensor in such way, that it is possible to transmit and decode data. This works by limiting the number of digitized pixels per frame to a small area around modulated lightsources. With this configuration, the sensor is able to sample the image of the communication partner at over 7300 frames per second. The received signal directly contains the transmitted decoded information, since ToF pixels are sensitive to phase-shift differences.

Communication Modes

Time-of-Flight cameras could be either used to communicate with each other, or with different electronic devices. While it is not difficult to implement a PLPSK transmitter, receiving PLPSK is not trivial, since a photonic mixer device is required. ToF cameras however are able employ alternative modulation schemes, such as pulse position modulation. A receiver would just need to be able to detect the presence of light within certain time-slots. Time-shifts could be implemented either directly by accessing the illumination unit, by varying the frame-rate, or the number of read-out pixels.

Optical line-of-sight communication is not limited to two communication partners. The aforementioned concepts can be employed to multiplex communication between multiple partners. For synchronization, the adapted frequency for each communication partner can be stored, and the PLL adapted each time, when switching the focus to a different partner. If multiple devices want to communicate with a single node, all other devices could alternatively adjust their frequency to the node. If the focus of an application is on low latency rather than throughput, the sensor can be read-out completely, instead of adapting the readout region to the location of each communication partner.

Localization Principles with Location-Aware Optical Communication

Location-awareness and re-usability are the main motivations for optical communication based on image sensors. While 2D image sensors are only capable to determine the incident angles of a line-of-sight connection, Time-of-Flight sensors can locate communication partners in 3D.

Since Time-of-Flight imaging systems can be used as optical transmitter and receiver, the simplest use case is optical communication between two Time-of-Flight cameras. When both communication partners measure their mutual distance, they can both locate each other in 3D. In the application of e.g. encrypted device authentication, both partners can forward their mutual distance measurement and check for consistency. This effectively defeats relay attacks, where an attacker relays communication without alteration, using two relay boxes. In such relay attack, the distance between sender and relay box A is not consistent to the distance of receiver and relay box B.

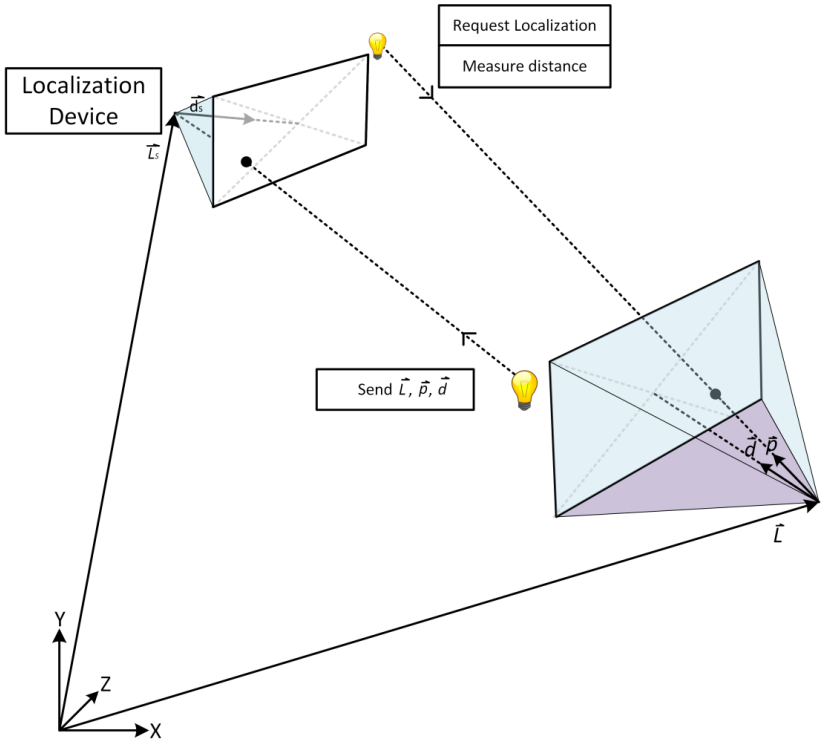


Fig. 4 Depth sensing, combined with optical communication leads to 5-dof localization with just one base-station. The base-station is equipped with an image sensor, and transmits its position $\vec{L}$ and the direction $\vec{p}$ vector of the localization device. The localization device can calculate its own position $\vec{L_S}$ and direction $\vec{d_S}$ by combining the received information with a distance measurement.

Communication between Time-of-Flight cameras could also be employed in localization solutions. In that case, a stationary camera with a light emitter serves as beacon. Electronic devices, equipped with a Time-of-Flight camera can contact one beacon to determine its position and orientation. An example could be an autonomous robot, desiring to navigate around a building.
The beacon's purpose is to forward its own position and the angles relative to the device. When simple LED beacons are used, at least three of them are necessary to determine the camera's position, even when the camera can determine the distance to each beacon. If the beacon however features an image sensor, as depicted in Fig. 4, the incidence angle of the line-of-sight can be determined by the beacon and forwarded to the device.
This enables 5-dof localization of embedded devices, using a single beacon. The only unknown degree of freedom is the roll angle, since the beacons optical signature is invariant to rotations around this axis. Sensor fusion with a gravity sensor or computer vision methods can effectively help reach full 6-dof localization. In the case of autonomous robots, this angle might already be locked due to the camera mounting method.
While beaconing with Time-of-Flight sensors enables a miniaturized localization system, simpler and cheaper beacons might be desirable. If three or more beacons are visible at the same time, a device using an imaging sensor can reach 6-dof localization by solving the perspective-n-point problem.

When using depth sensors, the relative 3D position between camera and each beacon is directly available. This enhances the positioning robustness, since both trilateration and triangulation can be used to determine the position.

When no absolute localization within a predetermined coordinate system is desired, it is sufficient to use beacons which initially do not hold information about their location. A device can determine the relative positions of the beacons by using a 3D camera. With optical communication, it is possible to assign IDs to each of them and re-localize them in the local coordinate system. It is also possible to forward localization information to these beacons, so that simpler devices with 2D cameras can later on use them for navigation. In the last section of this chapter, the idea of forwarding positional information to small IoT devices is further discussed.

Augmented Internet of Things

So far, the main focus of augmented reality (AR) is fusing virtual and real worlds in order to receive an augmented world. The main mechanisms so far are putting virtual objects into a live camera stream (e.g. Google Tango, Qualcomm Vuforia), or embedding them into the viewport of a user (e.g. Microsoft HoloLens). We propose a concept with our OpticSec3D approach, to use 3D location-aware optical communication in order to enable embedded devices to interact with the augmented world. This would enable electronic devices within the viewport of an AR, to transfer information and interaction possibilities to the augmented world. These devices could be any kind of system, requiring human interaction, such as light switches, payment terminals, heating and climate control. They could also serve as virtual signs, or display promotions in supermarkets, or be used for pairing with local WIFI or Bluetooth connections.

A Time-of-Flight depth sensor on such AR system can receive optical signals from such devices and can use distance measurements to determine the 3D position. When the 3D position relative to the AR device is known, it can be embedded into the augmented reality. The concept, of mapping a 3D position into AR space, is shown in Fig. 5. Since the pixel position of the depth camera is different to the augmented color image stream, the 3D position needs to be transformed to color camera image space and projected to the 2D image space.

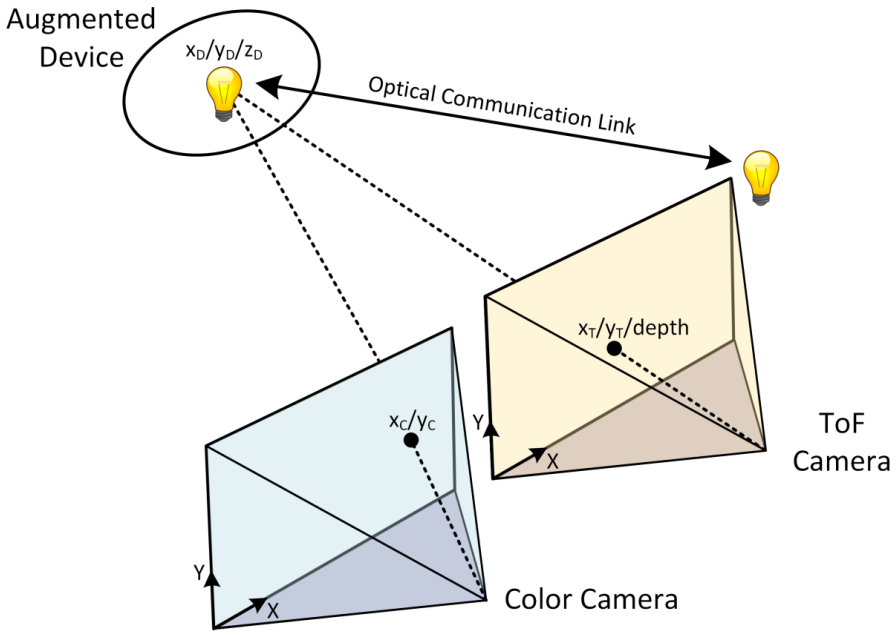


Fig. 5 With 3D-location-aware optical communication, electronic devices can be embedded into augmented reality. A Time-of-Flight sensor is able to combine depth measurements with pixel coordinates (x_T, y_T) to calculate the 3D coordinates (x_D, y_D, z_D) of the augmented device. Using calibrated camera parameters, it is possible to transform and project the 3D position to the image coordinates (x_C y_C) of an augmented image stream.

When the 3D position of a stationary device is determined, there are two possibilities. On the one hand, AR device can be equipped with a variety of pose tracking sensors. In this case, the 3D position can be placed into the augmented coordinate system and the pose tracking sensor would keep track of the position at this point. If the AR system does not feature such additional tracking system, the position solely relies on tracking the modulated light source. This requires continuous tracking, which means, that depth measurements need to be incorporated into the communication protocol.
It is however sufficient to conduct these measurements infrequently, since only larger changes in distance create different pixel mappings to the augmented reality stream. For rotations, it is possible to track the incident angle of the modulated light sources during communication with low latency.

CASE STUDIES ON OPTICAL LOCALIZATION AND TRACKING SYSTEMS

As of 2016, the upcoming applications for high-performance localization and tracking solutions are found in the automotive field in automated driving systems, and in consumer electronics in augmented and virtual reality devices. In this section, we introduce such high-performance localization and tracking mechanisms and how they already found their way into products.

Google Tango

Google started Tango with the goal to provide smartphones and tablets a human-like understanding of the environment. The central ambition is to produce a reference design and software to enable smartphone and tablet vendors to implement Tango's functionality into their devices. A large part of Tango is to support application developers by providing support, tools and development kits.
A Tango enabled mobile device is able to record and locate itself in a 3D map of its environment and track its pose at a very high update rate. This enables indoor localization and augmented reality. Google categorizes the aspects of its platform into motion tracking, area learning and depth perception. These main features are mutually dependent from each other and rely on a sophisticated software and hardware implementation.

Motion tracking:
Project Tango devices feature 6-dof motion tracking. While current mobile devices are often capable of orientation tracking, by using gyroscopic magnetic and gravity sensors, Tango adds sophisticated 3D pose tracking. This is realized with a sensor fusion approach, including inertial sensors and a motion tracking camera.
The motion tracking camera has a very wide field of view and records black and white images. Tango detects features and uses frame-to-frame feature correspondences to detect camera movements. The data from the inertial sensors complements the data of the motion tracking camera to receive a higher update rate and to increase robustness during strong motions and featureless images.
The outcome from motion tracking is continuously available pose information, containing the position and orientation of the device, but neither provides a reference coordinate system or localization in a larger context.

Area learning:
The goal of area learning is to build a 3D model of the environment. This is accomplished by a simultaneous mapping and localization approach. Data from the color, depth and motion tracking sensors are combined to create a 3D presentation of the environment. Image features which are especially unique are saved as landmarks. According to Google, these landmarks are saved about every 50 cm of tracked camera displacement. Landmarks along with the available 3D data enable fast localization within previously recorded datasets. The captured area learning data can be exported and stored into area description files (ADF).
These files enable Tango devices to remember the environment without re-scanning. It is also possible to use externally provided area description files for localization. An example would be indoor navigation system, where the building was initially recorded with Tango. Waypoints for navigation could be placed into such an area description file to directly support navigation.
The area description file along with navigation information is then provided to Google Tango enabled devices, and they are able to precisely locate themselves in the coordinate system of the provided file. Navigation and location based services are possible, if the ADF is associated to mapping data.
Jeon et al. (2016) enhance this concept, by scanning the indoor environment with a 3D laserscanner, and use their own method to build a database. Google Tango enabled devices can use the database for localization, but also to push new data to the database. This enables to update the database automatically with change to the environments.

Depth understanding:
The latest iteration of Google Tango incorporates a depth sensor based on Time-of-Flight technology. This sensor is important for area learning, but also gives the Tango platform an immediate understanding of the geometric context. Since Time-of-Flight is a dense depth sensing method, it is possible to gather 3D information on texture-less flat surfaces. This helps to create a denser map during area learning, but also enables the augmented reality applications which are one of the most significant selling points of the platform. With depth understanding, it is possible to integrate virtual objects into the augmented world. Besides gaming and other entertainment purposes, depth sensing enables to measure distances without measuring tape, and even preview furniture in its determined surrounding before buying.

Head-mounted Virtual Reality Devices

Virtual reality is currently the most demanding application for indoor positioning and tracking systems. The demand for operating range of consumer-grade head mounted devices (HMD) is limited, but the precision, update rate and latency are critical. User experience motion sickness, if there is a discrepancy between their motion and visual stimuli. A common term is the motion to photon delay, which is the time from a user's movement to the point, when the displays in the HMD update their images to accommodate the movement.
The main measure to reduce this delay is a high display frame-rate. This is currently 90 frames per second in the current top products such as the Oculus Rift and the HTC Vive. On the other hand, the

tracking sensors require an even higher update rate. Since visual output and tracking are usually not synchronized, a higher tracking update rate enables to associate frames with better tracking results.

The common denominator of current VR tracking systems is that they are based on sensor fusion approaches, using MEMS-based inertial sensors for a high update rates in combination with optical sensors for positioning and error correction.
For the optical tracking part, there exist two major paradigms, as illustrated in Fig. 6. While the exact definition is debated, the main difference is the position of the optical localization device (Foxlin, 2002). An inside-out tracking system has the localization device mounted on the device, while outside-in systems have a stationary tracking device, facing the tracked object. Fixed reference points are used to localize the device within a given coordinate system. The fixed locations can be either active beacons, or passive visual markers. Even normal visual features might be sufficient in future iterations of HMD tracking systems. In 2016, Oculus invited journalists to demonstrate a prototype, based on Inside-out tracking and just using visual features of a typical living room environment (Orland, 2016).

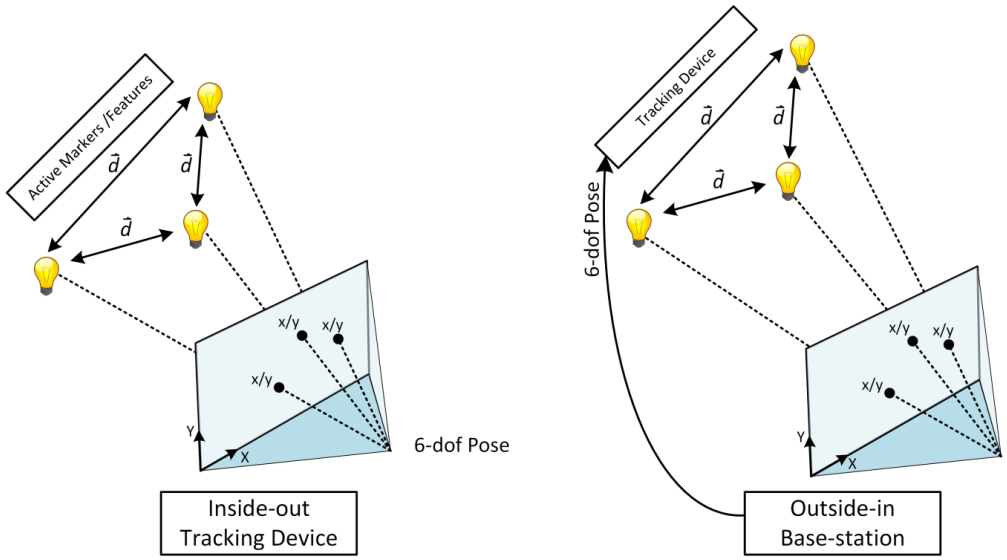


Fig. 6 The difference between outside-in and inside-out tracking. 2D-projections (x,y) of features or active markers with known relations $\vec{d}$, are used to determine the 6-dof pose of the tracking device. In inside-out tracking, the tracking device observes the static environment. In outside-in tracking systems a stationary base-station observes the device and forwards the pose over a communication link.

The advantage of inside-out tracking is that the system is not limited to certain boundaries, as long as enough reference points are within the field of view of the tracking sensors. A problem of practical implementations however is that usually with 2D image sensors, only incident angles of the line-of-sight to the reference points can be measured. Since it is not feasible in consumer electronic, to let the user calibrate the positions of the reference points, the points need to be located. This can be achieved by image to image correspondences, but requires view disparity and is subject to inaccuracy. Another reason, why as of 2016, inside-out tracking is not used in virtual reality is that the pose tracking precision of active visual markers depends on the distribution of the markers. At least three markers are required to be in sight at the same time, and they must not be arranged in a straight line.
An outside-in tracking system has one or more localization sensors mounted on a fixed point, while the tracked objects are observed. This currently dominant tracking paradigm can be categorized into the following principles:

Tracking with LED Markers and Image Sensors:
This technology is currently used by Oculus Rift, OSVR and PlayStation VR. These outside-in tracking systems are based on 2D infrared cameras, which sense active markers on the HMD. It case of the Oculus Rift, these markers consist of LED lights, which are toggled by the tracking system for identification. Since the positions of these markers on the HMD are fixed and known, finding the pose of the head-mounted device can be mapped to the perspective-n-point problem. This problem describes finding the 6-dof pose of a set of points in 3D space, by analyzing 2D projections on images, gathered by a calibrated camera. While the pose can be determined by sensing three or more points, the robustness and precision are enhanced by additional points. The problem with outside-in tracking is the stationary position of the camera. This limits the usage to viewport of the tracking camera. Due to limited sensor resolution, the precision decreases with distance limiting the tracking solution to medium sized rooms. This can however be counteracted by using multiple tracking cameras.

Tracking with Laser Beacons and Photodiodes:
This tracking solution is used by HTC Vive HMD in its Lighthouse tracking system. It is based on photosensitive element instead of image sensors. These elements are distributed among the surface of the device. Two base stations are positioned in front of the user. Each features two rotating lasers, which scan the X and Y axis of the room at precisely 60 Hz. A synchronization pulse sequence is

broadcasted to all sensors. Then each element counts the time until the laser beam reaches its position. With the timing information, it is possible to reconstruct the angle of the laser, when it reached the element. By using two lasers, it is possible to triangulate the position of the element. Using the 3D positions of all elements, it is possible to determine the 6-dof pose of the HMD. The advantages of this system, are that it has an increased range, minimum computation overhead, potentially lower latency, and the possibility to track multiple devices without additional effort in the base stations (Deyle, 2016). The update-rate however is too low for optical tracking alone, and thus sensor fusion with inertial sensors is necessary. Another advantage is that the base-station is independent of the tracked device. The positional information could be calculated on the tracked device itself.

The scanning principle of Lighthouse also permits 3rd parties to use the base stations to develop their own tracking solutions. Since inexpensive microcontrollers and photodiodes can be used, this tracking system has the potential to be widely used for localization and positioning in areas like internet of things or smart-homes.

State-of-the-art HMDs are dedicated to electronic entertainment. The devices are localized by the tracking system into a coordinate system, defined by the base stations. This pose is then transformed into the coordinate system of the virtual world. If the pose of the base station within a mapping system of the real world is known, virtual reality tracking systems can provide high fidelity localization. Another aspect is awareness of the immediate surroundings. Virtual worlds usually do not have boundaries, so the users need to be prevented from stumbling over objects or colliding with walls. An elegant solution would be to detect these obstacles by using a depth camera. The first generation of consumer grade HMDs does not feature such a camera, but let the user manually define virtual boundaries before usage.

Vehicle Localization

This section briefly introduces some special use-cases where different forms of localization and tracking are employed in the very important vehicular context.

Fleet management:
A company owing a number of cars (for example a taxi company or a truck company) can strongly benefit from knowing the current position of all their cars in order to improve the offered service. Since such systems do not require a very precisely measured location they get along with a GPS device. Those systems are inexpensive and are currently widely used for fleet management, since they can be built up using a mobile phone with GPS capability. A famous example is the taxi company Uber, which displays the current position of the closest available vehicles via their smartphone application. Additionally, the GPS in the car replaces the taximeter, since it tracks the actually driven distance and time.

Assisted driving functions:
Some assisted driving functions require environment data of the direct surroundings, for example the detection of other cars and objects for parking assistance or collision avoidance (Winner, 2015). For that purpose sensors for object detection are mounted on modern vehicles. Those include radar sensors, LiDAR sensors, ultrasonic sensors and cameras. Since each of those sensors has shortcomings in certain environments, typically more than one sensor technology is implemented and considered by the assisted driving system.
Other assisted driving functions require the vehicle to not only know its direct surroundings, but also know the current position on the map. It can be very beneficial for a collision avoidance system to have information about the current context (for example an urban area or a motorway) (Levinson, 2008). This can easily be acquired using GPS localization.
Then there are also systems which communicate with infrastructure objects (Vehicle-to-infrastructure, V2I) or other cars (Vehicle-to-vehicle, V2V) in order to gain additional knowledge of the surrounding environment. This can be information about dangers on the upcoming road, the timing of the next traffic lights or other relevant information for the vehicle. Today, the information is communicated

using the mobile network or a WiFi-like short range network. Yet, the presented novel optical line-of-sight communication techniques may be used in future.

Automated driving:
Although fully autonomous driving is not reality yet, automakers are putting a lot of effort in that topic in order to preserve their position in the market. Localization of a vehicle using only the GPS is considered to be insufficient for the use in (partwise) automated driving systems (Bar, 2014). The current approaches depend on up-to-date HD maps and the exact position of the vehicle within that map. The exact position in addition with the data from multiple environmental perception sensors enables the potential for fully autonomous driving.
The solely use of current GPS systems for localization is not precise and reliable enough for direct mapping into HD maps. Therefore the issue can be resolved by using the environmental perception sensors (radar, LiDAR, cameras) of the vehicle. For instance, Time-of-Flight based sensors can measure the distance to certain points of interest in the environment and align them with the corresponding points within the HD maps.
Using the inertial sensors and odometry data, it is possible to keep tracking the exact position on the map even if the GPS is in a non-functional state (for example in a tunnel).

FUTURE CONCEPTS

Inside-out Tracking without Active Markers

Vision based inside-out tracking has is already extensively been used in augmented reality. Google Tango, Microsoft HoloLens and Qualcomm Vuforia are well known examples. In these applications, the required latency and update rate allows feature based tracking and localization.
In augmented reality, latency is more tolerable, than in virtual reality, because it does not cause motion sickness. The 2D imaging pipeline of an inside-out tracking system introduces such latency which creates a massive technical hurdle in high fidelity tracking and localization applications such as virtual reality. Despite these circumstances, Oculus presented a working prototype in 2016, using four motion capture cameras in order to provide 6-dof tracking without requiring any peripherals.

Position forwarding to IoT devices

With an increasing number of 6-dof location-aware devices on the consumer market, it is possible to use image sensors along with optical communication to detect and localize embedded devices of all kind. An example could be a smartphone equipped with a location-aware AR platform such as Google Tango and an optical communication solution. By sensing the presence of embedded IoT devices, it would be possible to localize them and use a communication channel to directly forward them their position.
Potential applications involve secure device pairing, location-aware temperature control, geometry-aware audio systems or all kinds of intelligent sensors. If desired, surveillance cameras could be provided their pose in order to be able to reliably track persons throughout buildings.

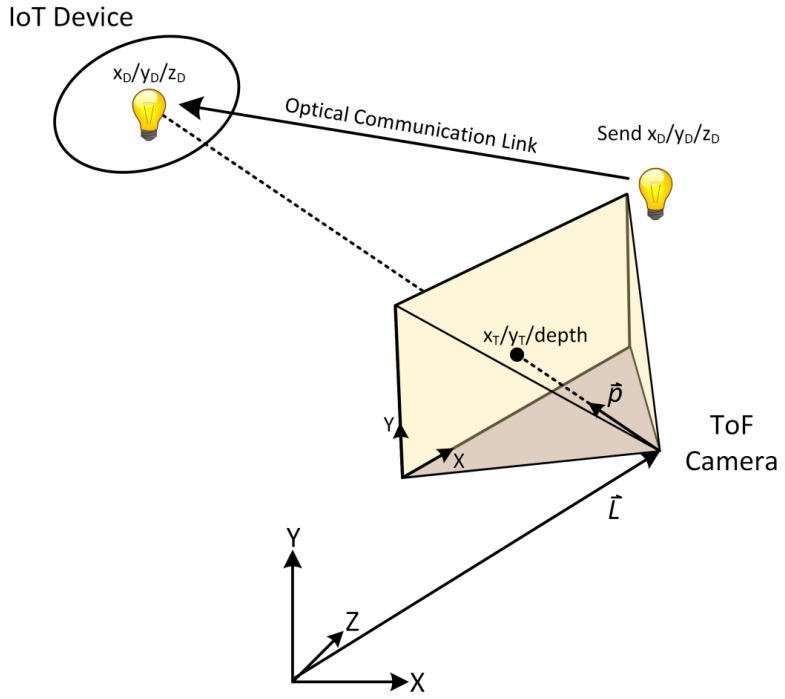


Fig. 7 A device is aware of its own location $\vec{L}$ and orientation $\vec{p}$ is able to localize embedded IoT devices in 3D. It can forward the localization information via optical communication.

In professional settings such as fabrication, workshops or warehouses, the location of tools and parts can be determined by using optical communication along with positioning.

CONCLUSION

This chapter discusses concepts, methods and opportunities for optical localization of cyber-physical systems and future IoT devices. A focus is placed on Time-of-Flight depth imaging systems, and how they can benefit existing and future localization systems. Examples of high-performance localization and tracking systems in existing products are introduced in form of case-studies.
As the field of visible light communication shows, optical communication is a crucial aspect of modern optical localization systems. We introduce our own location-aware optical communication approach, based on Time-of-Flight depth sensors. With this approach, it is going to be possible to localize communication partners in 3D with high accuracy and very low latency.
We are also confident that 3D-location aware optical communication will close the gap between augmented reality and IoT, by enabling embedded devices to participate in an augmented world.

REFERENCES

(2016). (Wikitude GmbH) Retrieved 12 05, 2016, from Wikitude: http://www.wikitude.com/

Biswas, J., & Veloso, M. (2012). Depth Camera based Localization and Navigation for Indoor Mobile Robots. *Proceedings of IEEE International Conference on Robotics and Automation.*

Boger, Y. (2016). (OSVR) Retrieved 11 21, 2016, from The VRguy's Blog: http://vrguy.blogspot.de/

Bouguet, J.-Y. (2016, 12 11). Retrieved from Camera Calibration Toolbox for Matlab: https://www.vision.caltech.edu/bouguetj/calib_doc/#start

Campo-Jimenez, G., Martin Perandones, J., & Lopez-Hernandez, F. (2013). A VLC-based beacon location system for mobile applications. *International Conference on Localization and GNSS* (pp. 25–27). Turin: IEEE.

Censi, A., Strubel, J., Brandli, C., Delbruck, T., & Scaramuzza, D. (2013). Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor. *IEEE/RSJ International Conference on Intelligent Robots and Systems 2013.*

Chetverikov, D., Eichhardt, I., & Jankó, Z. (2015). A Brief Survey of Image-Based Depth Upsampling. *KEPAF*.

Dai, L., Zhang, F., Mei, X., & Zhang, X. (2015). Fast Minimax Path-Based Joint Depth Interpolation. *IEEE Signal Processing Letters*, 623-627.

Deyle, T. (2016, 12 11). *Valve's "Lighthouse" Tracking System May Be Big News for Robotics*. Retrieved from Hizook: http://www.hizook.com/blog/2015/05/17/valves-lighthouse-tracking-system-may-be-big-news-robotics

Dhome, K. M.-B.-A. (2015). Bundle adjustment revisited for SLAM with RGBD sensors . *14th IAPR International Conference on Machine Vision Applications (MVA).*

DIY Position Tracking using HTC Vive's Lighthouse. (2016, 12 11). (Github) Retrieved 11 21, 2016, from Github: https://github.com/ashtuchkin/vive-diy-position-sensor

Do, T.-H., & Yoo, M. (2016). An in-Depth Survey of Visible Light Communication Based Positioning Systems. *Sensors*.

Ercan, A. T. (2015). Fusing Inertial Sensor Data in an Extended Kalman Filter for 3D Camera Tracking. *IEEE Transactions on Image Processing*.

Ferstl, D., Reinbacher, C., Ranftl, R., Ruether, M., & Bischof, H. (2013). Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation. *013 IEEE International Conference on Computer Vision*, (pp. 993-1000). Sydney.

Foxlin, G. W. (2002). Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications , 22*(6), 24 - 38.

I. Takai, S. I. (2013). LED and CMOS image sensor based optical wireless communication system for automotive applications. *IEEE Photonics Journal*.

Jeon, C. J., Ji, M., Kim, J., Park, S., & Cho, Y. (2016). Design of positioning DB automatic update method using Google tango tablet for image based localization system. *Eighth International Conference on Ubiquitous and Future Networks (ICUFN).* Vienna: 644-646.

Jovicic, A. (2016). *Qualcomm Lumicast: A high accuracy indoor positioning system based on visible light communication.* Qualcomm Flarion Technologies.

Kopf, J., Cohen, M. F., & Lischinski, D. (2007). Joint Bilateral Upsampling. *SIGGRAPH* .

Lee, S., & Jung, S. (2012). Location awareness using Angle-of-arrival based circular-PD-array for visible light. *In Proceedings of the 18th Asia-Pacific Conference on Communications (APCC),*, (pp. 480–485). Jeju Island.

Lepetit, V., Arth, C., Pirchheim, C., Ventura, J., & Schmalstieg, D. (2015). Instant Outdoor Localization and SLAM Initialization. *Proceedings of the International Symposium on Mixed and Augmented Reality*.

Liu, M., Qiu, K., Che, F., Li, S., Hussain, B., Wu, L., & Yue, C. (2014). Towards indoor localization using visible light. *International Conference on Intelligent Robots and Systems (IROS)*, (pp. 143–148). Chicago.

Mauro Biagi, S. P. (2015). LAST: A Framework to Localize, Access, Schedule, and Transmit in Indoor VLC Systems. *Journal of Lightwave Technology*, (pp. 1872-1887).

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., . . . Fitzgibbon, A. (2011). KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology.*

Orland, K. (2016). *Oculus working on wireless headset with "inside-out tracking"*. Retrieved from ars Technica: http://arstechnica.com/gaming/2016/10/oculus-working-on-wireless-headset-with-inside-out-tracking/

Plank, H., Holweg, G., Herndl, T., & Druml, N. (2016). High performance Time-of-Flight and color sensor fusion with image-guided depth super resolution. *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, (pp. 1213-1218). Dresden.

Roberts, R. D. (2013). A MIMO protocol for camera communications (CamCom) using undersampled frequency shift ON-OFF keying (UFSOOK). *Globecom Workshops*, (pp. 1052-1057).

Shen, Y. L. (2015). Dense visual-inertial odometry for tracking of aggressive motions. *IEEE International Conference on Robotics and Biomimetics (ROBIO).*

Steinbrücker, F., Sturm, J., & Cremers, D. (2011). Real-time visual odometry from dense RGB-D images. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops).* Barcelona.

Tobias, M. H. (2005). Robust 3D Measurement with PMD Sensors. *Range Imaging Day,*. Zürich.

Törnqvist, H. O. (2013). Why would i want a gyroscope on my RGB-D sensor? *2013 IEEE Workshop on Robot Vision (WORV).*

Vedaldi, A., & Soatto, S. (2008). Localizing Objects With Smart Dictionaries. *ECCV*.

Woodman, O. J. (2007). An introduction to inertial navigation.

Yu, J., & Zhao, J. (2012). Segmentation of depth image using graph cut. *Fuzzy Systems and Knowledge Discovery (FSKD),*, (pp. 1934-1938).

Yuan, W., Howard, R., Dana, K., Raskar, R., Ashok, A., Gruteser, M., & Man-dayam, N. (2014). Phase messaging method for time-of-flight cameras. *Conference on Computational Photography (ICCP).*

Zhu, J., Wang, L., Yang, R., Davis, J. E., & Pan, Z. (2011). Reliability Fusion of Time-of-Flight Depth and Stereo Geometry for High Quality Depth Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1400-1414.

KEY TERMS AND DEFINITIONS

Active markers: A light source which serves as point of orientation for tracking and localization systems. Infrared light is predominately used, since it is invisible to the human eye.

Beacon: An optical beacon is an active light source which transmits its ID or position to the localization device.

Degree of Freedom: The position of an object in 3D space has a degree of freedom for position on each 3D axis. The orientation of an object has also three degrees of freedom which are the rotations around each 3D axle.

Head-mounted device: An electronic device, which is mounted on the head of a user. In the current form, either information is blended into the field view user (augmented reality), or the field of view is replaced by a displays, simulating a three dimensional space (virtual reality).

Time-of-Flight: In context of 3D sensing systems, this refers to the time photons need to travel to the measurement point and back to the device. This time is proportional to the measured distance.

Pose: Description of the geometric position and orientation of an object. A relative pose can be used to transfer objects to different coordinates systems or to describe movements.

Line-of-sight communication: Communication partners need to face each other for data exchange.

Design of a Low-Level Radar and Time-of-Flight Sensor Fusion Framework

Josef Steinbaeck*, Christian Steger†, Gerald Holweg*, and Norbert Druml*
*Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, gerald.holweg, norbert.druml}@infineon.com
†Graz University of Technology, Graz, Austria
steger@tugraz.at

***Abstract*—We present an open hardware and software platform to efficiently fuse heterogeneous sensor data in an automotive/robotic context. The framework presented in this paper provides researchers a base platform in order to develop and evaluate sensor fusion strategies. In contrast to similar approaches, this framework exploits in particular the raw radar data and enables the fusion at low-level.**

The proposed system utilizes low-level data from radar sensors as well as indirect (e.g. 3D imaging) and direct (e.g. LIDAR) Time-of-Flight (ToF) sensors. After a configurable amount of pre-processing at sensor-level, the sensor data is transferred to a centralized platform and aligned temporally and spatially. We demonstrate the transformation of radar data into the 3D coordinate system in order to fuse it with point cloud data from ToF sensors. Due to the modular structure of the framework, it also enables the exploration of various system partitioning concepts.

***Index Terms*—radar, automotive sensors, time-of-flight, autonomous vehicles, sensor fusion**

I. Introduction

Autonomous robots and vehicles are very popular topics of today. Various car manufacturers have already announced to introduce fully autonomous driving to the streets within the next decade. However, first working systems have been introduced as early as 2005 at the DARPA Grand Challenge [1], [2]. At this event, autonomous vehicles were able to drive through a 212 km path in the Mojave Desert in the southwest of the USA. Two years later, in the DARPA Urban Challange in 2007, vehicles of various teams navigated safely through an urban environment [3], [4], [5]. These vehicles already used a various number of different sensors (radar, LIDAR, ultrasonic, 2D-camera, etc.). Thus, the environmental perception architectures of these systems already provide a solid foundation in order to build a state-of-the-art perception module for a vehicle or a robot.

Since then, sensor technology has improved. New sensors are available and the processing power has increased. Especially the higher processing capabilities have straightened the way for machine-learning based perception approaches. Deep learning is a commonly used approach today and has different characteristics compared to the traditional rule-based algorithms.

Most of the latest passenger vehicles come with at least some sort of assisted driving functionality. The cars offer dedicated assisting functions like adaptive cruise control, lane assistance or blind-spot detection. These functions are usually provided by the sensor module and are isolated from the remaining system. Most of the current off-the-shelf automotive-qualified perception sensors do most of the processing in the module itself and only communicate the output to the remaining system via an automotive bus like CAN or FlexRay. Thus, these sensor modules are in general not favorable for autonomous vehicles, where decisions are made by a centralized module using the data from many different sensors.

Today, there exist various software frameworks which already come with interfaces to different available sensors and built-in libraries to process and utilize the sensor data. A prominent open-source solution for robotic software development is the ROS (Robot Operating System) [6]. The ROS framework was initially used in robotics, but is nowadays also used in autonomous vehicles. However, due to the lack of real-time capabilities it is rather used for scientific proposes and/or to demonstrate single components in the processing chain.

Due to the open-source nature of the ROS framework, the whole ROS ecosystem is rather built for open available robotic sensors than for automotive sensors. Thus, there is a lack of sensor interfaces for automotive sensors, which limits the research works in that field. In this work we present an approach to integrate a low-level radar sensor into the ROS framework. Additionally, we combine heterogeneous low-level environmental perception data in a centralized point.

To summarize, the contributions of this paper to the scientific community are:

- An open ROS architecture enabling the low-level fusion of heterogeneous environmental perception sensor data including temporal and spatial alignment.
- The integration of a low-level radar sensor into the ROS framework.
- The demonstration of a low-level radar and the fusion with direct/indirect ToF sensor data.

Section II gives a short overview of related approaches in research and industry and highlights similarities to our approach. The sensors used for the work in this paper are presented in Section III as well as the hardware platform and the software architecture in ROS. Additionally, we introduce a practical project where we employ the proposed sensor fusion framework. Afterwards, we present real-world measurement

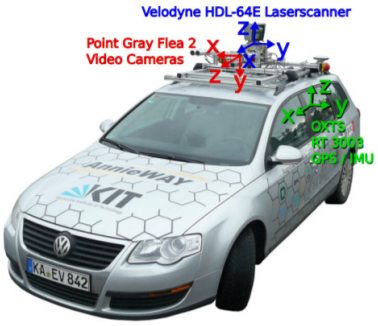


Fig. 1. Recording platform of the KITTI dataset. A standard passenger car is equipped with multiple cameras, a LIDAR scanner and a GPS localization system. Picture obtained from [9].

data using our framework (Section IV). Finally, Section V contains a short conclusion of the work.

II. Related Work

There already exists a vast amount of published work in the field of robotics and automated driving. However, due to the lack of market-available sensors with low-level access, no open sensor fusion framework exploiting low-level radar data exists. Thus, researchers face the burden to design and implement a custom system from scratch in order to evaluate novel sensor fusion strategies. Nevertheless, the architecture of already existing platforms reveals precious information in order to build a capable framework. Thus, this section presents previously published work including relevant information on how to build a framework.

A prominent approach to build a sensor platform for automated vehicles is published in [7]. The researchers built an autonomous vehicle with a set of market-available sensors (cameras and laser scanners). The platform is intended to act as base for researchers to develop algorithms for high-level tasks like scene recognition or path planning. They also released Autoware, a ROS-based software stack aimed for self-driving vehicles.

Researchers of the Carnegie Mellon University modified a regular passenger car in order to make it a research vehicle for autonomous driving [8]. The authors modified a stock car to be controllable via drive-by-wire. The vehicle uses GPS sensors, an IMU sensor, wheel odometry, as well as RTK corrections via mobile for an exact position detection. The environmental perception system of the vehicle uses multiple LIDAR and radar sensors as well as video cameras.

The authors of [9] built a recording platform which provides the KITTI dataset. A picture of the vehicle, equipped with various sensors is shown in Fig. 1. They built a sensor rig on top of a standard passenger vehicle and equipped it with a laser scanner and multiple video cameras. There exist an open benchmarking service allowing researchers to compare their algorithms to other existing ones.

The authors of [10] present the automotive sensor platform used to collect the *Oxford RobotCar Dataset* of sensor data in automotive environments. They use a stereo camera, monocular cameras, 2D LIDARs, a 3D LIDAR and a GPS navigation system. This dataset is also used to evaluate the performance of algorithms in the automotive domain (e.g. machine learning or mapping) using real-world sensor data. Multiple datasets of different scenes as well as a software development kit are openly available on the project website.

A very recent work to built an autonomy research platform is presented in [11]. They show a drive-by-wire conversion method of a market-available Toyota Prius V. Multiple relatively inexpensive sensors are mounted on the car in order to demonstrate autonomous vehicle functionality.

In contrast to the previously mentioned approaches, the work presented in [12] also describes the use of radar sensors for the environmental perception on their research vehicle. They perform radar and LIDAR sensor fusion with an architecture performing pre-processing at sensor level and centralized fusion. However, similar to other related publications, the presented approach depends on pre-processed radar data in the sensor module and does not exploit low-level information of the radar sensors.

To the best of our knowledge, there is currently no open sensor fusion platform available to the scientific community exploiting low-level radar data. This work offers researchers an open sensor fusion framework to explore novel sensor fusion concepts based on low-level radar data.

III. An Environmental Perception Framework For Low-Level Sensor Fusion

This work provides an open perception architecture available to the research community. The framework consists of off-the-shelf market-available components and provides scientists a ready-to-use framework to carry out research by using multiple environmental perception sensors. The framework is capable of making available and processing data from radar sensors, indirect/direct ToF sensors (e.g. LIDAR or ToF 3D imaging) and RGB cameras.

First, we present the sensors used in our setup. Afterwards, we show the implementation of pre-processing at sensor level as well as an open architecture to connect the heterogeneous sensors. Finally, we show a way to combine ToF and radar data within a common coordinate system. This common representation can be used for further data processing and algorithm development.

A. Used Sensors

We utilize a radar development kit, ToF sensors and a monocular camera. Each of these open-available sensors are presented in this section.

1) Radar: We selected the *RadarLog*[1] as radar sensor. This evaluation platform comes with a high-speed USB 3.0 interface capable of streaming the raw data from the radar sensor. We

[1]RadarLog: A platform for microwave radar data capturing and logging (http://www.inras.at).

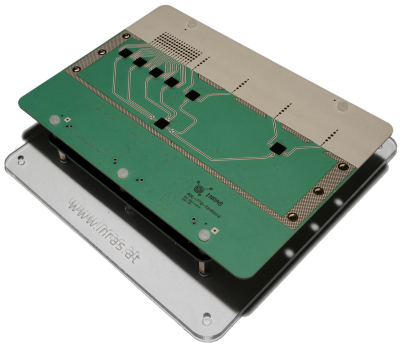


Fig. 2. RadarLog: radar development kit. The 77 GHz frontend provides 16 receive and four transmit channels.

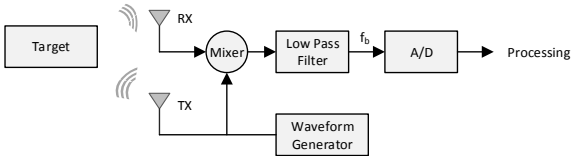


Fig. 3. Main building blocks of an FMCW radar. The transmitted and received signal are mixed to obtain a lower frequency used for digital signal processing.

use the kit with an RF frontend providing 16 receive and four transmit channels. Fig. 2 shows a picture of the RadarLog's frontend.

The radar sensor transmits electromagnetic waves and receives their reflections. In the automotive industry the majority of radar sensors use Frequency-Modulated-Continuous-Wave (FMCW) signals. A detailed state-of-the-art overview of automotive radar technology is published in [13]. The main building blocks of an FMCW radar are shown in Fig. 3. The RadarLog is capable of transmitting fast chirped FMCW signals (see Fig. 4).

The transmitted and received signals are mixed and low-pass filtered, resulting in the low-frequency beat signal f_b. The signal's frequency depends on the bandwidth B, the target's range R, the chirp duration T_p, the speed of light c, the center frequency f_c and the target's radial velocity v_r. As seen in Equation 1, the beat frequency consists of a range dependent part (f_{range}) and a velocity dependent part ($f_{doppler}$).

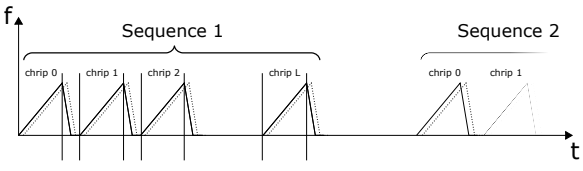


Fig. 4. Fast chirped FMCW radar sequences. One measurement consists of L chirps and every chirp is sampled at N times. This waveform enables simultaneous range and velocity measurement.

$$f_b = \underbrace{\frac{2BR}{T_p c}}_{f_{range}} + \underbrace{\frac{2 f_c v_r}{c}}_{f_{doppler}} \tag{1}$$

If the chirp duration (T_p) is very small, the range component outweighs the velocity component. Thus, $f_{doppler}$ can be neglected for single fast chirps and the range can be computed as seen in Equation 2.

$$R = \frac{c\,T_p}{2\,B} f_b \tag{2}$$

The maximum unambiguous range is determined by the duration of a full single chirp in order to not overlap the next chirp (Equation 3).

$$R_{max} = \frac{c\,T_p}{2} \tag{3}$$

The range resolution depends on the bandwidth of the frequency chirps B (Equation 4). A higher chirp bandwidth B results in a higher range resolution.

$$\Delta R = \frac{c}{2\,B} \tag{4}$$

The radial velocity is determined by evaluating the resulting phase shifts between L linear chirps. In order to unambiguously detect the phase difference between two consecutive chirps $\Delta\varphi$, it has to be lower than π (positive or negative). Thus, the maximum unambiguous velocity of a chirp FMCW radar is limited to the term as shown in Equation 5 [14].

$$v_{max} = \frac{c}{4\,T_p\,f_c} \tag{5}$$

The resolution of the radial velocity v of a chirp sequence of L chirps is calculated as seen in Equation 6 [14].

$$\Delta v = \frac{c}{2\,L\,T_p\,f_c} \tag{6}$$

Due to the low frequency of the beat signal, digital signal processing is possible at a reasonable complexity in order to determine the range and velocity of the targets. A 2D Fast Fourier Transform (FFT) is calculated using the beat signal in order to estimate the range and velocity of the radar targets.

Multiple radar receive antennas can be utilized to estimate the direction of arrival (DoA) and thus, the angle to radar targets (see Fig. 5). The different antennas detect the reflection of an object in the far-field with a phase-difference of $\Delta\phi$. Using the wavelength λ of the radar wave, the detected phase difference $\Delta\phi$ results in an additional distance Δx to the target:

$$\Delta x = \frac{\Delta\phi}{2\,\pi} \cdot \lambda \tag{7}$$

Using that relation, the angle θ to the target can be calculated as seen in Equation 8. In practice, the angular estimation is performed by calculating an additional FFT over the array of equally spaced ($d = \lambda/2$) receive channels.

$$\theta = \arcsin \frac{\Delta\phi\,\lambda}{2\pi\,d} \tag{8}$$

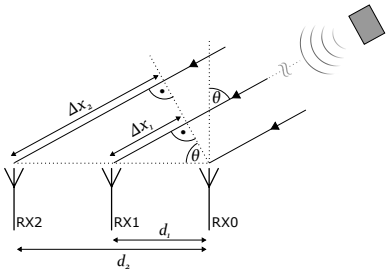


Fig. 5. Variation of the received signal phase due to the target's angle on an array of receive antennas. This phase difference can be used to estimate the direction of arrival (DoA).

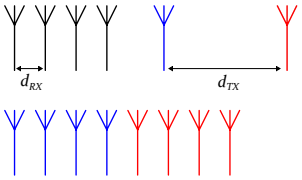


Fig. 6. Virtual antenna array. Multiple appropriate located transmit antennas can be used to form a bigger virtual antenna. This approach improves the angular resolution significantly.

A common way to further improve the angular resolution is to utilize multiple transmit antennas. The use of M_{TX} transmitters and M_{RX} receivers allows to form $M_{TX} \cdot M_{RX}$ transmit-receive pairs, so called *virtual antennas* (see Fig. 6). To achieve a maximum-sized virtual antenna, the distance between the transmitters is set to $d_{TX} = M_{RX} \cdot d_{RX}$. The distance between the receiving channels should not exceed $\lambda/2$ in order to achieve the maximum aperture and maintain unambiguity. The waveforms of the transmit antennas have to be orthogonal in order to differ at the receiver side. We use temporal multiplexing of the antennas. Thus, the single chirps are transmitted successively by every antenna, with only one antenna active at a time.

2) Time-of-Flight Camera: We selected ToF 3D imaging cameras (indirect ToF principle) as range sensors in order to capture 3D point clouds of the scene. The indirect ToF sensors can be exchanged easily by direct ToF sensors (e.g. LIDAR), since the obtained point cloud data is very similar.

A 3D imaging camera illuminates the scene with modulated infrared light. An array of photonic-mixing-device (PMD) pixels measures the phase difference between the transmitted an the received light. Using the phase difference, it is then efficiently possible to compute the distance for every pixel. The authors of [15] give a more detailed overview of the ToF principle.

The distance d to the reflected object can be determined using the speed of light $c \approx 3 \cdot 10^8 \frac{\mathrm{m}}{\mathrm{s}}$, the modulation frequency f_{mod} and the phase difference $\Delta\varphi$ (see Equation 9) [16].

$$d = \frac{1}{2} \cdot \frac{c}{f_{mod}} \cdot \frac{\Delta\varphi}{2\pi} \tag{9}$$

Fig. 7. Time-of-flight camera: CamBoard pico monstar from pmdtechnologies. The sensor provides a resolution of 352×287 pixels and a range of up to 6 m.

Equation 10 shows that the maximum unambiguous distance $d_{u,max}$ is limited by the modulation frequency [16].

$$d_{u,max} = \frac{1}{2} \cdot \frac{c}{f_{mod}} \tag{10}$$

The used ToF camera, a *CamBoard pico monstar*[2] is depicted in Fig. 7. This camera offers a resolution of 352×287 pixels and has a field of view (FOV) of 100°×85°. The sensor is connected and powered via USB 3.0 and provides a frame rate of up to 60 FPS.

3) Monocular camera: Since the focus of this work is on range data processing, we use the monocular camera mainly for reference purposes (e.g. for visual evaluation of measurement data). Thus, we selected a simple webcam (PlayStation Eye), capable of outputting 640×480 pixels at a frame rate of 60 FPS. The camera can be connected to the processing platform via an USB 2.0 interface. The structure of the provided data is similar to commonly used automotive cameras which are typically also running at comparable low resolutions in order to meet real-time processing requirements.

B. Hardware Architecture

This section describes the hardware architecture of our approach. Additionally, we provide information about the data rates of the used sensors. Fig. 8 shows a picture of the used sensor setup. The two ToF cameras are operating simultaneously with a different field of view. The radar sensor is mounted in forward-facing direction overlapping with the ToF camera's field of view. The sensors are connected via USB to low-level processing modules, capable of receiving the raw sensor data, pre-processing and raw data logging. The pre-processed sensor data is then transferred to a centralized sensor fusion platform, which is in charge of combining the heterogeneous sensor data. In the context of autonomous vehicles/robots, the fused data will then be further processed and used for higher level tasks (e.g. path-planning or localization).

Fig. 9 shows the block diagram of the used components. The processing platform(s) are standard x86 notebooks running Ubuntu. Depending on the target application, the whole

[2]CamBoard pico monstar: A 3D imaging development kit of the pico family (https://pmdtec.com/picofamily/).

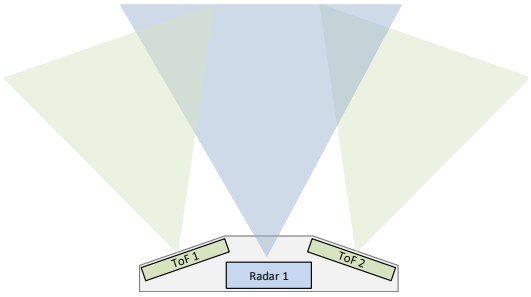


Fig. 8. Sensor setup arrangement with the field-of-view of every sensor visualized. The monocular camera is omitted in this figure for reasons of simplicity.

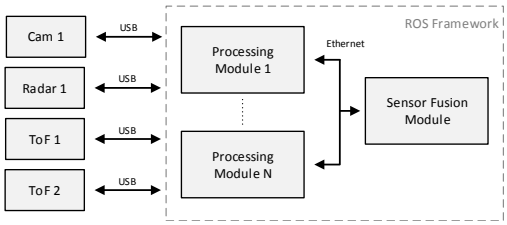


Fig. 9. Main processing blocks of the framework. The processing module(s) are in charge of low-level communication with the individual sensors as well as pre-processing and logging. These modules can be distributed onto multiple hardware platforms depending on the processing demand.

processing chain can be running on a single computer or be distributed onto multiple hardware platforms.

The amount of raw data from one radar measurement depends on the samples per chirp N, the number of chirps L and the receive antennas M. Thus, for a 16 Bit digital representation of a sample, the data size D_{Radar} of a single measurement is:

$$D_{Radar} = N \cdot M \cdot L \cdot 16\,\text{Bit} \tag{11}$$

The number of chirps affects the velocity resolution while the number of samples per chirp affects the range resolution. Additionally, a higher number of receive antennas increases the angular resolution. Thus, these parameters have to be chosen accordingly to the target application. Similar to 2D cameras, a more fine-grained data resolution of radar leads to a higher data rate. Typical values of N=1024 samples, L=128 chirps, M=16 antennas, a frame rate of 10 FPS, and a sample data depth of 16 Bit result in a data rate of about 335 MBit/s.

The ToF 3D imaging camera provides phase-difference data for every pixel. These values are transferred as 16 Bit integers. A popular and simple method to calculate the distance and amplitude value is to use four phase-difference values (raw measurements) for every pixel. Equation 12 shows the data size D_{ToF} of a single ToF measurement.

$$D_{ToF} = ToF\ Pixels \cdot 4 \cdot 16\,\text{Bit} \tag{12}$$

As a practical example, a ToF camera with a resolution of 352×287 pixels running at 15 FPS utilizes a data rate of about 194 MBit/s.

The monocular camera provides either uncompressed or JPEG compressed video. In the uncompressed mode three 8 Bit values per pixel are provided to the interface. Thus, the data rate at 15 FPS is about 111 MBit/s.

The sum of the raw data rates from multiple sensors already results in a network utilization rate exceeding the limits of many of today's automotive buses (CAN, Flexray). Thus, there is the need of a certain form of data compression at sensor level in order to enable the transfer without violating the networks maximum data rates.

C. Software Architecture

The sensor fusion framework's software is implemented using ROS nodes. Fig. 10 shows an overview of the ROS architecture. As seen in the figure, each sensor has its own ROS interface node in charge of the sensor specific data acquisition and the sensor configuration. In order to evaluate the data *offline*, all raw sensor data can be recorded by an independent logging node. The receive nodes add a timestamp to the sensor data in order to temporally align the asynchronous data correctly during further processing steps.

Every sensor type also comes with a configuration node in charge of translating a high-level configuration into the low level sensor specific representation. Each sensor's processing node performs the sensor-specific pre-processing of the raw data. This node can also perform some form of compression to decrease the communication overhead to the central fusion platform.

1) Radar: The received beat signal is sampled and converted to digital for each measurement sequence (see Fig. 3). The digital samples of a measurement are then transferred to the radar receive node. The samples are reshaped to a 3D matrix with the dimension of N samples, L chirps and M receive channels. Three FFTs are calculated in each dimension in order to obtain a range-velocity-angle matrix. The full data cube contains precious low-level information which is of high value for a centralized sensor fusion module. We implemented the radar ROS nodes in python, since the the evaluation kit comes with a python interface. The FFT calculation could also be done very efficiently in hardware. The authors of [17] show a parallel implementation of a 3D FFT processing algorithm on an FPGA.

In order to demonstrate a possible radar-compression, we implemented a cell-averaging constant false alarm rate (CA-CFAR) detection algorithm to identify peaks in the range-doppler image [18]. Afterwards we utilize the estimated angle of these peak points and perform clustering in the range-angle-velocity domain. The processing node publishes a custom ROS message `ScatterCenters` containing a list of all detected scattering centers.

2) Time-of-Flight: The ToF receiver node receives the raw sensor data (arrays of phase-difference measurements) and calculates the distance and amplitude value for every pixel (see

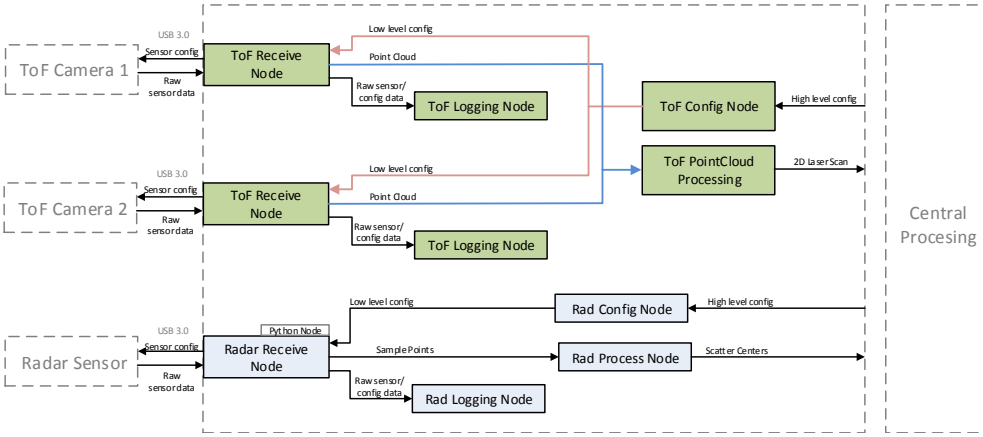


Fig. 10. Overview of the ROS architecture of the sensor pre-processing. Every sensor comes with a ROS node for logging, data-processing and configuration handling. The nodes of the monocular camera are omitted in this figure for simplicity reasons.

Equation 9. Using that information, the node calculates a point cloud and publishes a `sensor_msgs::PointCloud2` message containing the 3D points (32 Bit float) of the scene. Additionally a depth image (32 Bit float) and a gray-scaled amplitude image (32 Bit integer), used as a crucial low-level confidence metric, are published as `sensor_msgs::Image` messages.

The configuration node is used to dynamically configure the ToF camera, also during runtime. The most influencing parameters of a ToF camera are the exposure time, the frame rate and the modulation frequency. A higher exposure time extents the range and improves the signal-to-noise-ratio, but also limits the frame-rate of the system and causes motion blur/artifacts. The modulation frequency influences the unambiguous range of the camera, since the obtained phase value is ambiguous outside the $2\,\pi$ interval. However, two measurements with different modulation frequencies can be combined in order to extend the unambiguous range. In rather static environments and slow speeds, a low frame rate might be beneficial in order to enable a higher range and image quality.

3) Monocular camera: The monocular camera node works in a similar way as the other sensor nodes, but without actual raw data processing. The logging node logs the uncompressed video images while the processing node's only job is to publish the image data without any processing. The configuration node is in charge of setting the exposure time if the camera is not using auto-exposure mode.

D. Practical Application

An unmanned areal vehicle (UAV) is constructed as part of the Autonomous Car To Infrastructure communication mastering adVerse Environments (ACTIVE) project. This vehicle is equipped with multiple environmental perception sensors as well as an IMU, a GNSS and a vehicle to infrastructure (V2I) antenna.

The main goal of the project is to show the ability to achieve highly accurate positioning using V2I communication. A stationary road side unit (RSU) sends relevant map and positioning information to vehicles in its range. Utilizing this additional data, a car can determine its position very precisely. This can for example be beneficial at multi-lane intersections, where precise lane information can help to position the vehicle in the target lane.

The environmental perception sensors are mainly employed for the acquisition of real-world data in adverse environments. This data is used to evaluate the performance of the sensors in various operating conditions. The vehicle is intended to be used in a structured environment without obstacles or other vehicles present. However, a basic obstacle detection is implemented for safety reasons. This module makes use of the environmental perception data in order to avoid collisions during runtime.

IV. Results

This section contains measurements from a test scene using the proposed framework of this paper. The radar sensor, two ToF 3D imaging sensors and the monocular camera are used in order to perform environmental perception of a test scene. The used parameters and the obtained sensor data as well as the spatial alignment of ToF and radar data is shown in this section.

Fig. 11 shows an image of the test scene captured with the monocular camera. The test scene is an office space with a radar corner cube reflector placed at a distance of about 2 m from the sensor in an angle of about 5°.

Table I shows the parameters we used to configure the fast chirp sequences of the radar development kit. According to Equation 4, the maximum range resolution is 7.5 cm for a bandwidth of 2 GHz. The velocity resolution is about 0.17 m/s (Equation 6) with a maximum velocity of 4.8 m/s (Equation 5). The effective values during processing are additionally depending on the sample frequency and zero padding during the FFTs.

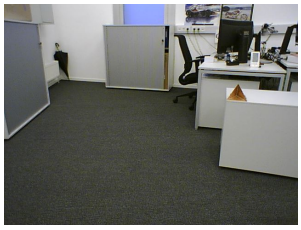


Fig. 11. RGB camera image of the office test scene. There is a corner cube reflector positioned at a distance of about 2 m to the sensor platform.

TABLE I
RADAR WAVEFORM CONFIGURATION PARAMETERS.

Parameter	Symbol	Value
start frequency	f_{start}	76 GHz
stop frequency	f_{stop}	78 GHz
up-ramp duration	T_{up}	128 μs
down-ramp duration	T_{down}	64 μs
pulse time	T_p	200 μs
samples per chirp	N	1024
number of chirps	L	128
receive channels	M_{RX}	16
transmit channels	M_{TX}	1
measurement time	T_{int}	1000 ms

Fig. 12 shows the range-doppler image of the test scene using the RadarLog kit with the configuration parameters stated in Table I. The range-doppler image calculation is performed by the radar processing node by calculating two FFTs. The range of the plot is cropped to 10 m in order to show the close proximity in more detail.

The range-angle plot obtained from the radar data is shown in Fig. 13. It shows the highly reflecting corner cube in front of the radar. The angular resolution is distributed non-linearly and decreases when moving away from the center line.

The ToF 3D imaging camera is used with the configuration parameters stated in Table II. The exposure time is a maximum value, since the exposure mode is set to auto-exposure.

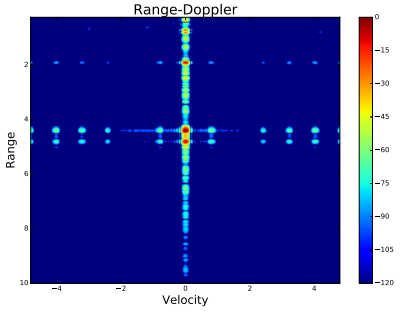


Fig. 12. Range-doppler plot of a single radar measurement. This image is obtained after the 2D FFT calculation over the N samples and L frequency chirps. The measured scene contains a wall at a distance of about 2 m slightly to the right of the radar sensor.

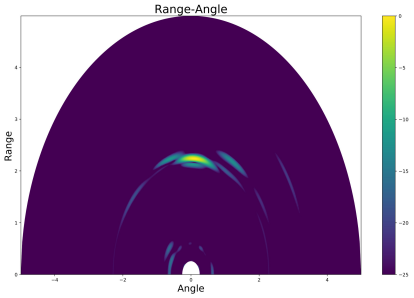


Fig. 13. Range-angle plot of a single radar measurement. This image is obtained after the 2D FFT calculation over the N samples and M receive channels. The measured scene contains a wall at a distance of about 2 m slightly to the right of the radar sensor.

TABLE II
TIME-OF-FLIGHT 3D IMAGING CONFIGURATION PARAMETERS.

Mode	Range
Exposure Time	1 ms
Frame Rate	10 FPS
Modulation Frequency	20 MHz

We captured sensor data with the setup shown in Fig. 8 and with the parameters listed in Table I and II. The radar sensor data is pre-processed to obtain scatter centers and fused with the ToF point clouds. A visualization of the spatially aligned output is shown in Fig. 14. The scatter centers are visualized using cylinders with the diameter as indicator for the peak power for the detected reflection. As seen in the figure, the obvious object is reflected very well.

V. CONCLUSION

A low-level sensor fusion platform was designed containing a radar sensor, multiple ToF sensors and a monocular camera. We are integrating the presented system as part of the environmental perception platform on an unmanned areal vehicle. The platform is capable of measuring real-world sensor data for *offline* evaluation. Additionally, the pre-processed data is used as an input for the emergency braking module which is in charge of detecting obstacles during runtime.

The integration of the radar sensor into ROS allows the utilization of low-level radar data into existing perception systems. There exist a vast amount of ROS packages, which can be used for further data processing and sensor fusion depending on the target system. The hardware and software framework built in this work is a crucial foundation to perform low-level radar measurements. Researchers can use the proposed platform as a starting point to rapidly evaluate/test novel sensor fusion strategies.

ACKNOWLEDGMENTS

The authors of this paper would like to thank the Austrian Research Promotion Agency for funding the Autonomous CarTo Infrastructure communication mastering adVerse Environments (ACTIVE) project with the number 855010.

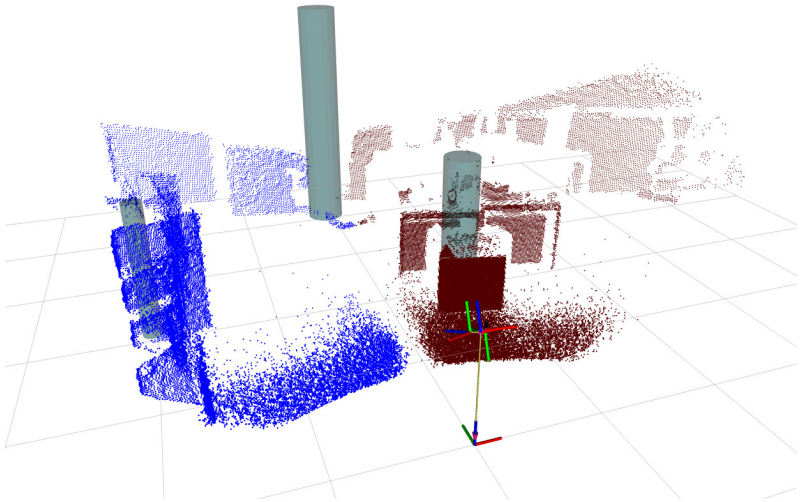


Fig. 14. Visualization of two ToF pointclouds and radar targets using the ROS visualization tool RViz. The radar targets are shown as cylinders with the diameter depending on the peak power. The cylinder's heights indicate the uncertainty in the vertical angle and depend on the radar sensor's field-of-view.

References

[1] C. Urmson, C. Ragusa, D. Ray, J. Anhalt, D. Bartz, T. Galatali, A. Gutierrez, J. Johnston, S. Harbaugh, H. Yu Kato, W. Messner, N. Miller, K. Peterson, B. Smith, J. Snider, S. Spiker, J. Ziglar, W. Red Whittaker, M. Clark, P. Koon, A. Mosher, and J. Struble, "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, vol. 23, no. 8, pp. 467–508, 2006.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, September 2008.

[4] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, August 2008.

[5] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, October 2008.

[6] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[7] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, November 2015.

[8] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2013, pp. 763–770.

[9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[10] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research*, vol. 3, no. 2014, 2014.

[11] F. Naser, D. Dorhout, S. Proulx, S. D. Pendleton, H. Andersen, W. Schwarting, L. Paull, J. Alonso-Mora, M. H. Ang, S. Karaman, R. Tedrake, J. Leonard, and D. Rus, "A parallel autonomy research platform," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 933–940.

[12] D. Göhring, M. Wang, M. Schnürmacher, and T. Ganjineh, "Radar/lidar sensor fusion for car-following on highways," *The 5th International Conference on Automation, Robotics and Applications*, pp. 407–412, 2011.

[13] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Next generation radar sensors in automotive sensor fusion systems," in *2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2017, pp. 1–6.

[14] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Short-range FMCW monopulse radar for hand-gesture sensing," *IEEE National Radar Conference - Proceedings*, vol. 2015-June, no. June, pp. 1491–1496, 2015.

[15] N. Druml, G. Fleischmann, C. Heidenreich, A. Leitner, H. Martin, T. Herndl, and G. Holweg, "Time-of-flight 3d imaging for mixed-critical systems," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, July 2015, pp. 1432–1437.

[16] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, March 2001.

[17] E. Hyun, W. Oh, and J. H. Lee, "Two-Step Moving Target Detection Algorithm for Automotive 77 GHz FMCW Radar," *2010 IEEE 72nd Vehicular Technology Conference - Fall*, pp. 1–5, 2010.

[18] F. Meinl, E. Schubert, M. Kunert, and H. Blume, "Real-Time Data Preprocessing for High-Resolution MIMO Radar Sensors," *Towards a Common Software/Hardware Methodology for Future Advanced Driver Assistance Systems*, p. 133, 2017.

Time-of-Flight Cameras for Parking Assistance: A Feasibility Study

Josef Steinbaeck[1], Norbert Druml[1], Allan Tengg[2], Christian Steger[3], and Bernhard Hillbrand[2]

[1]Infineon Technologies AG, Graz, Austria, {josef.steinbaeck, norbert.druml}@infineon.com
[2]Virtual Vehicle Research Center, Graz, Austria,{allan.tengg, bernhard.hillbrand}@v2c2.at
[3]Graz University of Technology, Graz, Austria, steger@tugraz.at

Parking assistance is one of the most demanded assisted driving functionalities of today. In contrast to 2D cameras, time-of-flight (ToF) sensors provide real 3D data which can be used to inform the driver about the vehicle's surroundings. State-of-the-art ToF cameras are compact, inexpensive and capable of providing high frame-rate 3D data with minimal computational overhead. In this paper, we evaluate the feasibility of ToF cameras used as perception sensors in parking assistance applications. ToF cameras are highly capable for parking assistance applications, but for outdoor usage, an enhanced illumination unit is necessary.

1. Introduction

Starting with May 2018, a new regulation in the USA and Canada became effective, making it mandatory for car manufacturers to install a rear-view camera into every new vehicle. The law was enforced to counteract the high number of accidents that occur while vehicles are backing up. A publication by the US National Highway Traffic Safety Administration reports an estimation of 292 total annual back-over fatalities within the USA [1]. Reasons include overlooking and occlusion of obstacles due to the limited field-of-view, particularly to the vehicle's back.

Modern technology is capable of providing additional information to the driver in order to support maneuvers at low speed. Rear-view cameras have already been integrated into many upper-class vehicles during the last decades. Multiple cameras around the ego vehicle can be used to assist the parking process with a top-down bird's view perspective of the close environment. Ultrasonic sensors have been used as distance sensors for parking assistance since a long time already. Many modern cars come with these sensors, since they are comparably small, cheap and easy to integrate into the vehicle's chassis. However, ultrasonic sensors are not precise enough to achieve a high resolution.

ToF cameras provide resolutions of over 100k pixels, millimeter precision and frame rates of more than 100 frames per second (FPS). In contrast to ultrasonic sensors, ToF data can be utilized to create an accurate 3D model of the close surroundings. In this paper, we evaluate the feasibility of a 3D ToF camera for parking assistance. We mounted one ToF camera on a passenger vehicle and inspected the data quality in different environments. Additionally, we designed parking assistance software which visualizes the 3D points to the driver. Considering the results of this real-world use case, we point out the advantages and disadvantages of ToF cameras for parking assistance. To sum up, the contributions of this paper are:

- Evaluation of ToF data for parking assistance.
- Exploration of different mounting options on a passenger vehicle.
- Visualization of the 3D data to the vehicle's dashboard.

2. Related Work

ToF cameras are used in various applications like augmented reality, face-tracking and 3D localization. Prominent approaches in the automotive context include driver/interior monitoring and hand-gesture recognition [2], [3]. Scheunert et al. already presented an approach where a PMD camera is used to detect the free space of a parking slot in 2007 [4]. However, the used camera with a resolution of 16x64 pixels is by far inferior to a modern ToF camera. Gallo et al. show an approach where a ToF camera is used to detect curbs and ramps in order to perform safe parking [5]. Ringbeck et al. mount a ToF camera on a car with a powerful light source (8W optical power) and achieve a range of up to 35 m [6].

A vision-based system, utilizing multiple fish-eye cameras around the vehicle to realize automatic parking is shown by Wang et al. [7]. The authors use inverse perspective mapping of four fish-eye images to provide a bird's eye view of the vehicle's close surrounding.

Compact, inexpensive and high-resolution ToF cameras are relatively new to the industry. To the best of our knowledge, there is no work available to the scientific community, evaluating a state-of-the-art ToF camera for parking assistance. This paper fills this gap by presenting an approach to utilize a modern (2018) ToF camera to obtain 3D data of the ego vehicle's environment.

3. A Time-of-Flight Camera for Parking Assistance

We mounted a ToF camera onto a passenger vehicle in order to use it with a parking assistance system. The ToF data is used to visualize depth information of the environment to the vehicle's dashboard screen.

3.1 Time-of-Flight Principle

The prevalent way to realize indirect ToF cameras is illuminate the scene with modulated infrared light and utilize photonic mixing device (PMD) pixels to detect the reflections. Each pixel measures a value that indicates the correlation between the received signal and a reference signal. The so called four-phase algorithm is used to determine the distance and amplitude for every pixel [8]. Four measurements with different phase-shifted versions of the transmitted signal as reference signal are used to calculate the phase difference and the amplitude for each pixel. The distance of a ToF pixel can be easily determined using the phase difference, the speed of light and the modulation frequency.

The pixels integrated in the selected ToF camera implement a suppression of background illumination (SBI) circuitry [6]. This circuit prevents the pixels from saturation when exposed to an unmodulated light source with a spectral component in the same range as the ToF working frequency (e.g., sunlight). However, the pixels still experience noise from ambient light. Thus, in applications with bright ambient light, the illumination power has to be selected accordingly.

3.2 Evaluation Setup

We use the *CamBoard pico monsta*r as ToF camera. The camera comes with the *IRS1125C REAL3 3D Image Sensor* developed by Infineon and pmdtechnologies. The image sensor has a resolution of 352×287 pixels, a field-of-view of 100°×85° and provides up to 60 frames per second. The illumination is performed by four vertical-cavity surface-emitting lasers (VCSEL) at a wavelength of 850 nm.

We mounted the ToF camera on a Ford Mondeo passenger vehicle, facing backwards, while considering the following requirements/trade-offs:

- The camera shall capture the edges of the vehicle in order to determine the boundaries.
- The ground area captured by the field-of-view shall be maximized.
- The mounting position shall be as far to the back of the vehicle as possible.

3.3 Visualization of Time-of-Flight 3D Data

We designed and implemented software to visualize the obtained 3D data on the dashboard screen of the vehicle. The data has to be presented in a simple enough way to even allow an untrained driver to interpret the data without effort.

The software uses the OpenGL API for rendering the 3D point cloud on the screen. There are several toolkits (i.e., GLUT or Open Inventor) available that simplify the visualization of 3D data. However, as the desired target platform is an automotive on-board computer, the native OpenGL API was chosen to keep the software as portable as possible. This low level API allows defining several properties of the scenery like the field-of-view of the camera, the viewing distance as well as light sources.

In that virtual 3D world, the data from the ToF camera is rendered as a bunch of cubes. Performance of both, the visualization software and the graphic hardware, is an important issue at that point, since about 100k points have to be drawn for every frame. By adding a 3D model of the ego vehicle, the viewer gets a better understanding of the scene. This requires an exact calibration and registration to preserve the scale of the voxels in relation to the ego vehicle. In the last step, the OpenGL's virtual camera is placed in the 3D world to view the scene from any perspective.

4. Results

The setup, as already presented in Section 3.2, was used to perform measurements in different parking environments. The first evaluation took place in an indoor parking lot without any sunlight present. The ToF camera provided an almost flawless amplitude and distance image. The close surroundings including the obstacles (parked car, walls) were clearly visible in the distance image. We performed another evaluation of the ToF data in an outdoor parking lot in bright sunlight. As expected, this scene resulted in a noisier amplitude image. Since low amplitude values correspond to a lower confidence, some pixels within the distance image were discarded.

In presence of bright sunlight, the *camBoard pico monstar* showed reduced performance. The main reason for this is that the camera's illumination unit is not optimized for outdoor, long-range applications and thus, produces insufficient laser output power. Yet, there already exist optimized outdoor cameras that integrate the same PMD ToF sensor chip but with stronger illumination units which achieve a working range of 35m in adverse/bright ambient light situations.[1]

Two different options to visualize the ToF data are shown in Fig. 1. The visualization is streamed to the vehicle's dashboard screen. Especially the top-down view of the point cloud, after removing the drivable area, is highly capable of supporting the driver in parking tasks. This perspective enables the driver to estimate distances around the car intuitively.

[1] O3M 3D sensor for mobile applications (https://www.ifm.com/at/en)

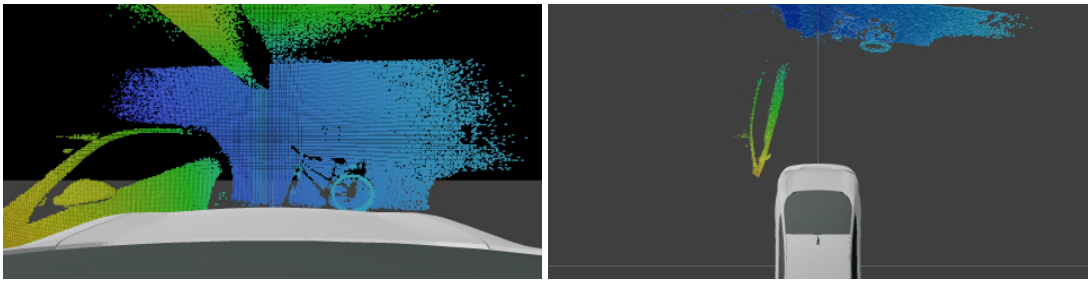


Fig. 1. Visualization of ToF data during a parking task, viewed from different perspectives.

5. Conclusion

ToF cameras provide precise, high frame-rate 3D data of the environment. This data can be used to obtain a detailed 3D model of the surroundings. We mounted a ToF camera on a passenger vehicle and implemented software, capable of streaming the point cloud of the surrounding environment to the dashboard and visualizing it to the driver. Considering the results of this work, we conclude that ToF cameras are feasible to be used in parking assistance systems. A promising scenario is the combination of a top-down surround-view video stream with the 3D data from multiple ToF sensors. Augmenting the top-down 2D color image with the 3D points from the ToF cameras could provide the driver with detailed and easily-interpretable visual information about the car's surroundings.

Acknowledgements

The authors of this paper would like to thank the Electronic Component Systems for European Leadership Joint Undertaking for funding the *IoSense: Flexible FE/BE Sensor Pilot Line for the Internet of Everything* project with the number 692480.

References

[1] R. Austin, "Fatalities and injuries in motor vehicle backing crashes," National Highway Traffic Safety Administration, Tech. Rep., 2008.

[2] D. Demirdjian and C. Varri, "Driver pose estimation with 3d time-of-flight sensor," in 2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems, March 2009.

[3] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in 2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), May 2015.

[4] U. Scheunert, B. Fardi, N. Mattern, G. Wanielik, and N. Keppeler, "Free space determination for parking slots using a 3D PMD sensor," 2007 IEEE Intelligent Vehicles Symposium, 2007.

[5] O. Gallo, R. Manduchi, and A. Rafii, "Robust curb and ramp detection for safe parking using the Canesta TOF camera," 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Work-shops, CVPR Workshops, 2008.

[6] T. Ringbeck, T. Möller, and B. Hagebeuker, "Multidimensional measurement by using 3-D PMD sensors," Advances In Radio Science, 2007.

[7] C. Wang, H. Zhang, M. Yang, X. Wang, L. Ye, and C. Guo, "Automatic parking based on a bird's eye view vision system," Advances in Mechanical Engineering, 2014.

[8] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," IEEE Journal of Quantum Electronics, March 2001.

Occupancy Grid Fusion of Low-Level Radar and Time-of-Flight Sensor Data

Josef Steinbaeck*†, Christian Steger†, Eugen Brenner†, Gerald Holweg*, and Norbert Druml*
*Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, gerald.holweg, norbert.druml}@infineon.com
†Graz University of Technology, Graz, Austria
{steger, brenner}@tugraz.at

***Abstract*—We present an approach to fuse radar and *time-of-flight* (ToF) range sensor data into an occupancy grid. Fusing the low-level data at sensor level prevents the loss of precious information during compression and pre-processing. Constructing the low-level occupancy grid from raw sensor data enables the detection of occupied cells which are not clearly visible by any of the single sensors. Fusion of the heterogeneous sensor data enhances the perception quality since single sensors fail in certain conditions. Thus, the fusion at low-level holds a high potential to enhance the perception quality for automotive/robotic applications.**

We demonstrate our approach with real-world data from a mobile sensor platform with three ToF cameras and a 77 GHz high-resolution radar sensor. An occupancy grid is created whenever synchronized sensor data from all sensors is available. The proposed method performed successful detection of multiple pedestrians in different test scenarios. Our approach to build an occupancy grid from radar and optical range sensors can be used as a base in various short-range perception applications (e.g., in robotics or mobile devices).

***Index Terms*—radar, time-of-flight, robotics, sensor fusion**

I. Introduction

Occupancy grids are often used as a base for map creation or to estimate dynamic objects in a scene. However, the quality of the representation depends on the quality of the underlying sensor data. Single sensors are often not capable to work under any circumstances and cannot guarantee their functionality. Thus, multiple heterogeneous sensors are used to mitigate weaknesses of single sensors. A prominent approach to increase the representation quality is to fuse multiple sensors.

Depth sensors are a popular choice to perform environmental perception in the robotic/automotive context. Lidar is superior to most other sensors in terms of precision, resolution, speed, and range. However, lidar sensors struggle in various conditions like rain, fog, snow, or dust, because the light pulse is not capable to pass through these small particles. Radar sensors are inferior to lidar in multiple aspects, like the angular resolution. However, the mm-waves from radar are capable to pass through small particles. Radar still performs valid measurements in adverse weather conditions like rain, fog or snow. Thus, most safety conscious environmental perception platforms have both: radar and lidar. But these two sensors have to be fused in order to get most of the provided environmental information. In clear weather conditions, a system might be fully confident using the lidar unit only. In more difficult weather conditions, a radar sensor (together with other sensors) can be crucial for the same system to ensure functionality.

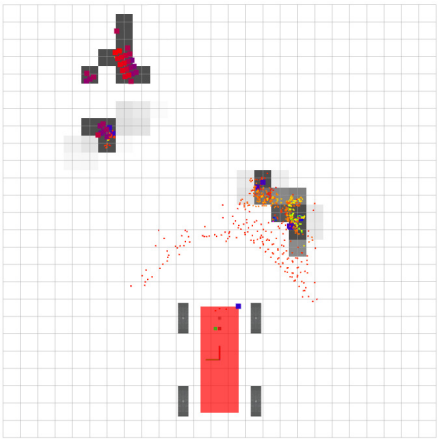


Fig. 1. An example of a resulting occupancy grid after fusing radar and ToF sensor data. The grid is visualized together with the underlying ToF and radar data using the ROS visualization tool *RViz*.

Different approaches to fuse multiple sensors have already been presented to literature. Popular approaches are lidar and radar fusion [1], radar and camera fusion [2], or lidar and camera fusion [3]. Similar sensor fusion concepts are already implemented in many of today's research vehicles and robots.

ToF cameras (indirect range imaging) have not been investigated sufficiently for sensor fusion systems. These low-cost cameras are capable of capturing objects in the scene very precisely, at a high rate and with low processing effort. In contrast to radar or direct ToF (e.g., lidar), ToF cameras are rather designed for short-range applications. Due to the related working principle, using ToF exclusively results in similar vulnerabilities as lidar. Fusing the ToF sensor data with radar has a high potential to significantly increase the perception reliability in the short-range.

In this paper we present an approach to fuse ToF and radar range data. We show an attempt to fuse low-level radar and ToF data utilizing an occupancy grid. This fills one of the gaps

in scientific literature regarding the fusion of ToF and radar. To summarize, the contributions of this paper to the scientific community are:

- An approach to fuse ToF and radar sensor data for close-proximity perception.
- A 2D occupancy map implementation using the *Robot Operating System* (ROS).
- A novel update approach for occupancy maps using sensor data from multiple sensors.

Section II gives a short overview of related approaches in research and industry, and highlights similarities to our approach. The sensors used for the work in this paper are presented in Section III as well as the structure of the provided raw sensor data. Additionally, we present the pre-processing tasks of each sensor and a way to fuse the data into an occupancy grid. Section IV shows resulting occupancy grids from real-world measurement data. Finally, in Section V we present a short conclusion of the work.

II. Related Work

Occupancy grids where first introduced by Moravec et. al. [4] in 1985, where they represented sonar data in an occupancy grid. Occupancy grids have been popular since quite some time and where first namely used for robotic navigation in [5]. The authors of [6] evaluated the performance of a navigation task using occupancy grid fusing. They created occupancy grids using different variations of sensors and compared their performance.

There also exist approaches where radar and lidar data is fused into an occupancy grid. Garcia et al. [7] present an approach to fuse a long range radar with a laser scanner mounted on a truck. The work presented in [8] shows an approach to build an occupancy grid map using high-resolution radar sensors. They use a radar sensor model to insert the radar point into the occupancy grid and show different pre-processing techniques for the raw data. The authors performed experiments where the detected free space matches the free space in the real world scene.

To the best of our knowledge, the fusion of synchronous radar and ToF sensor data has not been sufficiently addressed by the scientific community yet. This is why we fill this gap, by presenting an approach to perform occupancy mapping for short-range perception using radar and ToF range sensors.

III. Fusion of Low-Level Radar and Time-of-Flight 3D Sensor Data

We utilize the sensor platform presented in [9] to obtain the data from the different sensors. The sensors are synchronized via an external trigger signal in order to have the measurement data temporally aligned. The heterogeneous sensor data is then pre-processed and transferred into a common coordinate system. Afterwards, the data is fused into an occupancy grid, where each cell holds a value for the probability of occupancy. An overview of the processing steps is depicted in Fig. 2.

A. Time-of-Flight Camera

We use three *CamBoard pico monstar*[1] ToF cameras in our setup. These cameras utilize special *photonic mixer device* (PMD) pixels in order to determine the phase difference between transmitted and received modulated infrared light. A distance and an amplitude value is calculated for every pixel in the scene, by the application of the *four-phase algorithm* [10]. Using the camera parameters, the pixel position and the corresponding distance value, it is possible to calculate a 3D point for every pixel. The resulting point cloud in the 3D space consists of points from all pixels in the image plane. The corresponding amplitude value of the pixels can be utilized as a confidence value.

The used ToF cameras can be configured to run at up to 60 frames per second and provide 352×287 pixels. The cameras are horizontally aligned and rotated by an angle of 45° to each other. Since each camera has a field-ov-view of $100^\circ\times85^\circ$, there are overlapping areas in the captured scene. The illumination time and the frame rate can be configured during runtime in order to adapt to the use case and the reflectivity of the scene. We use the cameras in *slave-mode* where a new measurement is performed each time an external trigger signal changes its level to high. USB 3.0 is used to receive data from the cameras as well as to supply them with power. The cameras come with a software library called *royale*, providing functionality to receive the raw data from the cameras and to perform certain pre-processing tasks (e.g., error corrections) as well as the point-cloud calculation.

To summarize, the output of each of the ToF camera modules is a point-cloud containing about 100k points (Fig. 3). Each point is characterized by the three cartesian coordinates (x/y/z), a confidence value, an intensity value and a noise value.

As seen in the data processing flow chart (Fig. 2), the ToF data of every camera is first pre-processed before it is merged with the other point clouds. Pre-processing consists of several error correction steps, down-sampling of the point cloud and transformation into a common coordinate frame. Since the positions and the alignment of the sensors is known, the transformation can be directly applied to every point. After the pre-processed point clouds from the three ToF cameras are ready, they are merged into a single point cloud (Fig. 4). The merged point cloud still holds a confidence value, an intensity value and a noise value for every point.

B. Radar Sensor

We use one radar sensor, a *RadarLog*[2] in our setup. The radar sensor comes with four transmit and 16 receive antennas and is capable to perform *frequency modulated continuous wave* (FMCW) measurements. In our setup, the radar is used with sequences of fast chirps in order to enable the estimation of range, velocity and angle. Since the radar sensor does not

[1]CamBoard pico monstar: 3D imaging development kit of the pico family (https://pmdtec.com/picofamily).

[2]RadarLog: A platform for microwave radar data capturing and logging (http://www.inras.at).

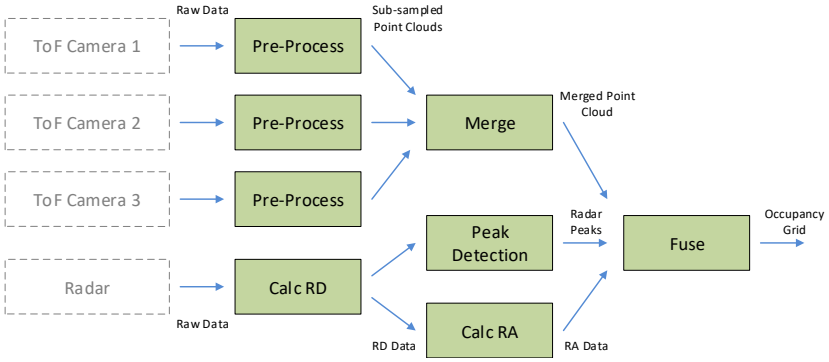


Fig. 2. Processing flow of the low-level occupancy grid creation using sensor data from three ToF cameras and one radar sensor.

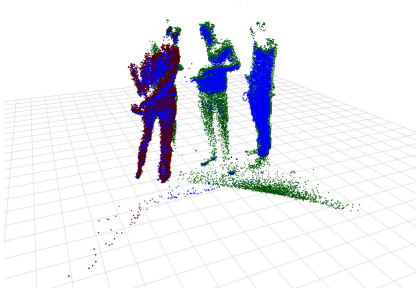


Fig. 3. The full point clouds of the three ToF cameras, transformed with individual coordinate frames. Every point holds a value for x, y, z, intensity, confidence and noise.

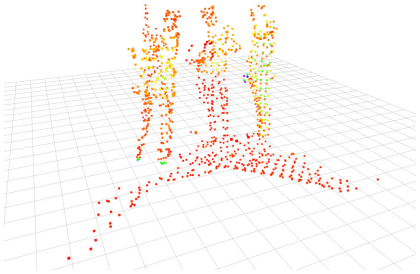


Fig. 4. The down-sampled and merged point cloud from the three ToF cameras. The color of the points depends on the respective intensity. Every point comes with a (x/y/z) value as well as a value for intensity, noise and confidence.

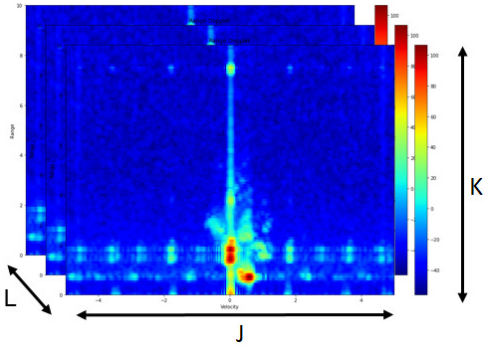


Fig. 5. L range-Doppler images after 2D FFT with a dimension of J×K bins per image. Peaks in this image correspond to a radar reflection of an object at a certain range moving at a certain speed.

provide any elevation information, the targets are mapped to the sensor's mounting height without any vertical information. The radar development kit is mounted on the platform in front-viewing direction.

The radar transmits sequences of FMCW frequency chirps and receives delayed responses of these sequences depending on the objects in the scene. The transmitted and the received signal are then mixed and low-pass filtered in order to obtain the so-called *IF signal* without any high-frequency components. The *IF signal* of each receiving antenna is sampled and converted to digital for further processing. This sampled data is then processed by software in order to obtain information of objects in the scene.

Every digitized measurement contains N samples for M ramps and L antennas. Two FFTs are calculated in order to generate the range-Doppler image of the objects in the scene (see Fig. 5). One range-Doppler image is generated for each of the receive antennas. Thus, after the second FFT the output are L range-Doppler images with a dimension of J×K bins each, depending on the FFT bins of the first two FFTs.

A third FFT is calculated over the L antennas with K bins to obtain angular information. The output is a 3D cube with the dimension I×J×K containing range, velocity and angle information. A possible representation are J range-angle images after the third FFT with a dimension of I×K bins per image (see Fig.6).

We are interested in a single range-angle image containing all velocities, which can be directly projected into the scene. Our approach to generate an overall range-angle image $\mathrm{I}(i,k)$

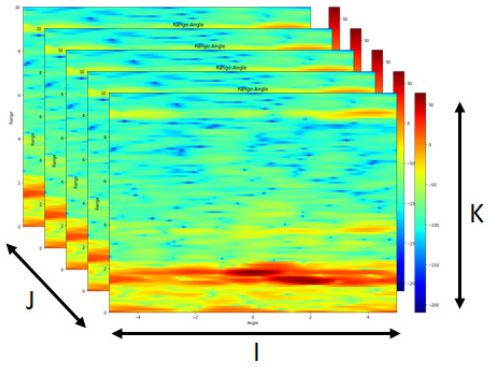


Fig. 6. J range-angle images after the third FFT with a dimension of I×K bins per image. There is one range-angle image for every velocity bin.

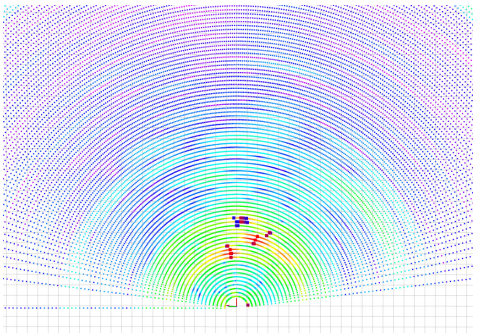


Fig. 7. Visualization of the radar range-angle map and the detected radar peaks. Every bin of the range-angle image is projected into the corresponding 2D space (top-down) for this bin. The color corresponds to the magnitude of the radar reflection.

is to take the maximum values of the J range-angle images $\mathrm{I_j}(i,k)$.

$$\mathrm{I}(i,k) = \max\{\mathrm{I_j}(i,k)\} \tag{1}$$

The overall range-angle image can be mapped into the scene as seen in Fig. 7. A peak in this image indicates whether there is an object present at the corresponding position in the scene. However, the overall range-angle image also includes all static targets since it is composed without any velocity preselection. Strong reflections cause side lobes in the range-angle image, making it difficult to detect multiple targets at the same range. Thus, the overall-range angle representation is rather feasible for visualization purposes than to detect targets.

Another option is to perform a peak detection in the range-Doppler image. We use a *constant false alarm rate* (CFAR) algorithm to detect significant points in the range-Doppler image. The angle of these single peaks is then calculated using a beam-forming algorithm (Bartlett beamformer). Eventually a list of peaks with a (range/angle/velocity) value for every point is obtained. Knowing the alignment of the sensor, this list of peaks can be transformed to (x/y/velocity) values for every point and visualized into a common 2D coordinate space. Detected peaks are shown in Fig. 7, together with the range-angle map.

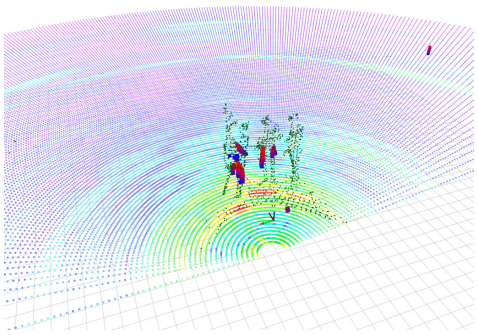


Fig. 8. Visualization of the sensor data before the fusion. It contains of the down-sampled and merged ToF point cloud, the detected radar peaks and the radar range-angle map.

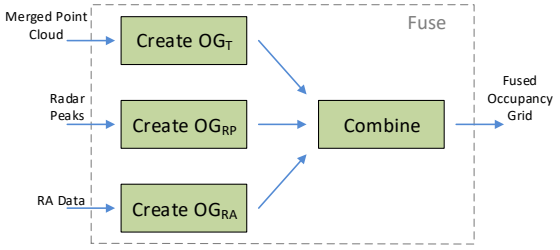


Fig. 9. Detailed data flow diagram of the fusion block. We construct three preliminary occupancy grids which are then fused to an overall occupancy grid.

C. Occupancy Grid Fusion

After the data from the different sensors is pre-processed and ready, the data fusion is initiated. At that point, the sensor data is already available in a common coordinate space as seen in Fig. 8. The occupancy grid fusion is triggered every time new pre-processed data from all sensors is available. We present an approach to fuse the merged ToF point cloud, the radar range-angle map and the radar peak list into a common occupancy grid.

First three individual occupancy grids are created using the three data-streams (merged ToF point cloud, radar range-angle map and radar peak list). In a final step, the individual grids are fused into an overall grid (see Fig. 9). The occupancy grids are represented as 2D top-down grids covering an area of interest in front of a robot/vehicle. Every cell contains an occupation probability value between zero and 100. The width, height and resolution of the occupancy grid are customizable. For the experiments in this paper we set the width to 10 m, the height to 16 m and the resolution to 0.25 m.

1) Time-of-Flight: For the ToF sensor, the point cloud is inserted into the occupancy grid. The ToF point cloud is

cropped to an area of interest. Points below and above a certain z value are not included into the occupancy grid. Also points below and above a certain range are discarded. All remaining points are inserted into the occupancy grid at their corresponding (x/y) position. Each point's confidence value is scaled to the range between zero and 100. This normed value is then used to calculate the occupation probability. If the normed confidence value of the point is higher than the cell's probability the cell's probability is updated. The probability of an occupancy grid cell $OG_T(i,j)$ is updated using the normed ToF confidence $conf$ as follows:

$$OG_T(i,j) = \begin{cases} conf & \text{, if } conf > OG_T(i,j), \\ OG_T(i,j) & \text{, otherwise.} \end{cases} \quad (2)$$

2) Radar peaks: The list of radar peaks is used to create another occupancy grid. Radar peaks are only considered if they are above a certain range and within the limits of the grid. Since the peak detection is already performing a thresholding in the CFAR detection, every detected peak is assumed to be a target. Thus, the probability of the occupancy grid cells is updated with every radar peak as seen in Equation 3.

$$OG_{RP}(i,j) = \begin{cases} 100 & \text{, if peak at } (i,j), \\ OG_{RP}(i,j) & \text{, otherwise.} \end{cases} \quad (3)$$

3) Radar range-angle: The third step is to create an occupancy grid from the radar's overall range-angle image. The reflected magnitudes are compared to a distance-dependent reference value in order to extract potential targets. As first step a confidence value is calculated for every range-angle value. The confidence values are determined by the ratio of the reference value and the magnitude of the range-angle image. The occupancy grid's probability value $OG_{RA}(i,j)$ is then updated as seen in Equation 5.

$$conf = \begin{cases} 0 & \text{, if } 20\log(\frac{mag}{ref}) < 0, \\ 100 & \text{, if } 20\log(\frac{mag}{ref}) > 100, \\ 20\log(\frac{mag}{ref}) & \text{, otherwise.} \end{cases} \quad (4)$$

$$OG_{RA}(i,j) = \begin{cases} conf & \text{, if } conf > OG_R(i,j), \\ OG_{RA}(i,j) & \text{, otherwise.} \end{cases} \quad (5)$$

4) Fused Occupancy Grid: The final occupancy grid is constructed by fusing the three pre-computed occupancy grids (Equation 6). Clearly identified targets from the OG_T and OG_{RP} are modeled to directly effect the final occupancy grid.

$$OG(i,j) = \max \left\{ \begin{array}{l} OG_T(i,j), \\ OG_{RP}(i,j), \\ \frac{OG_T + OG_{RP} + OG_{RA}}{3} \end{array} \right\} \quad (6)$$

Fig. 10. RGB image of the test scene captured with the fisheye camera. Two pedestrians are standing close to the platform and two others are further away.

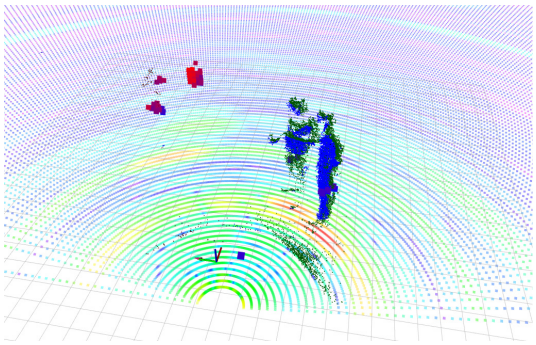


Fig. 11. Visualization of the raw data from the ToF and radar sensors. The two closer pedestrians are identified by ToF point clouds, while the other two pedestrians are captured by radar.

IV. Results

In this section we demonstrate the performance of our approach within a certain test scenario. For the scenario, the fused low-level occupancy grid is compared to the occupancy grid of the single sensors. In order to give humans a reference image of the test scene, the sensor platform is also equipped with a fisheye camera. Fig. 10 shows the fisheye camera image of the test scene. The full ToF point cloud, the projected radar range-angle map and the detected radar peaks are visualized in Fig. 11. The ToF camera (in the used configuration) is able to capture the two closer persons in detail. The radar peak detection clearly identifies the two other pedestrians. The radar peaks are more pronounced in the range-Doppler map since these two persons were moving.

The individual occupancy grids are created from the already pre-processed sensor data. The ToF-only occupancy grid output of the three ToF cameras is presented in Fig. 12. As clearly visible, the points of the ground floor are not considered in the final occupancy grid. Fig. 13 shows the occupancy grid generated from the radar peak data only. As seen in the figure, the occupancy grid captures all four pedestrians, also the two further away from the platform. Fig. 14 shows the occupancy grid generated from the radar range-angle map only. This grid is constructed from the range-angle map.

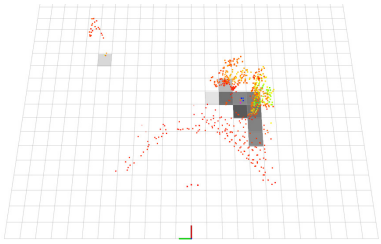


Fig. 12. Occupancy grid of ToF data only. The ground floor is not included into the grid. The two closer pedestrians are clearly detected while the third is not marked as fully occupied.

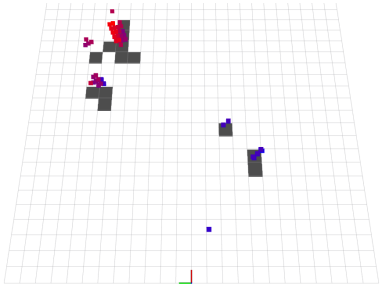


Fig. 13. Occupancy grid of the radar peaks array. The peak too close to the origin is discarded since it is below the minimum range.

The fusion of the three preliminary occupancy grid maps results in the final occupancy grid shown in Fig. 15. The figure also shows the down-sampled point cloud and the detected radar peaks as a reference. All four pedestrians are detected and marked as occupied cells in the fused occupancy grid. The representation of obstacles located close to the sensors might be incomplete since their boundaries are not fully captured. However, in most applications the knowledge of an existing close obstacle is sufficient to react accordingly.

V. Conclusion

We presented an approach to fuse heterogeneous range-sensor data into a common occupancy grid. The grid is constructed from the sensor data of multiple synchronized sensors. After individual pre-processing for each sensor, three independent occupancy grids are created. A fused occupancy grid is then generated based on the individual grids.

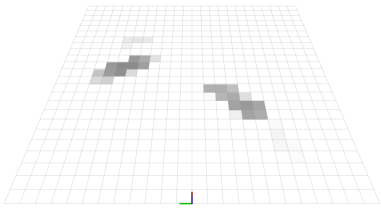


Fig. 14. Occupancy grid created with the data from the radar range-angle map.

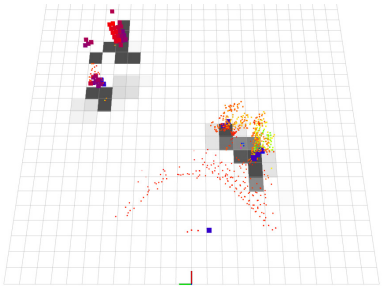


Fig. 15. The resulting occupancy grid after fusing the three individual occupancy grids.

The fused occupancy grid has a higher expressiveness than the individual occupancy grids. As shown in the results section, the additional data from the second sensor provides valuable information for modeling the environment. Obstacles, not detected by solely radar or ToF are more likely to be detected by the fusion of these two sensors. Thus, the low-level fusion at sensor-level provides a significant advantage compared to the fusion of high-level sensor data. Since the memory consumption of the occupancy grid is low compared to the raw data, the grid is suitable to be transferred via low-speed buses. Our approach can be adopted for various applications (e.g., detection of obstacles on a mobile device).

Acknowledgments

The authors of this paper would like to thank the Austrian Research Promotion Agency for funding the *Autonomous CarTo Infrastructure communication mastering adVerse Environments* (ACTIVE) project with the number 855010.

References

[1] D. Göhring, M. Wang, M. Schnürmacher, and T. Ganjineh, "Radar/lidar sensor fusion for car-following on highways," in *The 5th International Conference on Automation, Robotics and Applications*, Dec. 2011, pp. 407–412.

[2] K. Kim, C. Lee, D. Pae, and M. Lim, "Sensor fusion for vehicle tracking with camera and radar sensor," in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, Oct. 2017, pp. 1075–1077.

[3] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system," *Electronics*, vol. 7, no. 6, 2018.

[4] H. P. Moravec and A. Elfes, "High resolution maps from wide angle sonar," vol. 2, Apr. 1985, pp. 116–121.

[5] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, Sep. 2003.

[6] P. Stepan, M. Kulich, and L. Preucil, "Robust data fusion with occupancy grid," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 1, pp. 106–115, Feb. 2005.

[7] "High level sensor data fusion for automotive applications using occupancy grids," *2008 10th International Conference on Control, Automation, Robotics and Vision, ICARCV 2008*, no. Dec., pp. 530–535, 2008.

[8] M. Li, Z. Feng, M. Stolz, M. Kunert, R. Henze, and F. Küçükay, "High resolution radar-based occupancy grid mapping and free space detection," Mar. 2018, pp. 70–81.

[9] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Design of a low-level radar and time-of-flight sensor fusion framework," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Aug. 2018, pp. 268–275.

[10] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, Mar. 2001.

ACTIVE - Autonomous Car to Infrastructure Communication Mastering Adverse Environments

Josef Steinbaeck[1,4], Norbert Druml[1], Thomas Herndl[1], Stefan Loigge[2], Nadja Marko[2], Markus Postl[2], Georg Kail[3], Reinhard Hladik[3], Gerhard Hechenberger[4], Herbert Fuereder[4], Christian Steger[5], Eugen Brenner[5], and Christian Schwarzl[2]
[1]Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, norbert.druml, thomas.herndl}@infineon.com
[2]Virtual Vehicle Research Center, Graz, Austria
{stefan.loigge, nadja.marko, markus.postl, christian.schwarzl}@v2c2.at
[3]Siemens AG, Vienna, Austria
{georg.kail, reinhard.hladik}@siemens.com
[4]Siemens Mobility GmbH, Vienna, Austria
{gerhard.hechenberger, gerhard.fuereder}@siemens.com
[5]Graz University of Technology, Graz, Austria
{steger, brenner}@tugraz.at

***Abstract*—Precise localization is crucial for autonomous navigation, especially for autonomous driving. GNSS localization is prone to a number of errors and is not sufficient to provide reliable positional data in all situations. Most existing approaches for fine-grained positioning are not working reliably in difficult weather conditions. In this paper we present a method to tackle that problem by performing precise localization by exploiting the angle-of-arrival of V2X communications.**

During a 30-months project, we built an unmanned vehicle capable of determining its precise location via V2X communication. In order to safely navigate in the environment and detect obstacles in its path, the robot is also equipped with environmental perception sensors (time-of-flight and radar). We evaluated the proposed localization method during a test-drive on a precisely mapped parking lot. The resulting localization precision was improved by over 60 percent compared to the standard GPS localization.

***Index Terms*—radar, automotive sensors, time-of-flight, autonomous vehicles, sensor fusion**

I. Introduction

An accurate location estimation is crucial for most autonomous applications (robotic/automotive) to perform the navigation task. Today, localization is mainly performed using a *Global Navigation Satellite System* (GNSS) like the *Global Positioning System* (GPS) or Galileo. The quality of the localization accuracy strongly depends on external factors (e.g., weather/atmospheric influences) and the surroundings. Especially in city scenarios with tall buildings, walls and no direct line of sight to GNSS satellites, highly accurate localization is not reliably possible. Regular GPS can only achieve an accuracy of up to $\pm 3\,\mathrm{m}$. Adding a fixed reference station improves the accuracy to about $\pm 0.1\,\mathrm{m}$ for a limited geographical location. This approach is very feasible for agricultural applications and commonly used in this sector.

Most autonomous systems of today use GNSS only for a coarse localization since the reliability is not sufficient for precise driving tasks (e.g., multi-lane sections or narrow roads). Environmental perception sensors are utilized together with a detailed map of the environment to obtain a precise location estimation. Particle filters are a popular tool to match the sensor measurements with the structures of a map in order to precisely estimate the location within that map. However, for this approach to work robustly, the detailed map has to be updated constantly, a difficult task in permanently changing environments. Thus, the map features have to be as invariant to changes as possible. Ground penetrating radar is a sophisticated approach, where a radar is pointed to the ground in order to localize the vehicle [1]. This localization method is robust, since the ground material is typically independent to weather, light and seasonal changes. But the method also

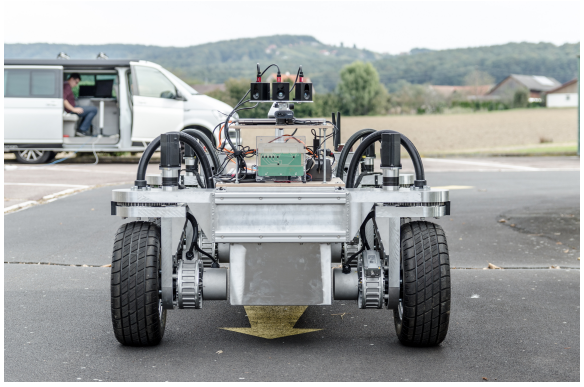


Fig. 1. The SPIDER, an unmanned robot platform with four individually controllable wheels. The *Robot Operating System* (ROS) framework is used to control the robot. An environmental perception platform with multiple sensors is mounted at the top of the robot as well as an OBU for V2X communication.

requires detailed radar maps of the ground which are often not available.

Vehicle-to-infrastructure (V2I) and *vehicle-to-everything* (V2X) communication are emerging technologies of today. *Road side units* (RSU) provide the functionality to inform vehicles in the proximity with supportive data (e.g., informations about a complex road scenario). This can for example be the structure of lanes in a crossing section with information about each lane's flow directions. There exist approaches where RSUs are extended in their functionality to also provide localization information to the vehicles. Since the required communication protocol is already available and standardized, these techniques hold a high potential to support vehicle localization. The works presented in [2] and [3] show approaches where vehicle localization is performed using RSUs. Both use time-of-arrival/time-difference methods as well as dead-reckoning to determine the vehicle's location on the road without the use of GPS. Fascista et. al. [4] show a GPS-free localization method which exploits the *angle of arrival* (AOA), determined using an *uniform linear array* (ULA) of antennas. They use the *multiple signal classification* (MUSIC) algorithm to estimate the angle of the vehicle to the RSU [5].

The *Robot Operating System* (ROS) is a commonly used tool among researchers to build robot applications. ROS [6] is a popular choice for fast prototyping, since it provides a set of ready-to-use open source libraries and tools for robotics. In [7], the authors present an automated driving research platform which utilizes Autoware, a ROS-based framework.

In this paper we focus on exact localization using a single RSU in combination with GNSS. The RSU can send/receive information to/from the vehicle, while also detecting the AOA. Using the inaccurate GNSS from the vehicle, the exact location of the RSU and the obtained angular information, a precise localization is possible. This localization method works even in difficult environment situations like snow, rain or fog where vision based systems tend to fail. We built an unmanned robot (see Fig. 1) in order to demonstrate precise localization using an RSU with angular detection. As a safety feature, the vehicle is also equipped with state-of-the-art perception sensors in order detect obstacles in the vehicle's path.

To summarize, the contributions of this paper to the scientific community are:

- The electro-mechnanical construction of an unmanned research vehicle using state-of-the-art automotive components for electrical cars.
- A ROS architecture for organizing the robot and the data flow between the sensors and actuators.
- A novel method for vehicle localization with V2X infrastructure.
- A sensor fusion approach using radar and optical range information (ToF) for obstacle detection.

The Section II is divided into three subsections. First, we describe the construction and the composition of the unmanned research vehicle. Then, we present a detailed description of the localization using an extended RSU. Eventually, we introduce the environmental perception module of the platform. Real-world measurements with the vehicle are presented in Section III, while Section IV gives a short conclusion of the work.

II. Autonomous Car to Infrastructure Communication Mastering Adverse Environments

The goal of the work was to build an unmanned robot platform, capable of precise localization using infrastructure support. We built a prototype of an extended RSU, capable of measuring signal directions to achieve that. The robot is additionally equipped with environmental perception sensors in order to perform basic obstacle detection.

A. SPIDER: An Unmanned Research Vehicle

The *Smart PhysIcal Demonstration and Evaluation Robot* (SPIDER) is an unmanned, electrical vehicle built from scratch. The constructed robot chassis consists of a robust aluminum frame with a wheelbase of 1.2 m. Each of the four robot wheels comes with an electrical drive as well as a steering engine. Thus, the platform can change its heading while remaining in the same location.

The SPIDER robot is equipped with multiple electronic components. The chassis holds enough space for the battery, drive-controllers and the processing unit in the waterproof interior of the robot. The robot is equipped with an *on-board-unit* (OBU, *Cohda Wireless*) for V2X communication, a WiFi antenna in order to communicate with the on-board processing unit and a GPS module for coarse location estimation. Additionally, the robot has a mounting point for an environmental perception platform on top of the robot. The robot is also equipped with an emergency-stop button on its rear end, which can stop all operations and apply the brakes. Additionally, the platform can be stopped manually with a remote emergency break connected via an active-on connection.

The platform provides two different control functions: automatic control and manual control. In manual mode the platform is controlled using an operator panel (e.g., a video game controller). The communication between between these two participants is encrypted via a session key, exchanged using public key cryptography. In automatic mode, the platform is capable of driving a pre-defined path. The robot calculates the resulting controls and sends them to the actuators.

For the high-level control, the robot is equipped with an x86-based processing unit running Linux and ROS. The computer is located in the inside of the waterproof chassis and is powered by the robot's battery. The computer has to handle requests from the operator panel, from the sensors and to perform the respective processing. The processing unit is capable of logging the sensor data of the robot via the ROS tool *rosbag*. For low-level control, the robot is equipped with an automotive state-of-the-art, safety-focused microcontroller (Infineon's AURIX). The microcontroller communicates with the motor/steering controller of the four wheels and handles the battery/charging management.

B. Precise Vehicle Localization Using V2X Infrastructure

V2I systems typically consist of an OBU and an RSU. The OBU is a device mounted on the driving vehicle capable to send data to infrastructure using wireless communication (IEEE 802.11p standard). The RSU is an element of fixed-location in the infrastructure capable of receiving and transmitting messages to/from an OBU. An RSU sends the state and timing information of upcoming traffic lights via *signal phase and timing* (SPAT) messages to the vehicle. Information about the topology of the crossing section is exchanged via MAP messages. Siemens offers standardized products (RSUs) capable of transmitting and receiving V2I messages. One method to support the vehicle localization with an RSU is to calculate an error-correction vector in the location-fixed RSU and send this vector to the vehicle, so the vehicle can correct the GPS error. Another method is to exploit the receiving angle of the messages from the vehicles in order to support the localization. In this section we combine these two methods and extend a standard RSU with special hardware in order to enable precise localization.

1) Extended RSU: We built a prototype capable of receiving V2X messages and determining the angle to the vehicle. This prototype can be used with standardized V2X infrastructure and detect the angle to the RSU for precise localization. The Siemens RSU is extended with the following components in order to provide precise localization.

- **Antenna array** with four antennas, capable of receiving the wireless communication signals (802.11p) from the OBU.
- **Software defined radio** to output the four-channel IQ data to the *field-programmable gate array* (FPGA).
- **Xilinx FPGA** to provide the four-channel IQ data as well as the GPS location to a PC.
- **x86 PC** running Linux for the AOA estimation and the location calculation.

Fig. 2 shows the components of the V2X localization as arranged for the test drive. The angle accuracy of the prototype was evaluated in prior to the test drive in a laboratory environment. The deviation between the real and the measured angle was below $\pm 2°$ for all measured angles.

2) Processing: The flow-diagram of the processing on the prototype is shown in Fig. 3. The OBU of the SPIDER robot sends *cooperative awareness messages* (CAM) to the RSU. The CAM messages contain the location received from the GPS receiver of the vehicle. These CAM messages are also received by our prototype with an ULA with four channels. After decoding, the GPS and the MAC information are available additionally to the raw data from the four channels. The MUSIC algorithm is applied to the raw data in order to obtain the angle between the OBU and the RSU. The *location calculation* module calculates the offset between the vehicle's GPS location and its estimated real location. First it transfers the GPS coordinate into an own coordinate frame (x,y). A Kalman filter is then used to estimate the offset to the real location using the GPS data and the AOA. The offset is then sent back to the vehicle via a SPAT message. The vehicle can then use this information to correct its current location.

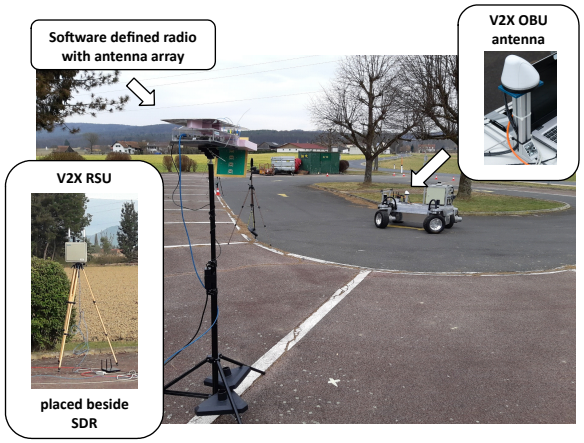


Fig. 2. Components of the V2X localization: *On Board Unit* (OBU), *Software Defined Radio* (SDR) and *Road Side Unit* (RSU).

C. Environmental Perception Platform for Obstacle Avoidance

The perception platform is equipped with multiple ToF-cameras and a radar sensor mounted in front-facing direction. A battery allows the independent mobile operation of the platform. Fig. 4 shows a picture of the platform mounted on the SPIDER.

1) Deployed sensors: Three *CamBoard pico monstar*[1] ToF cameras are mounted on the platform. ToF cameras use an active infrared illumination in combination with special *photonic mixing device* (PMD) pixels in order to obtain a distance image of the scene [8]. In addition to the distance value, an amplitude value can be determined for every pixel in the scene. The CamBoard pico monstar comes with a resolution of 352×287 pixels and runs with up to 60 frames per second. Compared to other range sensors, ToF cameras are compact, inexpensive and only introduce a minimal processing overhead. A point cloud of the scene can be calculated from the distance image using the camera properties and the lens parameters. To summarize, the output of each of the ToF cameras is a point-cloud containing about 100k points. Each point is characterized by the three cartesian coordinates (x,y,z), a confidence value, an intensity value and a noise value.

We use the *RadarLog*[2] radar development kit on our platform. This kit comes with a 77 GHz radar frontend with 16 receive and four transmit antennas. The RF frontend of the radar is capable of transmitting fast chirped *frequency modulated continuous wave* (FMCW) signals. Reflections of the mm-waves in the scene are received by the antennas, mixed, low pass filtered and converted to digital. After certain

[1] CamBoard pico monstar: 3D imaging development kit of the pico family (https://pmdtec.com/picofamily/).

[2] RadarLog: A platform for microwave radar data capturing and logging (http://www.inras.at).

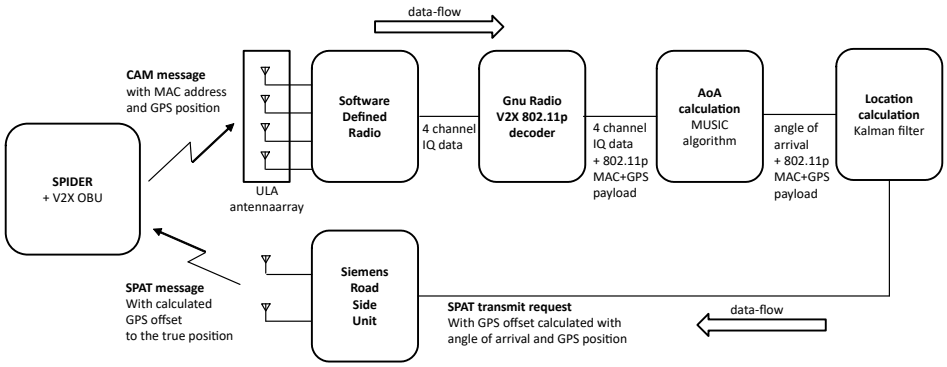


Fig. 3. Signal flow of the V2X localization using the extended RSU.

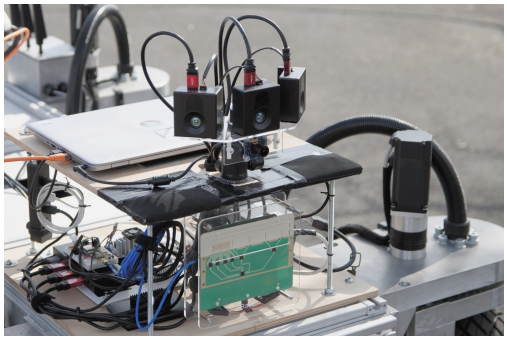


Fig. 4. The environmental perception platform consists of three ToF cameras, one radar and one fisheye camera.

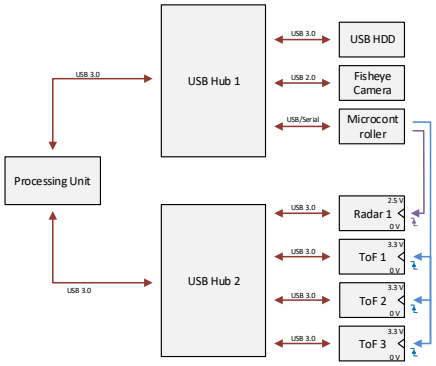


Fig. 5. The signal flow of the environmental perception platform. Two USB hubs are used to connect all sensors to a single processing unit.

signal processing steps, the range, velocity and angle to targets in the scene are obtained. The deployed radar sensor has a horizontal beamwidth of 76.5° and a vertical beamwidth of 12.8°.

2) Mobile sensing platform: The sensors are attached to a platform (Fig. 4) which can be directly mounted on the SPIDER robot. The platform comes with an individual power supply (lithium battery) to allow an independent operation. All sensors are mounted in front-facing direction. The three cameras are mounted with overlapping field-of-views in order to provide redundancy in the area in front of the vehicle. Fig. 5 shows an overview of the signal flow between the single components of the system. The processing unit is connected to the sensors via USB as well as to an USB HDD in order to log the raw sensor data.

All sensors are integrated into the platform via the ROS framework. ROS nodes are used to receive the raw data from the sensors and to perform individual (pre-)processing. After processing, the data is fused into a common representation, an occupancy grid. The occupancy grid is then transferred to the SPIDER's processing module and consulted for obstacle avoidance. A detailed description of the occupancy grid creation can be found in [9].

3) Temporal and spatial synchronization: The sensors are synchronized using external trigger signals for the ToF cameras as well as for the radar sensor. These external signals are generated on a dedicated microcontroller. The trigger sequence ensures that the radar and the ToF sensors capture the same scene. The ToF cameras are triggered sequentially with a short delay, in order to not influence each other. The radar measurement is triggered at the same time as the first ToF sensor.

Since each of the sensors operates in their own frame, the data also has to be spatially aligned into a common coordinate frame. The transformations between the single ToF cameras can be implicitly obtained by aligning the overlapping regions of the point clouds. This is done by applying the *iterative closest point* (ICP) algorithm to the three point clouds. The initial alignment of this iterative algorithm is set to the manually measured offsets, given by the mounting positions. The transformation between the radar and the ToF cameras is obtained by manual calibration. A special target (well-visible in ToF and radar data) is placed in the common field of view in order to obtain the sensor disalignment.

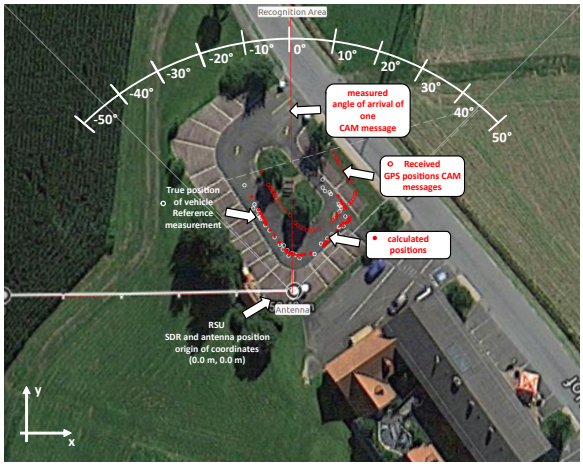


Fig. 6. Visualization of the precise localization obtained from the extended RSU during the test drive. The original data from the GPS is displayed as well as the *ground truth* location data from a reference system.

III. Results

In order to test the system in a real world scenario, we placed the SPIDER in an empty parking lot and performed different evaluations. The RSU and the SDR were positioned close to each other at a precisely known location in order to perform the best possible corrections. We extracted multiple reference points of the parking lot using a highly accurate GNSS RTK rover. In order to evaluate the quality of the environmental perception platform, we placed several targets on the parking lot. The targets for the ToF camera consisted of high-reflective materials (e.g., street signs) and low-reflective materials (e.g., neoprene), while the radar targets were corner cube retroreflectors.

For the V2X localization evaluation, we drove a trajectory on the parking lot and performed GNSS corrections using our extended RSU prototype. The angular data from the RSU was used to correct the vehicle's location on the track. Fig. 6 shows the measured points of the SPIDER's GPS, the corrected GPS points using our prototype and the *ground truth* locations from a reference system. The original GPS data from the SPIDER was off by up to eight meters from the ground truth locations, while the error of the corrected points was always below three meters. The precision was improved by at least 60 percent for all measurement points. Fig. 7 shows a comparison of the distance error of the of the SPIDER's GPS and the corrected locations with the RSU. As an additional reference, we manually analyzed the the environmental perception data to determine the vehicle's location. The distances to objects with known locations are consulted in order to determine the vehicle's location. The comparison of the resulting locations at a certain timestamp is depicted in Fig. 8. The small deviation can be explained by the mounting offset of the OBU from the sensor platform on the vehicle. The achieved accuracy of the RSU localization depends on multiple factors. Among the

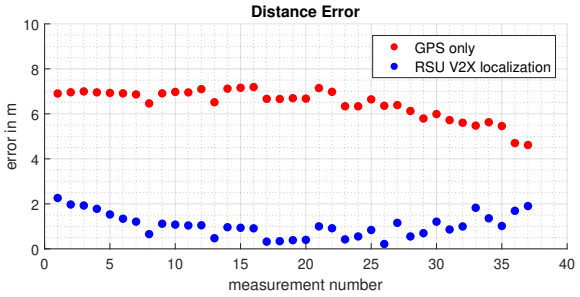


Fig. 7. Localization error of the extended RSU approach compared to the GPS-only solution. The plot compares the recorded data during the test drive.

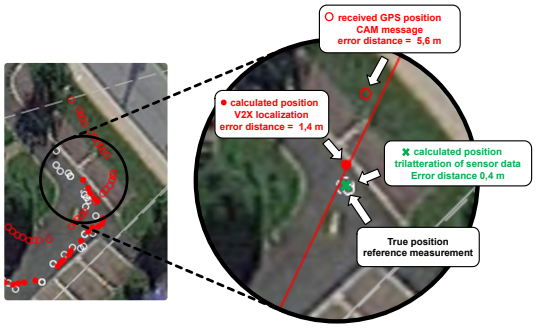


Fig. 8. Detail comparison of the localization error at a single timestamp. Additionally to GPS and V2X localization, the data from the environmental perception sensors is utilized to calculate the vehicle's location.

most influential parameters are:

- the quality of vehicle's GPS data,
- the number of received messages,
- the alignment of the antenna array to the vehicle,
- the distance of the vehicle from the antenna array, and
- the objects between the antenna array and the vehicle.

In order to test the perception capability of the platform, multiple pedestrians were positioned in front of the robot (see image in Fig. 9). The ToF and radar sensor data is visualized

Fig. 9. Fisheye image of the test scene. There are four pedestrians in front of the vehicle. Obtained from [9].

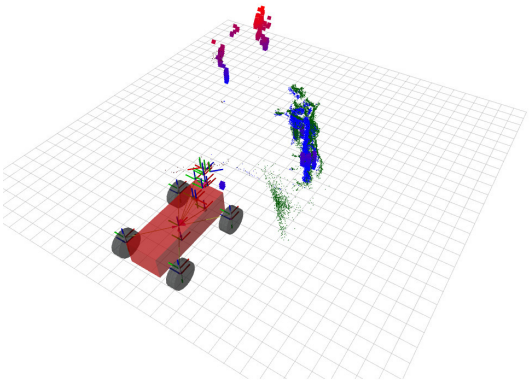


Fig. 10. Visualization of the detected radar peaks and the ToF points. The smaller points represent the ToF points, while the larger points correspond to radar targets.

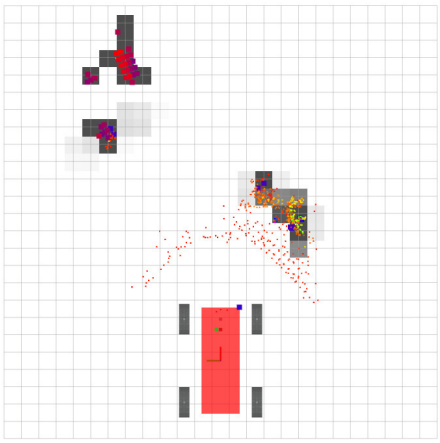


Fig. 11. The resulting occupancy grid. The underlying radar and ToF target points are visualized as a reference. Obtained from [9].

in Fig. 10. The detected peak points are visualized together with the full point cloud of the three ToF cameras. The ToF sensor (with the used settings) is capturing the closer two pedestrians, but struggles to detect the other two. Nevertheless, the radar sensor provides significant targets at the positions of the two further-away pedestrians. In combination, the sensors are capable to detect all four pedestrians. The data is then processed to an obstacle grid and transferred to the SPIDER in order to perform obstacle avoidance. The resulting occupancy grid is depicted in Fig. 11. All pedestrians are recognized by the combination of the different environmental sensors and are marked as occupied cells in the occupancy grid.

IV. Conclusion

We successfully demonstrated a V2X localization prototype, an extended RSU with angle-measurement functionality. The prototype receives the SPIDER's location from GPS via a CAM message, corrects the location error and sends it back via a SPAT message. In the executed measurements, the localization accuracy could be improved by over 60 percent compared to GPS only. A robust and precise localization approach is beneficial for applications in robotics and automated driving. V2X localization infrastructure can be placed at areas of interest in order to perform precise localization. Possible applications are automatic lane positioning in multi-lane crossing sections or correct positioning on temporary lanes caused by construction sites.

Since the proposed form of localization relies on an RSU, it can provide fast reaction times and the data quality does not depend on the weather condition. The localization method is not disturbed by rain, snow or fog. Thus, the approach can be used in difficult weather situations where other localization approaches fail (e.g., visually aided localization). In order to further improve the robustness, RSU localization could be fused with other methods like ground-penetrating radar. For this approach, the RSU could communicate detailed radar scattering information about a region of interest. Another option to further enhance the localization quality is the fusion of the presented approach with vision based methods.

Acknowledgments

The authors of this paper would like to thank the Austrian Research Promotion Agency for funding the *Autonomous CarTo Infrastructure communication mastering adVerse Environments* (ACTIVE) project with the number 855010.

Virtual Vehicle Research Center is funded within the *Competence Centers for Excellent Technologies* (COMET) program by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry for Digital, Business and Enterprise (BMDW), the Austrian Research Promotion Agency (FFG), the province of Styria and the Styrian Business Promotion Agency (SFG). The COMET program is administrated by FFG.

References

[1] M. Cornick, J. Koechling, B. Stanley, and B. Zhang, "Localizing ground penetrating radar: A step toward robust autonomous ground vehicle localization," *Journal of Field Robotics*, vol. 33, no. 1, 2016.

[2] C.-H. Ou, "A roadside unit-based localization scheme for vehicular ad hoc networks," *International Journal of Communication Systems*, vol. 27, no. 1, 2014.

[3] L. Sun, Y. Wu, J. Xu, and Y. Xu, "An RSU-assisted localization method in non-GPS highway traffic with dead reckoning and V2R communications," *2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet 2012 - Proceedings*, 2012.

[4] A. Fascista, G. Ciccarese, A. Coluccia, and G. Ricci, "A localization algorithm based on V2I communications and AOA estimation," *IEEE Signal Processing Letters*, vol. 24, no. 1, 2017.

[5] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, 1986.

[6] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[7] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, November 2015.

[8] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, March 2001.

[9] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Occupancy grid fusion of low-level radar and time-of-flight sensor data," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, Aug 2019.

Context-Aware Sensor Adaption of a Radar and Time-of-Flight Based Perception Platform

Josef Steinbaeck*†, Andreas Strasser†, Christian Steger†, Eugen Brenner†, Gerald Holweg*, and Norbert Druml*
*Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, gerald.holweg, norbert.druml}@infineon.com
†Graz University of Technology, Graz, Austria
{strasser, steger, brenner}@tugraz.at

***Abstract*—This paper presents an approach to enhance the perception quality of a multi-sensor system by considering the context information. The platform's context state is obtained by combining data from the perception sensors and from additional context sensors. This context information is then utilized to dynamically adapt the sensing/processing parameters to the current context. Additionally, the system is capable to detect a reduced perception performance of individual sensors caused by environmental influences.**

The proposed approach was implemented on a multi-sensor platform, equipped with *Time-of-Flight* (ToF) cameras, radar sensors and multiple context sensors. The static *Robot Operating System* (ROS) architecture of the existing platform was extended to support automatic parameter adaption during runtime. In order to demonstrate the approach with real-world data, the platform was exposed to different scenarios. The proposed approach results in a significantly increased perception quality compared to the output of a static implementation.

***Index Terms*—self-adaptive, context-aware, radar, time-of-flight, sensor fusion, ROS**

I. Introduction

The configuration of automotive/robotic environmental perception sensors is among the key influence factors of a system's perception quality. A perception sensor cannot unleash its full potential if it is not configured properly. Traditional perception applications use static sensor configurations, selected based on the expected target environment. If the system is used in various environments and experiences environmental influences, a static sensor configuration limits the perception performance. Thus, modern sensors often have the ability to change some of their parameters during runtime. However, automatic adaptions based solely on the data from the sensor itself are often not sufficient to determine a feasible configuration for a varying context.

This work tackles the described shortcoming by incorporating additional context sensors into the determination process of the sensor parameters. The approach is demonstrated on a multi-sensor perception platform, used to evaluate sensor fusion strategies for research purposes. The platform consists of complementary perception sensors (*Time-of-Flight* (ToF) and radar) and can be mounted on robots and vehicles. Static parameters lead to an unsatisfactory perception performance when the system is used in changing environments. Thus, the platform was extended to support automatic parameter adjustment with the goal to enhance the system's perception performance. The main contributions of this paper to the research community are:

- The identification of ToF and radar sensor parameters, feasible for dynamic self-adaption.
- The extension of an existing *Robot Operating System* (ROS) perception architecture to support the dynamic reconfiguration of sensor and processing parameters.
- An approach to utilize the context information to assign a confidence value to each sensor's data stream.

Future perception systems deployed in changing environments can built upon the presented approach to enable sensor self-adaption and enhance the overall perception performance.

II. Related Work

There exist several approaches which utilize context-information in order to react to the current context. An introduction of the term *context-awareness* and commonly used concepts of context-aware systems are presented in [1]. The work presented in [2] discusses fundamental principles of context-aware adaptive systems. Additionally, they present a conceptual model with the ability to perform context-aware service adaption. The task of finding appropriate scene-dependent parameters is similar to taking a good photo of a given scene. Automatically taking high-quality images has not been achieved yet, but there exist works where the camera parameters are automatically adapted to the current scene [3].

The authors of [4] show a wearable robotic device which exploits contextual information in order to optimize the battery life. In contrast to the approach in this paper, the context information is only utilized by the processing modules, not to adapt low-level sensor configurations. Bloisi et al. [5] sketch a context-aware system architecture for an adaptive cruise control system of an intelligent vehicle. The architecture already sketches the use of context sensors for adaptive processing within the sensor/processing modules. However, solely cameras are used as perception sensors and the architecture is only presented on a high level. The authors of [6] and [7] present a system which allows robots to self-tune their perception parameters on-site. The system parameters are determined through empirical trials on-site, while this paper's approach tunes the parameters during operation.

To the best of our knowledge, the dynamic sensor configuration of environmental perception systems has not been

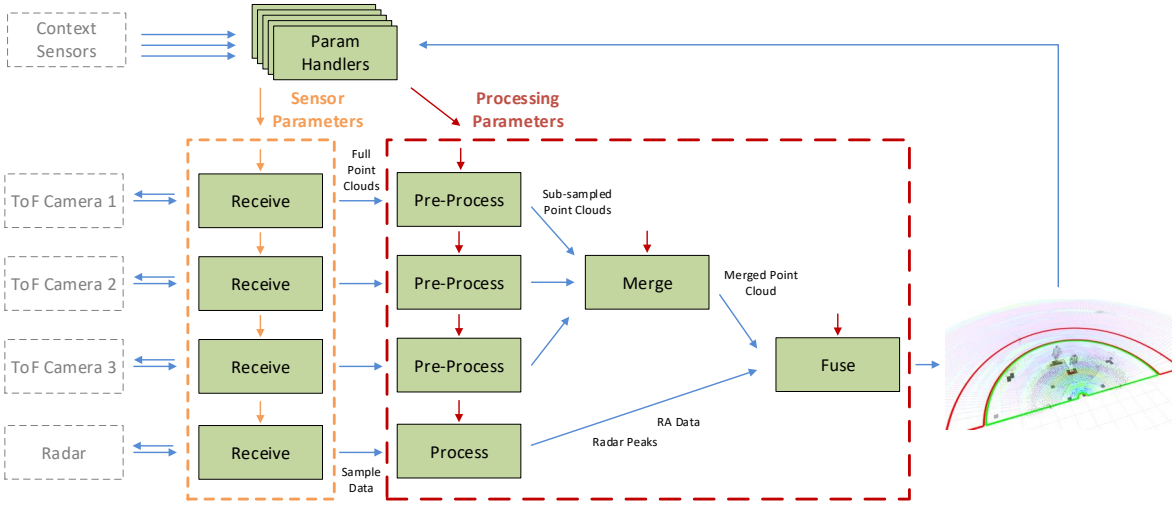


Fig. 1. Overview of the ROS nodes of the context-aware sensor fusion system. The system is able to dynamically adapt to its context during runtime.

sufficiently addressed by the scientific community yet. This paper fills this gap by presenting a ROS-based approach to dynamically change the sensor configuration of a perception system depending on the current context.

III. Context-Aware System Adaption

An existing environmental perception platform was selected as base platform for this work with the goal to perform context-aware sensor fusion. The platform consists of three ToF cameras and one radar sensor. Details about the platform's composition and the sensor synchronization/alignment are available in [8]. In order to get additional information about the environment, multiple context-perception sensors were installed on the the platform. Fig. 1 shows an overview of the extended sensor fusion architecture implemented in ROS. The original processing nodes were extended with services which provide functionality to re-configure certain parameters during runtime. Context sensors and (intermediate) outputs of the perception system are utilized to dynamically adjust the sensor and processing parameters.

This section presents which of the configurable parameters of the perception system are feasible to be changed during runtime. Popular context sensors are introduced which can be used to support the dynamic parameter adaption. Finally, a method to adapt the system parameters in case of context changes is presented. This also includes the fusion module's parameters, which are utilized to assign a context-based confidence value to each input stream.

A. Sensor Parameters

As seen in Fig. 1, the *receive* nodes are not only in charge of receiving raw sensor data, they also (re-)configure the sensors. These nodes implement services which provide other nodes an interface to request re-configuration of the sensors with custom parameters. The services are implemented as *ROS services* and are capable to change all influential parameters of the corresponding sensor. Some sensor parameters can be changed *on-the-fly* while others require the sensor to be restarted, causing a discontinuous data stream. The two different sensor types (ToF and radar) provide individual interfaces with different sensor parameters.

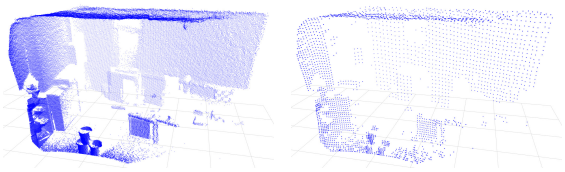
1) Time-of-Flight Sensor: Table I shows a selection of the ToF sensor's parameters. The most influential ones are:

- The **exposure time** affects the range of the camera and the signal-to-noise-ratio. A too high exposure time can have disturbing effects, like saturated pixels or motion blur. Thus, the optimal exposure time is scene-dependent and has to be chosen dynamically.
- The **operation mode** can be set to a number of pre-selected tuples of modulation frequency and measurement mode. This tuple defines the maximum unambiguous range and the duration of a single measurement. The measurement mode can be either the four-phase algorithm or the eight-phase algorithm [9].
- The **frame delay** defines the rate of measurements. A higher frame rate is advantageous for dynamic environments but increases the processing load.
- **Global binning** performs spatial pixel averaging in the distance image. The ToF cameras have a resolution of 352×287 pixels. Global binning can significantly decrease the processing load of the system, but it also decreases the resolution and detection ability.

Not all configurable parameters are listed in the table. Unlisted parameters include the activation of various filters and adaptive thresholds. An example of the impact of the sensor's configuration is shown in Fig. 2. This figure shows the ToF point cloud for different settings of the exposure time and global binning. The exposure time influences the range of the camera, while binning affects the resolution. It can be clearly

TABLE I
SELECTED PARAMETERS OF A TOF RECEIVE NODE.

Parameter	Symbol	Default Value
exposure time	t_{exp}	1500 μs
operation mode	$mode$	eight-phase
frame delay	t_{fps}	200 ms
global binning	bin	1



(a) t_{exp}=1700 μs, bin=1. (b) t_{exp}=500 μs, bin=5.

Fig. 2. The full point cloud of a single ToF camera with different sensor parameters, visualized using the ROS tool *RViz*.

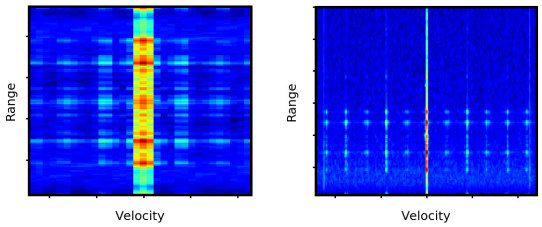
seen that the adaption of single sensor parameters can have a significant influence on the output.

2) Radar Sensor: Table II shows the a selection of the radar sensor's parameters. These parameters are used to compose the transmitted radar waveform, which defines certain characteristics of the radar measurement:

- The maximum range R_{max} and the maximum velocity v_{max} depend on a number of parameters and the speed of light c, as seen in (1) and (2). Typically, the **number of sampels per chirp** N and the **chirp duration** T_p are adjusted to change these characteristics during runtime. The bandwidth B and the center frequency f_c are typically set static.
- The range resolution ΔR and the velocity resolution Δv are defined as stated in (3) and (4). While the range resolution is typically fixed with a static bandwidth B, the velocity-resolution can be easily adapted via the number of **chirps per measurement** M or the **chirp duration** T_p. The angle resolution depends on the number of **receive channels** L_{RX}. Forming virtual antennas with multiple **transmit channels** L_{TX} can increase the number of physical receive antennas.
- The **frame delay** defines the time between two measurements and thus, the frame rate.

TABLE II
SELECTED PARAMETERS OF THE RADAR RECEIVE NODE.

Parameter	Symbol	Default Value
chirp duration	T_p	200 μs
samples per chirp	N	1024
chirps per measurement	M	128
receive channels	L_{RX}	16
transmit channels	L_{TX}	1
frame delay	t_{fps}	200 ms



(a) N=128, M=32. (b) N=1024, M=256.

Fig. 3. Radar range-Doppler image for two different configuration settings for the number of chirps M and the sample rate N.

$$R_{max} = \frac{N\,c}{4\,B} \tag{1}$$

$$v_{max} = \frac{c}{4\,T_p\,f_c} \tag{2}$$

$$\Delta R = \frac{c}{2\,B} \tag{3}$$

$$\Delta v = \frac{c}{2\,M\,T_p\,f_c} \tag{4}$$

The sample rate, the number of chirps and the receive antennas define the amount of received raw data. More received data enables higher precision measurements but also increases the processing load. In combination with the frame delay, this defines the data rate. A higher rate is beneficial for dynamic scenarios but increases the processing load. Fig. 3 shows the radar range-Doppler image for two different settings of the samples N and the number of chirps M. The output clearly shows the influence of these parameters on the maximum range R_{max} as well as on the velocity resolution Δv.

To summarize, the parameters listed in Table I and II have a significant impact on each sensor's output data. Thus, these parameters were selected for the dynamic adaption of the perception sensors during runtime.

B. Processing Parameters

Not only the sensor parameters, but also parameters during later processing can be adapted during runtime. Fig. 1 highlights the ROS nodes which provide services to change processing parameters. The processing parameters can have a great influence on the processing modules' output and thus, also on the output of the whole system. Examples of processing parameters are the peak detection threshold for radar processing or the voxel size for ToF point cloud processing.

Adjustable processing parameters can be changed *on-the-fly* and are applied immediately. In contrast, the sensor parameters are subject to a sensor-dependent delay until they become active. The dynamic adaption of processing parameters can be either performed *online* with live sensor data or *offline* with earlier recorded data. The offline evaluation enables the quantitative comparison of different self-adaption approaches

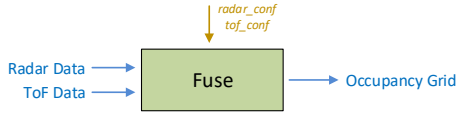


Fig. 4. Subscribed/published topics and provided service parameters of the fusion node.

performed on the same dataset. The sensor parameters can only be adapted during online evaluation, since they define the shape and characteristics of the raw sensor data.

The approach can also be used to provide supplementary parameters to a sensor fusion module (see Fig. 1), which combines data streams from different sensors into a common representation. This includes the dynamic assignment of confidence values to the input data streams of the fusion module if the corresponding sensor data is known to be disrupted. If the system senses that a perception sensor operates outside its ideal working conditions (e.g., heavy rain, direct sunlight), the confidence of this sensor's data stream is decreased for the fusion process. The fusion module utilizes the confidence value by assigning a lower weight to the corresponding sensor data or by discarding (part of) the sensor data.

Fig. 4 shows the fusion module with its corresponding inputs, outputs and supplementary parameters. The module's processing parameters include a confidence range for the ToF sensor's data stream *tof_conf* and for the radar sensor's data stream *radar_conf*. The data points within the corresponding sensor's confidence range are considered for fusion, while the other data points are discarded.

C. Context Perception Sensors

This section introduces multiple sensors of various fields, capable of providing measurement data which can be utilized to estimate a system's current context. The following sensors are popular choices to obtain context information of the environment:

- **Light sensor** detects the amount of light exposed to the sensor platform. Ambient light can cause a performance loss in vision based perception sensors.
- **Battery sensor** measures the remaining capacity of the battery by analyzing the voltage of the battery.
- **Rain sensor** detects if a system is exposed to rainfall, limiting the performance of certain perception sensors.
- **Temperature sensor** measures the ambient temperature in order to get a better understanding of the context.
- ***Inertial Measurement Unit* (IMU) sensor** detects motions of the platform.
- **Odometry sensor** detects the system's trajectory and the velocity.
- ***Global Navigation Satellite System* (GNSS) sensor** locates the platform on a map and is capable to assign it to a context (e.g., rural road, urban area).

A light sensor, a battery sensor, a temperature sensor and an IMU were selected to provide context information to the existing perception platform. Additionally, the environmental perception sensors of the system provide valuable data about the environment. Thus, the raw data and intermediate processing data is also utilized to generate a more exact estimate of the context.

D. Handling Context-State Changes

The most challenging task of a self-adapting system is to utilize the available data (context sensors, perception sensors, processed data) in order to perform re-configuration of certain parameters. The presented approach uses a parameter handler for each of the sensor and processing parameters (as indicated in Fig. 1). A subset of the available context data is consulted for the calculation of the individual parameters. Thus, each parameter handler subscribes to individual context data streams and evaluates the input data in order to determine an optimized value for the corresponding parameter. Finally, the parameter handler decides whether the sensor or processing parameter is adapted to the new value or not.

The data sources of the parameter handlers can be categorized into three categories: context sensors, perception sensors and processed data. The context sensors provide certain context information (e.g., light intensity or accelerations). The data provided by the environmental perception sensors themselves (ToF, radar) can also be used to extract context information. However, the raw data has to be processed (e.g., histogram) in order to act as an input for the parameter determination. Additionally, the intermediate and output data of the processing system is made available to the parameter handlers. These data streams provide a performance indication of the current parameters and can act as a feedback to the parameter handling modules.

Each parameter handler gets triggered whenever any new input data is available. A new parameter value is calculated using a model, derived from known dependencies, system knowledge and heuristics. The difference to the old parameter is then used to calculate the *gain* of the proposed parameter change. Since changing parameters is time-consuming, every parameter change is assigned with a *cost*. If the gain is higher than the cost, the parameter change is initiated. Control values of the parameter handlers (e.g., thresholds) are fixed during runtime but can be changed manually on every startup.

As an example, Fig. 5 shows the inputs and outputs of the parameter handler for a ToF sensor's *global binning* parameter. As seen in the figure, the parameter handler not only receives information from context sensors (IMU, light sensor), it also receives data from the perception sensor itself (ToF). The IMU data is consulted in order to detect if the system is currently moving. The light sensor gives an indication whether the ToF sensor is influenced by sunlight. The data from the ToF sensor itself is consulted to inspect the data quality within the area of interest and to gain knowledge about the presence of objects. These parameters are then used to calculate a new value for the ToF sensor's *global binning* parameter.

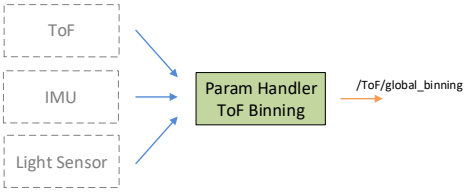


Fig. 5. The parameter handler node of a ToF sensor's *global binning* parameter.

IV. Results

This section presents the sensor output of the proposed sensor platform for two different use cases. In the first example, the system dynamically changes the sensor configuration according to the current context. The second use-case shows how the information of the context sensors is utilized to provide supplementary parameters for the system's sensor fusion module.

A. Context-Aware Sensor Adaption

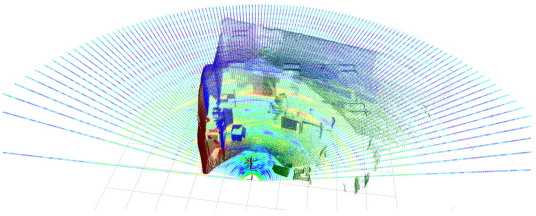
The context-aware adaption of sensor parameters is demonstrated for two different context states while being placed in the same scene. In the first scenario, the platform is kept static (not moving), while it is rotated in the second scenario. When the platform changes its field-of-view, it is beneficial to have a high frame rate to quickly react to scene changes. Using the full sensor data results in a high processing load and limits the measurement rate of the system. Thus, the system shall switch to a reduced data mode with a higher frame-rate when the system is moving. The IMU sensor is capable to detect movements of the platform, by measuring the specific force and the angular rate. The parameter handlers are configured to reduce the processing load and to increase the frame rate while the platform is moving. The data requirement D of the processing system's input data, consisting of the radar's raw sensor data and the ToF sensors' point clouds is stated in (5). Each radar sample point is represented by a 16 Bit integer value, while each ToF point cloud point is represented by six 32 Bit float values (x, y, z, intensity, noise, confidence).

$$\begin{aligned} D &= D_{Radar} + 3 \cdot D_{ToF} \\ &= N \cdot M \cdot L \cdot 16\,\text{Bit} + 3 \cdot points \cdot 6 \cdot 32\,\text{Bit} \end{aligned} \tag{5}$$

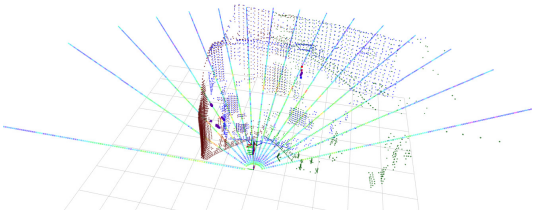
Fig. 6 shows the visualization of the combined sensor output for the two different context states. The sensor output is dissimilar for these two scenarios since the context-aware parameter handlers determine different parameter values. The determined parameters of the context-aware platform are presented in Table III. Additionally, the table shows the resulting data load during the static and the dynamic scenario. For the dynamic operation, the data is reduced by a factor of about 60, resulting in a lower processing load. The maximum achievable frame rate in the static scenario (high data density) is about 1 *frame per second* (FPS), while the frame-rate of the dynamic scenario (reduced data density) is about 20 FPS.

TABLE III
Parameter settings for different scenarios.

	parameter	static	dynamic
Radar	samples (N)	1024	64
	ramps (M)	512	32
	channels (L)	16	16
ToF	binning	1	6
Data		15.6 MB	0.26 MB



(a) Static scenario.



(b) Dynamic scenario.

Fig. 6. Visualization of the pre-processed sensor data for two different configuration scenarios.

B. Context-Aware Sensor Data Fusion

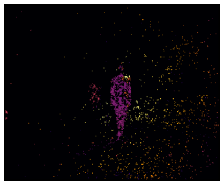
This section shows how to utilize the context information in order to provide supplementary parameters to the sensor fusion module. The ToF camera's distance measurement is based on illumination with modulated infrared light. The measurement is affected by ambient light with components in the same wavelength area. Thus, the camera shows reduced performance in the presence of bright sunlight. A light sensor is used as a context sensor to measure the ambient light radiating onto the ToF sensor. The measured light value is then transferred to the parameter handler for the ToF sensor's confidence range. This parameter handler calculates the confidence range for the affected ToF sensor and adjusts the corresponding parameter in the *Fusion* module.

In order to show the influence of sunlight, the ToF sensor's output of the same scene was captured during daytime and during nighttime. Fig. 7a and Fig. 7c show the fisheye images of both scenarios. The clean ToF distance image of the night scene is depicted in Fig. 7b, where the two pedestrians are clearly visible. The distance image during the day scene is shown in Fig. 7d. The sunlight causes an increased noise and the pedestrians are not fully recognizable anymore. Even though the sensor has an integrated circuit to suppress background illumination, the quality of the sensor

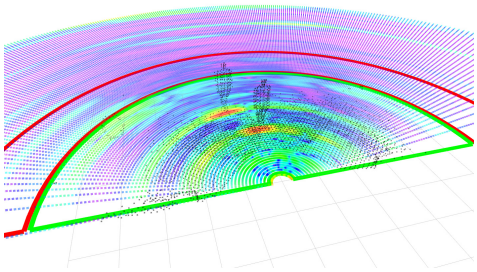
(a) Scene during night-time.

(b) ToF distance image at night-time.

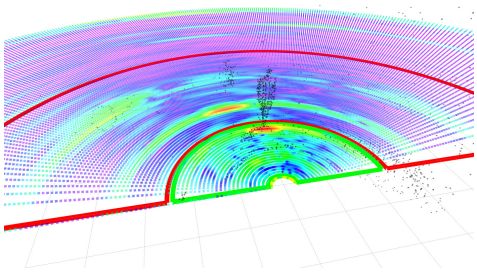
(c) Scene at day-time.



(d) ToF distance image at day-time.



(e) Combined visualization of radar and ToF data with an indication of the ToF sensor's confidence range at night-time.



(f) Combined visualization of radar and ToF data with an indication of the ToF sensor's confidence range at day-time.

Fig. 7. Degradation of a disrupted ToF sensor due to influence of bright sunlight.

output is reduced when used in this environment. Detecting that influence enables to supplement the fusion module with a limited confidence range for the affected sensor. Fig. 7e and Fig. 7f show the visualization of the pre-processed radar and ToF perception data with an indication of the ToF sensor's confidence range. The point cloud within the green semi-circle is assumed to be confident, while the remaining data is assumed unreliable and marked with the red semi-circle. The fusion module considers only the confident part of the ToF data when fusing the radar and ToF data streams. Excluding unreliable data from the fusion task decreases the chance of faults in the output data and enables the application of confidence-dependent fusion strategies.

V. Conclusion

The performance of environmental perception systems can be significantly increased by incorporating context perception sensors. Context sensors can provide valuable information in order to adapt sensor and processing parameters. They can also act as an additional input in order to perform sanity checks of the system. The proper configuration of the environmental perception sensors has a strong influence of the output data quality. Thus, the automatic adaption of the parameters during runtime is essential for perception systems in challenging environments.

This paper presents an extension of an existing perception platform in order to enable parameter self-adaption during runtime. Multiple context sensors were added to a multi-sensor perception platform which fuses ToF and radar sensor data. The context data is analyzed during runtime and considered in order to adapt the sensor and processing parameters. Using this platform, it was possible to demonstrate multiple use cases utilizing context-aware parameter adaption. Future perception systems can build upon this approach and include context information in order to enhance the perception performance.

Acknowledgments

AutoDrive has received funding within the Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's H2020 Framework Programme and National Authorities, under grant agreement number 737469.

References

[1] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[2] J. Z. Sun and J. Sauvola, "Towards a Conceptual Model for Context-Aware Adaptive Services," *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, pp. 90–94, 2003.

[3] H. Ahn, D. Kim, J. Lee, S. Chi, K. Kim, J. Kim, M. Hahn, and H. Kim, "A robot photographer with user interactivity," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5637–5643.

[4] D. Huen, J. Liu, and B. Lo, "An integrated wearable robot for tremor suppression with context aware sensing," *BSN 2016 - 13th Annual Body Sensor Networks Conference*, pp. 312–317, 2016.

[5] D. D. Bloisi, D. Nardi, F. Riccio, and F. Trapani, "Context in robotics and information fusion," in *Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge*, 2016, pp. 675–699.

[6] H. Hu and G. Kantor, "Efficient automatic perception system parameter tuning on site without expert supervision," in *Proceedings of the Conference on Robot Learning*, 2017, pp. 57–66.

[7] H. Hu and G. Kantor, "Compensating for context by learning local models of perception performance," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4629–4634.

[8] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Design of a low-level radar and time-of-flight sensor fusion framework," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 268–275.

[9] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, 2001.

A Hybrid Timestamping Approach for Multi-Sensor Perception Systems

Josef Steinbaeck*†, Christian Steger†, Eugen Brenner†, and Norbert Druml*
*Infineon Technologies Austria AG, Graz, Austria
{josef.steinbaeck, norbert.druml}@infineon.com
†Graz University of Technology, Graz, Austria
{steger, brenner}@tugraz.at

***Abstract*—Synchronized and precisely timestamped data from perception sensors is highly advantageous for the low-level fusion of multiple sensor data. Many open-available, low-cost perception sensors do neither provide hardware support for precise clock synchronization, nor provide timestamps with their measurement data. In this work, we present an approach to enable synchronization and accurate timestamping of hardware-triggerable sensors in multi-sensor perception systems.**

We utilize a hybrid timestamping approach, taking into account the timestamp of a hardware trigger and the software timestamp. The presented timestamping approach utilizes the trigger time to assign precise timestamps to the data streams of the perception sensors. Precise timestamps are mandatory in order to achieve a high perception performance in dynamic applications which utilize low-level data streams.

Additionally, we present an implementation of the approach on a multi-sensor perception platform, archiving a timestamp precision in the range of 2 ms. An existing Robot Operating System (ROS) architecture of the platform is extended to assign hybrid timestamps to the data streams. Additionally, we present a pedestrian detection implementation which fuses the timestamped data into a representation.

***Index Terms*—radar, time-of-flight, robotics, sensor fusion, ROS**

I. Introduction

Environmental perception is a key-enabler of autonomous systems (e.g., vehicles, robots, drones) and mobile cyber-physical systems (e.g., smartphones, wearables). These systems have to react to the environmental condition in order to fulfill their desired purpose. The perception modules of these devices typically incorporate data from multiple sensors in order to enhance their perception capabilities. Since all currently available sensors have weaknesses, multiple complementary and redundant sensors are utilized and fused in order the increase the overall perception quality and robustness. However, in order to perform high-quality fusion and to model temporal processes, each sensor's measurement time has to be known precisely. Synchronized sensor data is desirable in many applications, i.e, having a fixed relationship between the acquisition times of the single sensors. Sensor fusion at a low-level often requires data from all sensors to be captured at the same time. The data from certain sensors can not be combined in a feasible way if scene/pose changes occurred between the measurements.

There exist multiple methods and protocols which implement the functionality to synchronize and precisely timestamp multiple perception sensors (e.g., based on the Ethernet connection). However, since many consumer-grade sensors do not come with a precise clock and utilize USB as data interface, these concepts can not always be applied. Thus, many custom-build platforms match the receive-timestamps of non-synchronized sensor measurements in order to perform sensor fusion, e.g., as presented in [1].

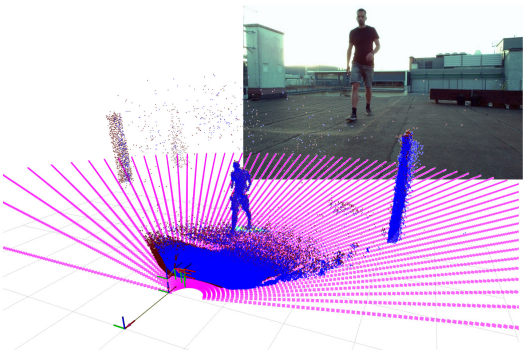


Fig. 1. Synchronized sensor data from multiple perception sensors (ToF, radar, vision camera), visualized using the ROS visualization tool *RViz*.

The fusion of approximately synchronized measurements might be acceptable for certain applications, but will significantly reduce the perception quality in others. The unknown sensor latency contained in the receive-timestamps can result in the fusion of measurements with significantly different acquisition times. In dynamic scenes, this will result in a scene change between the measurements which decreases the perception performance.

Synchronized measurements without precise timestamps (e.g., using hardware triggering) can utilize a sequence number in order to assign corresponding measurements. However, if unexpected framedrops or delays during data transfer occur, the sensor fusion will be performed on misaligned data and the perception quality will decrease.

In this work we introduce an approach to obtain synchronized and precisely timestamped data streams from externally triggered perception sensors. The method does not require any additional hardware support and enables simultaneous data acquisition and precise timestamping of the sensor measure-

ments. Framedrops during operation are detected and handled in order to avoid misaligned data streams. The presented approach is implemented on a *Robot Operating System* (ROS)-based platform, equipped with radar, *Time of Flight* (ToF) and vision sensors. Fig. 1 shows a visualization of raw sensor data provided by the platform. A simple and efficient pedestrian detection is implemented on the platform in order to demonstrate the robustness of the presented approach.

To summarize, the contributions of this paper to the scientific community are:

- A new approach to obtain synchronized and precisely timestamped measurements in a multi-sensor perception platform, consisting of market available sensors.
- A technique to detect and handle framedrops during operation.
- An overview of the ROS-based architecture of a perception platform, implementing this approach.
- A demonstration of pedestrian detection, utilizing the synchronized sensor data from the perception platform.

Related work, focusing on the synchronization and timestamping of multiple perception sensors is presented in Section II. In Section III we present a method to perform sensor synchronization and hybrid timestamping on a multi-sensor perception platform. An external microcontroller is used to synchronize the sensors, while a precise timestamp is added to the data streams of the perception sensors. Section IV shows the implementation of the approach on a research platform which performs the popular use case of pedestrian detection. Finally, Section V and VI summarize the results and give a short conclusion of the work.

II. Related Work

Since synchronization and precise timestamping is a common requirement for multi-sensor perception platforms, a number of related works has already been published within the research community. Brahmi et al. [2] perform a timestamping and latency analysis for multi-sensor perception systems deployed in *Advanced Driver Assistance Systems* (ADAS). They categorize sensors into three classes: sensors with global timestamps, sensors with relative timestamps and sensors without timing information. Different methods are presented to analyze the latency components contained in the timestamps in order to obtain the uncertainty of the measurement time. The *Controller Area Network* (CAN) bus is used as a low-latency bus in order to synchronize the timebase of multiple sensors. The authors conclude that a known temporal alignment is crucial in order to perform sensor fusion. In contrast to the presented paper, our approach focuses on synchronized sensor operation without automatically attached timing information.

The authors of [3] provide an overview of different synchronization strategies for automotive multi-sensor fusion. The work inspects the worst-case latency introduced by a sensor fusion system for asynchronous and synchronous sensor measurements. The authors show that non-synchronized measurements introduce a higher worst-cast latency to the fusion module, especially if the sensors operate at different measurement frequencies. Since our work targets low-level fusion of simultaneously acquired data streams, synchronized measurements are mandatory for the fusion process. Additionally, we focus more on the process of assigning and determining an accurate timestamp to each data stream.

Huck et al. [4] present an approach to perform exact timestamping of asynchronous sensor measurements. They implement a Kalman filter which estimates the sensor cycles as well as jitter-corrected software timestamps. A calibration method is presented to estimate and correct the different sensor latencies in a multi-sensor setup. In contrast to our approach, the authors utilize asynchronous sensor measurements and the method requires an additional calibration phase.

In [5], the authors address the problem of framedrops during the operation. The authors introduce a method to detect and handle framedrops in order to avoid a significantly wrong timestamp estimation. The detection of framedrops is also considered for the method presented in this paper. Depending on the regarded sensor we show custom methods to detect and handle dropped frames.

Guivant et al. [6] present an autonomous platform, able to create 3D maps of indoor and outdoor environments. They emphasize the relevance of precise timestamps, which are important for certain data fusion processes. The authors apply Bayesian estimation with the goal to obtain a highly accurate estimation of the timestamps

The *Precision Time Protocol* (PTP) [7] is commonly used to synchronize the clocks of local networks, while the *Network Time Protocol* (NTP) [8] is often use to synchronize the clocks of distributed networks over the internet. NTP typically achieves a precision in the dimension of single microseconds, while the PTP can achieve sub-milliseconds. Since PTP is well suitable for the sensor synchronization of perception systems, it is supported by several Ethernet-based sensors. But many openly available sensors do not provide an Ethernet interface (rather USB), and also no own clock source. Thus, these protocols are not supported by the sensors.

The authors of [9] present an approach to synchronize multiple perception sensors of a research vehicle using ROS. The approach utilizes a PTP time-synchronization between the processing system and the triggering microcontroller via an Ethernet connection. The triggering microcontroller transfers the trigger timestamps to the processing PC which adds the timestamps to the sensors' data streams. We use a similar approach to apply timestamps to the data streams, but use a USB-connected microcontroller which does not provide PTP support. Additionally, we show a way to deal with lost measurements during data acquisition.

Schneider et al. [10] propose a sensor synchronization method capable to be used for low-level fusion and in dynamic environments. The work targets the synchronization of a lidar and a vision sensor mounted on a research vehicle. The bearing information of the lidar sensor is used to trigger the camera at the exact moment, when the beams of the laser scanner pass the camera's field of view. A low-level real-time capable computer is in charge of generating the trigger event for the

camera at the right time. The presented paper lacks detail information about how the timestamps from the real-time platform are unambiguously assigned to the data packets. Our approach uses radar, ToF and vision sensors, but also makes use of a microcontroller for trigger generation.

To the best of our knowledge, methodologies to achieve synchronization of multiple consumer-grade perception sensors have yet not been sufficiently addressed by the scientific community. Additionally, there is a lack of openly available work dealing with real-world effects of perception sensors, like framedrops. This is why we fill this gap, by presenting a novel approach and a system architecture in order to perform synchronization and robust timestamping of multiple perception sensors.

III. A Hybrid Timestamping Approach for Multi-Sensor Perception Systems

This section introduces an approach to perform synchronization and timestamping of multiple perception sensors used in a ROS-based perception platform. First, we introduce the ROS-based design of the base perception system, capable of acquiring low-level data streams from multiple perception sensors. An external microcontroller is utilized to synchronize the sensors by generating trigger pulses for each of the sensors. Afterwards, we point out some issues that come along with software timestamping, where a timestamp is added to the data stream at the time the processing system receives the data. We end the section by presenting a hybrid timestamping method and showing how it can be used to handle framedrops during operation.

A. Base Perception System

The base perception system receives the raw data from multiple perception sensors, performs pre-processing and aligns the data. The data is packed into standardized messages and forwarded to other sub-systems for further processing. An overview of the base perception system's main modules can be seen in Fig. 2. The reception modules control the sensor configurations and receive the raw data from the sensors. Additionally, each sensor's pre-processing modules perform low-level processing of the raw data (e.g., compression). A trigger module is in charge of configuring the trigger microcontroller, which triggers the single sensors. In general, the base perception system is running on a non real-time capable x86 computer.

All sensors provide an input for an external trigger signal which can be used to trigger a new measurement. A microcontroller generates a trigger signal for all perception sensors, every time a new measurement is desired. Synchronized data acquisition is desired for the low-level fusion of sensor measurements from multiple sensors. The sensor measurements shall be started exactly at the same time. This way it is guaranteed that all sensors provide data from the same scene, even in dynamic scenarios. Synchronized measurements have the advantage that there is neither movement of the sensor platform, nor movement of dynamic objects between the sensors' measurements.

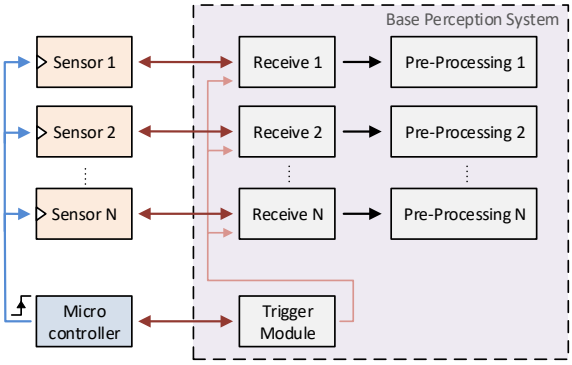


Fig. 2. Main processing blocks of the framework. The base perception system's modules are in charge of low-level communication with the individual sensors as well as the pre-processing.

The trigger module controls the frame rates of the single sensors by configuring the trigger microcontroller. The frame rate can be set for each sensor individually, but in order to provide synchronized acquisition, the rates have to be integral multiples of a common base rate. For the low-level fusion in this approach, we use a common frame rate for all sensors (rate of the slowest involved sensor). The trigger module allows a static configuration of the frame rate at startup, but also allows dynamic re-configuration during runtime.

The used perception sensors do not provide functionality to synchronize the local clock with an external source. Thus, the raw data of a measurement does not include any absolute timing information about the acquisition time. However, a subset of the used sensors are capable of attaching the local clock value of the measurement to the raw data. Knowing the oscillator frequency of the sensor, the receive module can utilize this value to perform additional timing analyses.

Each reception module holds a running sequence number, increased after each successfully received raw data packet from the sensor. This sequence number is added to the data streams provided by the base perception system. Without any timestamping information, this sequence number can be utilized to assign corresponding measurements in later processing. However, since no absolute timing is available, it is impossible to align additional (not-triggered) measurements to the system.

B. Software Timestamps

Software timestamps utilize the system time of the base perception platform at the time when the raw data from the sensors is received t_{sw}. This time does not equal the measurement time t_{meas} of the sensor, it is delayed by the measurement latency Δt_{sw}. The measurement latency is influenced by multiple parts, the main contributions are stated here:

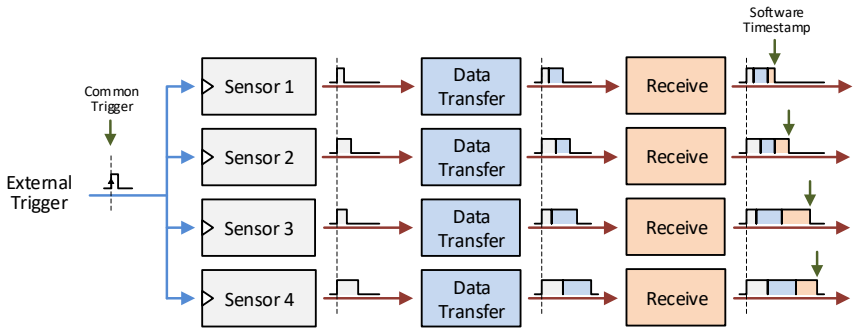


Fig. 3. Added measurement delays to the data streams of simultaneously triggered sensors before a software timestamp is assigned.

- Measurement acquisition, the time from starting the measurement until having the data digitized and ready to be transferred.
- Data transfer, the delay added for the transfer of the raw data via an interface.
- Reception, the time added from physical reception until the data is available to the software process (buffering, scheduling, etc.).

The relation between the software timestamp t_{sw} and the actual measurement timestamp t_{meas} for the i-th measurement is shown in (1).

$$t_{\mathrm{sw},i} = \Delta t_{\mathrm{sw},i} + t_{\mathrm{meas},i}. \tag{1}$$

The measurement delay Δt_{sw} is unknown and not static. Thus, raw data originating from the same trigger event, will result in different software timestamps in the respective reception modules. These different timestamps make the association of corresponding data-streams in later processing a difficult task to solve. Fig. 3 shows an illustration of the non-deterministic delays added to the data streams after the simultaneous acquisition. As indicated in the figure, the assigned software timestamp of the received data streams can differ significantly.

Since the sensors' data streams are timestamped with an unknown latency related to the measurement time, the timestamp cannot be used to accurately model dynamic processes. For example, the estimation of an object's velocity (and future positions) highly depends on accurate timestamps. Although estimation techniques can be used to reduce the effects of the latency component, the impact is still severe for many applications. Additionally, it is also not possible to add further sensor data to the measurement, which is not triggered by the same microcontroller. Software timestamps might be sufficient in cases which do not require real-time (e.g., visualization, buffered systems). But if the perception system is used for the estimation of dynamic scenarios (e.g., pose estimation, mapping, etc.), accurate timestamps are crucial. Thus, software timestamps are not sufficient for the targeted platform.

C. Hybrid Timestamps

A more feasible approach is to utilize the time of the microcontroller's trigger event which starts the perception sensors' measurements. The base perception system is notified at each trigger event in order to determine a precise estimation of the real measurement timestamp. We present an easily integrable method for the approximation of the trigger time with millisecond accuracy.

Since the trigger is generated on a non-synchronized microcontroller, the absolute time of the trigger activation can not be directly obtained. Thus, the microcontroller notifies the trigger module after every new trigger event, which in response estimates the measurement timestamp. This timestamp is then forwarded to the reception modules and added to the corresponding data streams. Fig. 2 shows the connection of the trigger module to the reception modules as well as to the microcontroller. The trigger module adds a sequence number to the timestamp, indicating the measurement number of the corresponding timestmap.

Fig. 4 shows the temporal interaction between the microcontroller, a perception sensor and a receive module for a single measurement. The microcontroller sends a notification to the trigger module at the time the trigger is activated. The trigger module creates a hybrid software timestamp t_{hsw} at the reception time of this notification. Based on that value, the module estimates the measurement timestamp $\hat{t}_{\mathrm{meas}}$ and forwards it to the reception modules. The timestamp of the i-th measurement arrives to the reception module before the i-th raw sensor data is received. At the time the reception module receives the raw data, it can directly assign it with the earlier received i-th timestamp. Additionally, the reception module generates a software timestamp t_{sw} when the raw data is received, used for additional error detection.

A short message is used to notify the trigger module of a new measurement, in order to keep the hybrid timestamp's latency as low as possible. The hybrid timestamp contains zero acquisition latency and the latencies introduced by the data transfer and for the reception are minimal. Thus, the hybrid software timestamp t_{hsw} contains a much lower uncertainty than the software timestamp t_{sw}. In order to further reduce

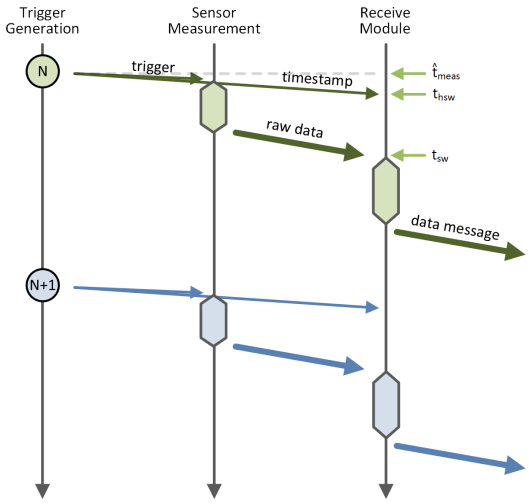


Fig. 4. Temporal interaction between the single modules involved in the hybrid timestamping procedure.

the included uncertainty, the latency is estimated based on prior measurements of the communication delay. The round trip delays of short messages between the microcontroller and the trigger module were measured. The half of the average round trip delay is then used as an estimate of the hybrid timestamp's latency $\hat{\Delta}t_{\mathrm{hsw}}$. The estimation of the i-th measurement timestamp $\hat{t}_{\mathrm{meas},i}$ can be calculated as stated in (2).

$$\hat{t}_{\mathrm{meas},i} = t_{\mathrm{hsw},i} - \hat{\Delta}t_{\mathrm{hsw}}. \tag{2}$$

In order to improve the robustness against single outliers in the communication time, the delay between hybrid software timestamps is analyzed. The duration T_{fps} between two measurements is defined by the frame rate, and equals the expected time difference between two consecutive hybrid software timestamps Δt_{hsw}. If these two delays differ too much, the new timestamp is calculated from the last timestamp using the known frame delay, as seen in (3).

$$\hat{t}_{\mathrm{meas},i} = \begin{cases} \hat{t}_{\mathrm{meas},i} & \text{, if } \Delta t_{\mathrm{hsw}} \approx T_{\mathrm{fps}}, \\ \hat{t}_{\mathrm{meas},i-1} + T_{\mathrm{fps}} & \text{, otherwise.} \end{cases} \tag{3}$$

D. Handling Framedrops

During the operation of the platform, measurements can occasionally get lost before they are received by the platform. These framedrops can have various reasons, like transmission errors, acquisition errors or overheating. Adapting the sensor parameters during runtime can also cause framedrops until the new settings are active. We show an approach to detect framedrops in the reception module by utilizing the estimated measurement timestamp and the software timestamp.

If the reception module does not receive raw data for a lost measurement, the module's sequence number gets misaligned in respect to the global sequence number. This causes a significant error in the timestamp assignment, since the receive module uses the sequence number to pick the corresponding timestamp. After two lost measurements, the receive module would permanently pick the timestamp corresponding to the the trigger event of two measurement cycles earlier. Thus, the detection and handling of framedrops is of major importance in order to perform robust timestamping in real-world conditions.

In order detect framedrops, the receive module compares the software timestamp of the current measurement $t_{\mathrm{sw},i}$ with the software timestamp of the previous measurement $t_{\mathrm{sw},i-1}$. If this difference Δt_{sw} is significantly higher than the frame delay T_{fps}, then at least one framedrop is likely to have occurred. The estimated number of skipped frames is then calculated using (4), where t_{tresh} denotes a custom threshold in order to compensate small fluctuations of the software timestamp's cycle. The sequence number of the receive module is then updated accordingly and normal operation is continued.

$$N_{\mathrm{skipped}} = \left\lfloor \frac{\Delta t_{\mathrm{sw}} + t_{\mathrm{tresh}}}{T_{\mathrm{fps}}} \right\rfloor - 1 \tag{4}$$

If a sensor is capable to attach the local clock value of the measurement acquisition time to the raw-data, this value can be used in a similar way to detect framedrops. In that case, the difference between the local clock values of two consecutive measurements is compared to the frame delay T_{fps}. This method is superior to the aforementioned method, since it excludes the unknown measurement latency (acquisition, data transfer, etc.). Thus, if a sensor provides a local clock value, this method is a robust way to determine framedrops.

IV. Application: Pedestrian Detection

We implemented the proposed synchronization and timestamping approach on a multi-sensor perception platform. In order to demonstrate the functionality of the method in a real-world use case, we implemented a pedestrian detection algorithm on the platform. This section first introduces the ROS-based perception platform and then gives an overview of the implemented use case.

A. Perception Platform

We utilize an existing ROS-based research platform, equipped with multiple environmental perception sensors and a processing unit. The system consists of two ToF cameras[1], a

[1]CamBoard pico monstar: 3D imaging development kit of the pico family (https://pmdtec.com/picofamily).

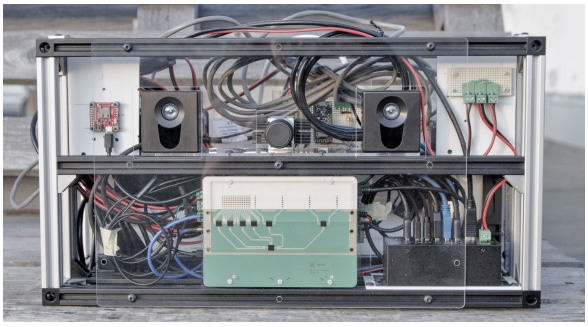


Fig. 5. Front view image of the research platform, consisting of multiple environmental perception sensors.

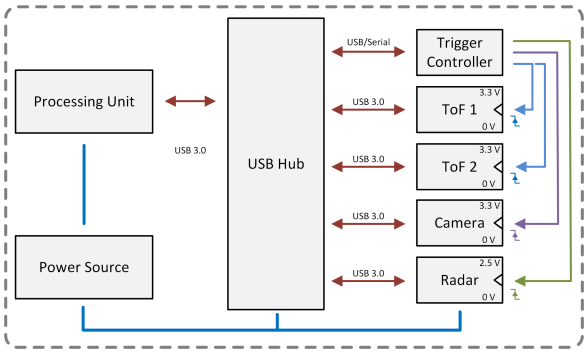


Fig. 6. Block schematic of the multi-sensor platform's major components.

radar sensor[2] and a vision camera[3]. The platform is equipped with a microcontroller which is in charge of generating trigger signals to control the data acquisition. A processing unit[4] implements the software modules of the ROS-based processing architecture.

The platform can be used stationary, but can also be mounted on a moving platform (e.g., vehicle or robot). A battery is included to power the platform in standalone-mode, while it can also be supplied externally. A front-view picture of the platform is presented in Fig. 5. The aluminum profile system enables easy-mounting of the platform on various objects/vehicles. Fig. 6 shows a simplified overview of the hardware components integrated in the platform.

All perception sensors, as well as the microcontroller are connected to the processing unit via USB. The sensors deployed in the proposed platform (radar, ToF, camera) are capable to be synchronized via an external trigger signal. The individual ToF cameras are triggered with a short delay (< 2 ms) to avoid parallel illumination of the two ToF cameras. The radar and the vision measurement are triggered at the same time as the first ToF sensor. The base perception system of the platform outputs data streams of temporary and spatially aligned, pre-processed sensor data. These data streams can then be used by further sub-systems in order to implement certain use cases without the need to align low-level data.

[2]RadarLog: A platform for microwave radar data capturing and logging (http://www.inras.at).

[3]Basler ace 2: An area scan camera for machine vision applications. (www.baslerweb.com).

[4]Intel NUC (NUC8i7BEK), 16 GB RAM, 1 TB HDD.

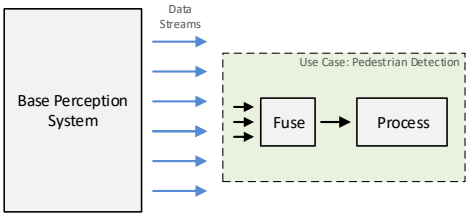


Fig. 7. High-level overview of the dataflow for the pedestrian detection use case. The input data of the use case is the temporally and spatially aligned sensor data from the base perception system.

The processing modules within the base-perception system (see Fig. 2) are implemented as ROS nodes (C++ and python). The reception nodes implement the custom low-level interface to the perception sensors, and provide the sensor data as standardized ROS messages to the remaining system. All data within the base-perception system (raw-data, pre-processed data, timestamps, etc.) is made available to other nodes and sub-systems via ROS messages. The ROS messages contain the estimated measurement timestamp, the position/orientation of the corresponding sensor as well as the raw/pre-processed data.

The timestamping approach presented in Section III is implemented in the trigger module (timestamp estimation) as well as in the reception modules (framedrop handling). The trigger module estimates a timestamp and assigns it to the current measurement number. This information (timestamp and measurement number) is then sent to the receive modules via standardized ROS messages. The receive modules store the tuple in a queue until the raw data of the corresponding measurement is received. The raw data is then translated into a standardized ROS message including the estimated measurement timestamp.

B. Use Case: Pedestrian Detection

We implemented an use case for the perception system, which performs basic pedestrian detection. Fig. 7 shows an overview of the sub-systems involved in this application. The base perception system provides pre-processed low-level data streams from multiple perception sensors as well as the spatial transformations between the sensor frames. The system handles the sensor synchronization and adds precise timing information to the data streams. Thus, the pedestrian detection sub-system does not have to perform any low-level data alignment and can directly start with fusing multiple data into a common representation. Afterwards, a detection algorithm is processing the fused data in order to detect pedestrians in the scene.

The pedestrian detection sub-system subscribes to the following data streams provided by the base platform:

- The undistorted image from the vision camera in the camera's optical frame.
- A list of detected radar targets (position, reflection power, Doppler velocity) in the radar sensor's frame.
- The down-sampled point clouds after pre-processing from the two ToF cameras in the ToF cameras' frames.
- The transformation tree containing the transformations between the single sensors' coordinate systems.

These data streams contain the acquisition timestamp, as well as the frame ID in their message header. The fusion module makes use of the timestamp in order to synchronize the data streams, according to their acquisition time. The fusion is performed each time, all inputs with the same timestamp are available to the module. Knowing the relative alignment between the sensors' coordinate frames at the acquisition time, the data can be transferred into a common representation.

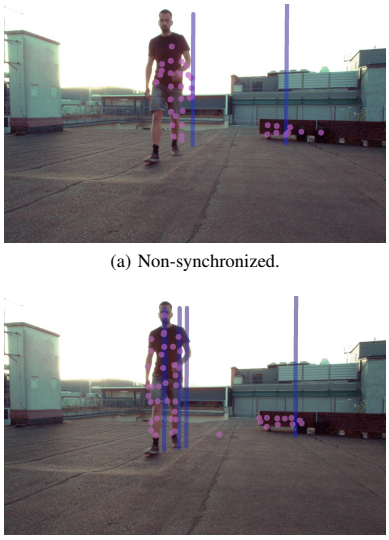
The detected targets from the radar sensor and the point cloud from the ToF sensor are projected onto the camera image plane by utilizing the camera's intrinsic parameters. Since the radar sensor is not capable to measure the elevation angle, the detected targets have unknown height and are projected onto the image as vertical lines. The point clouds from the ToF sensors is projected onto the image plane as single points. The output of the fusion module is an augmented five-channel image, containing two extra channels for the each projected range data type (ToF and radar). This image can then be used by a pedestrian detection module in order to detect pedestrians in the image.

The augmented image is utilized to perform pedestrian detection, with enhanced sensitivity in regions of interest which contain range data. We adapted a *Histogram of Oriented Gradients* (HOG) based algorithm, as presented in [11], in order to detect pedestrians. OpenCV provides a pre-trained *Support Vector Machine* (SVM) classifier, which utilizes HOG to perform pedestrian detection. This classifier is applied to the RGB image with a low threshold in order to identify regions of potential pedestrians (using a multi-scale sliding window). The depth data of the augmented image is then utilized to amplify bounding boxes which contain feasible range information. The resulting heat-map is then thresholded in order to detect and label pedestrians in the scene.

V. Results

In this section, we present the performance of the synchronization and hybrid timestamping approach presented in this paper. Additionally, we demonstrate the implementation of an use case, which utilizes the synchronized sensor data to perform pedestrian detection.

The importance of synchronized data for fusion is depicted in Fig. 8. Improperly timestamped data and underhanded framedrops can result in the fusion of data from different measurement times. The resulting misalignment leads to errors and significantly decreases a system's performance.



(a) Non-synchronized.

(b) Synchronized.

Fig. 8. Comparison of the fusion output for synchronized and non-synchronized data streams.

TABLE I
Round trip latency, evaluated for 15 000 round trip transfers.

Latency	Value
minimum	0.11 ms
maximum	8.19 ms
mean	1.65 ms
standard deviation	0.14 ms

The presented approach utilizes a microcontroller to generate the trigger signals, which is connected to the computation unit via USB. Thus, an inevitable delay is added when data is transferred from the microcontroller to the system's main processing unit. We measured the round trip latency of the USB/UART connection between the processing system and the microcontroller at a baud-rate of 115 200 Bd. For that purpose, 15 000 messages were sent to the trigger microcontroller and returned afterwards. Table I shows a statistical evaluation of the round trip latencies. As seen in the table, the average round trip delay is 1.65 ms, which leads to the an estimated one-way latency of 0.825 ms. The measurement acquisition duration of the sensors is in the range of 5 ms to 30 ms (depending on the sensor and the configuration). Thus, a timestamp precision in the range of single milliseconds is sufficient to be neglected in most use cases dealing with these heterogeneous sensors.

The presented timestamping approach is capable to work very robustly, even for high frame rates, if framedrops occur infrequently. Since a high workload leads to data inconsistencies, the maximum frame rate of the proposed system is limited by its framedrop handling (based on software timestamps). The system is capable to robustly provide synchronized and timestamped data at frame rates of up to 8 fps.

Fig. 9 shows an example of the pedestrian detection use

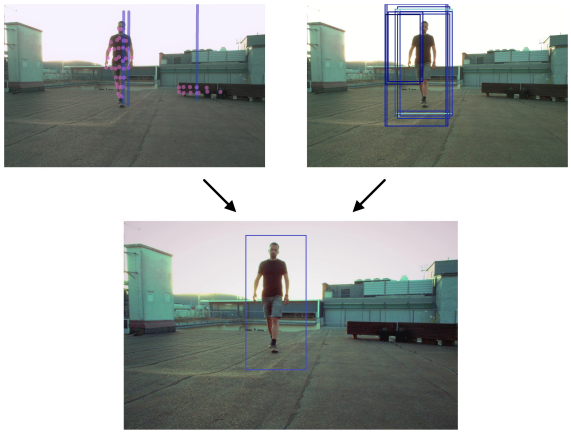


Fig. 9. Overview of the pedestrian detection use case's dataflow. The module utilizes the projected range data and potential pedestrian bounding boxes in order to robustly detect pedestrians in the scene.

case. The outdoor scene contains a pedestrian walking towards the perception platform on a flat ground. The range data from the radar and ToF sensors is projected onto the vision camera's image plane. As seen in the figure, the synchronized data streams are correctly projected onto the image plane (ToF: purple points, radar: blue lines). A HOG-based pedestrian classification is performed on the RGB image in order to detect bounding boxes potentially containing pedestrians. Based on the range data, the pedestrian detection module then creates a heat-map which amplifies feasible bounding boxes. In the final step, the enhanced heat-map is used to identify and label one or multiple pedestrians in the scene.

Since the sensors are synchronized, the approach also works in dynamic scenes, i.e., when the pedestrians and/or the system is moving. Non-synchronized sensors would sense a moving pedestrian at different positions in the image (see Fig. 8). Depending on the algorithm parameters, the pedestrian detection algorithm achieves a frame rate of 2-5 fps on the presented platform. The inclusion of range information enables a more sensitive pedestrian detection, increasing the detection rate. The detection performance especially benefits in open spaces, where all included range data originates from pedestrians. However, the approach struggles to detect pedestrians which are not directly facing the camera (e.g., laterally walking). The robustness of the detection could be further improved by re-training the HOG classifier with more general data, or by incorporating multiple consecutive images in the detection process (e.g., tracking).

VI. Conclusion

Synchronized and precisely timestamped sensor data is a key-enabler for the fusion of heterogeneous sensor data in multi-sensor perception systems. Improperly synchronized data can introduce effects like ghost-targets to the fusion module and decrease the quality of the perception system's output. In this work, we presented an approach to perform synchronization and timestamping of heterogeneous perception sensors. The presented method enables the synchronization of low-cost perception sensors, without built-in hardware synchronization. Additionally, we showed a way to recognize and handle framedrops, in order to ensure a robust operation.

We implemented the presented approach on a ROS-based perception platform deploying radar, ToF and vision sensors. The presented approach achieves a timestamping precision in the range of milliseconds, which is sufficient for a wide range of perception applications. Additionally, we demonstrated the fusion of the sensor data for the use case of pedestrian detection. Due to the hybrid timestamping, the range data is correctly projected onto the vision camera's image plane, enabling the detection of pedestrians in the scene.

The presented approach provides researchers a method to perform robust synchronization and timestamping with consumer-grade sensors. Non-synchronized, ROS-based perception platforms can utilize the approach in order to implement synchronization and to perform precise timestamping.

Acknowledgments

AutoDrive has received funding within the Electronic Components and Systems for European Leadership Joint Undertaking (ECSEL JU) in collaboration with the European Union's H2020 Framework Programme and National Authorities, under grant agreement number 737469.

References

[1] A. Mimouna, I. Alouani, A. B. Khalifa, Y. El Hillali, A. Taleb-Ahmed, A. Menhaj, A. Ouahabi, and N. E. B. Amara, "OLIMP: A heterogeneous multimodal dataset for advanced environment perception," *Electronics (Switzerland)*, vol. 9, no. 4, pp. 1–21, 2020.

[2] M. Brahmi, K. Schueler, S. Bouzouraa, M. Maurer, K. Siedersberger, and U. Hofmann, "Timestamping and latency analysis for multi-sensor perception systems," in *SENSORS, 2013 IEEE*, 2013, pp. 1–4.

[3] N. Kaempchen and K. Dietmayer, "Data synchronization strategies for multi-sensor fusion," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, vol. 85, no. 1, 2003, pp. 1–9.

[4] T. Huck, A. Westenberger, M. Fritzsche, T. Schwarz, and K. Dietmayer, "Precise timestamping and temporal synchronization in multi-sensor fusion," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 242–247.

[5] A. Westenberger, T. Huck, M. Fritzsche, T. Schwarz, and K. Dietmayer, "Temporal synchronization in multi-sensor fusion for future driver assistance systems," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2011, pp. 93–98.

[6] J. E. Guivant, S. Marden, and K. Pereida, "Distributed multi sensor data fusion for autonomous 3d mapping," in *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2012, pp. 1–11.

[7] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, 2008.

[8] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[9] A. Fregin, M. Roth, M. Braun, S. Krebs, and a. Flohr, "Building a computer vision research vehicle with ros," 2017.

[10] S. Schneider, M. Himmelsbach, T. Luettel, and H. Wuensche, "Fusing vision and lidar - synchronization, correction and occlusion reasoning," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 388–393.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 663 801 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
10.06.2020 Bulletin 2020/24

(21) Application number: **18211157.5**

(22) Date of filing: **07.12.2018**

(51) Int Cl.:
G01S 17/89 *(2020.01)* ***G01S 7/481*** *(2006.01)*
G01S 7/486 *(2020.01)* ***H04N 5/347*** *(2011.01)*
H04N 13/271 *(2018.01)*

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
KH MA MD TN

(71) Applicant: **Infineon Technologies AG**
85579 Neubiberg (DE)

(72) Inventors:
- **STEINBAECK, Josef**
  8010 Graz (AT)
- **PLANK, Hannes**
  8010 Graz (AT)
- **SCHOENLIEB, Armin**
  8054 Seiersberg-Pirka (AT)

(74) Representative: **2SPL Patentanwälte PartG mbB**
Postfach 15 17 23
80050 München (DE)

(54) **TIME OF FLIGHT SENSOR MODULE, METHOD, APPARATUS AND COMPUTER PROGRAM FOR DETERMINING DISTANCE INFORMATION BASED ON TIME OF FLIGHT SENSOR DATA**

(57) Examples relate to a method, an apparatus and a computer program for determining distance information based on Time of Flight (ToF) sensor data. The method comprises obtaining (110) the ToF sensor data. The method comprises determining (120) one or more saturated regions within the ToF sensor data. The method comprises determining (140) distance information for one or more boundary regions located adjacent to the one or more saturated regions based on the ToF sensor data. The method comprises determining (150) distance information for at least a part of the one or more saturated regions based on the distance information of the one or more boundary regions.

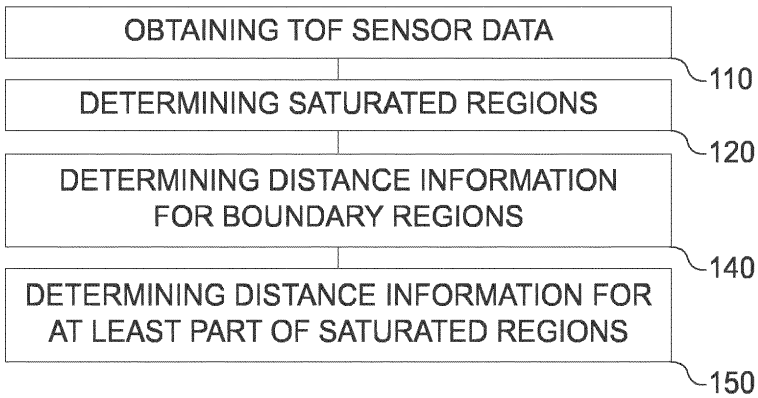


Fig. 1a

EP 3 663 801 A1

Bibliography

[1] B. Schoettle, "Sensor Fusion: A Comparison of Sensing Capabilities of Human Drivers and Highly Automated Vehicles," 2017.

[2] C. Urmson *et al.*, "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, vol. 23, no. 8, pp. 467–508, 2006.

[3] S. Thrun *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[4] M. Montemerlo *et al.*, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, September 2008.

[5] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, August 2008.

[6] J. Leonard *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, October 2008.

[7] J. Steinbaeck, C. Steger, E. Brenner, G. Holweg, and N. Druml, "Occupancy Grid Fusion of Low-Level Radar and Time-of-Flight Sensor Data," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, 2019, pp. 200–205.

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[9] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[10] C. Badue *et al.*, "Self-Driving Cars: A Survey," *Expert Systems with Applications*, 2020.

[11] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, 2020.

[12] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of Autonomous Car—Part I: Distributed System Architecture and Development Process," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131–7140, 2014.

[13] W. Shi, M. B. Alawieh, X. Li, and H. Yu, "Algorithm and hardware implementation for visual perception system in autonomous vehicle: A survey," *Integration*, vol. 59, pp. 148–156, 2017.

[14] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384–406, 2018.

[15] D. Gruyer, V. Magnier, K. Hamdi, L. Claussmann, O. Orfila, and A. Rakotonirainy, "Perception, information processing and modeling: Critical stages for autonomous driving applications," *Annual Reviews in Control*, vol. 44, pp. 323–341, 2017.

[16] L. Fridman *et al.*, "MIT Advanced Vehicle Technology Study: Large-Scale Naturalistic Driving Study of Driver Behavior and Interaction With Automation," *IEEE Access*, vol. 7, p. 102021–102038, 2019.

[17] F. Rosique, P. Navarro Lorente, C. Fernandez, and A. Padilla, "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research," *Sensors*, vol. 19, p. 648, February 2019.

[18] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2013, pp. 763–770.

[19] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making Bertha See," in *2013 IEEE International Conference on Computer Vision Workshops*, 2013, pp. 214–221.

[20] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An Open Approach to Autonomous Vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, November 2015.

[21] F. Naser *et al.*, "A parallel autonomy research platform," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 933–940.

[22] K. Burnett, A. Schimpe, S. Samavi, M. Gridseth, C. W. Liu, Q. Li, Z. Kroeze, and A. P. Schoellig, "Building a Winning Self-Driving Car in Six Months," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 9583–9589.

[23] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of Autonomous Car—Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 8, pp. 5119–5132, 2015.

[24] M. McNaughton, C. R. Baker, T. Galatali, B. Salesky, C. Urmson, and J. Ziglar, "Software infrastructure for an autonomous ground vehicle," *Journal of Aerospace Computing, Information, and Communication*, vol. 5, no. 12, pp. 491–505, 2008.

[25] J. Dickmann, N. Appenrodt, J. Klappstein, H. L. Bloecher, M. Muntzinger, A. Sailer, M. Hahn, and C. Brenk, "Making Bertha See Even More: Radar Contribution," *IEEE Access*, vol. 3, pp. 1233–1247, 2015.

[26] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," April 2018, pp. 287–296.

[27] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research*, vol. 3, 2014.

[28] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," 2019.

[29] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9552–9557, 2019.

[30] M. Chang *et al.*, "Argoverse: 3D Tracking and Forecasting With Rich Maps," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8740–8749.

[31] M. Meyer and G. Kuschk, "Automotive Radar Dataset for Deep Learning Based 3D Object Detection," in *2019 16th European Radar Conference (EuRAD)*, 2019, pp. 129–132.

[32] A. Mimouna, I. Alouani, A. Ben Khalifa, Y. El hillali, A. Taleb-Ahmed, A. Menhaj, A. Ouahabi, and N. ESSOUKRI BEN AMARA, "OLIMP: A Heterogeneous Multimodal Dataset for Advanced Environment Perception," vol. 9, pp. 1–24, March 2020.

[33] P. Sun *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2443–2451.

[34] B. Lamprecht, S. Rass, S. Fuchs, and K. Kyamakya, "Extrinsic Camera Calibration for an On-board Two-Camera System without overlapping Field of View," pp. 265–270, 2007.

[35] L. Heng, B. Li, and M. Pollefeys, "CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1793–1800, 2013.

[36] J. Domhof, J. F. P. Kooij, and D. M. Gavrila, "An Extrinsic Calibration Tool for Radar, Camera and Lidar," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8107–8113.

[37] R. Unnikrishnan and M. Hebert, "Fast Extrinsic Calibration of a Laser Rangefinder to a Camera," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., July 2005.

[38] M. Brahmi, K. Schueler, S. Bouzouraa, M. Maurer, K. Siedersberger, and U. Hofmann, "Timestamping and latency analysis for multi-sensor perception systems," in *SENSORS, 2013 IEEE*, 2013, pp. 1–4.

[39] T. Huck, A. Westenberger, M. Fritzsche, T. Schwarz, and K. Dietmayer, "Precise timestamping and temporal synchronization in multi-sensor fusion," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 242–247.

[40] A. Westenberger, T. Huck, M. Fritzsche, T. Schwarz, and K. Dietmayer, "Temporal synchronization in multi-sensor fusion for future driver assistance systems," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2011, pp. 93–98.

[41] A. Fregin, M. Roth, M. Braun, S. Krebs, and a. Flohr, "Building a Computer Vision Research Vehicle with ROS," September 2017.

[42] A. Harrison and P. Newman, "TICSync: Knowing when things happened," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 356–363.

[43] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[44] H. Hu and G. Kantor, "Efficient Automatic Perception System Parameter Tuning On Site without Expert Supervision," in *Proceedings of (CoRL) Conference on Robot Learning*, November 2017, pp. 57–66.

[45] M. Thonnat, S. Moisan, and M. Crubézy, "Experience in Integrating Image Processing Programs," in *Computer Vision Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 200–215.

[46] V. Klös, T. Göthel, and S. Glesner, "Adaptive Knowledge Bases in Self-Adaptive System Design," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, 2015, pp. 472–478.

[47] D. P. Chau, M. Thonnat, F. Brémond, and E. Corvée, "Online parameter tuning for object tracking algorithms," *Image and Vision Computing*, vol. 32, no. 4, pp. 287–302, 2014.

[48] N. Hochgeschwender, M. A. Olivares-Mendez, H. Voos, and G. K. Kraetzschmar, "Context-based selection and execution of robot perception graphs," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, 2015, pp. 1–4.

[49] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of machine learning research*, vol. 13, no. 2, pp. 281–305, February 2012.

[50] J. Kim, Y. Cho, and A. Kim, "Exposure Control Using Bayesian Optimization Based on Entropy Weighted Image Gradient," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 857–864, 2018.

[51] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[52] F. Castanedo, "A review of data fusion techniques," *The Scientific World Journal*, 2013.

[53] R. C. Luo, C. C. Chang, and C. C. Lai, "Multisensor Fusion and Integration: Theories, Applications, and its Perspectives," *IEEE Sensors Journal*, vol. 11, no. 12, pp. 3122–3138, 2011.

[54] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and M. Lienkamp, "A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection," in *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, October 2019, pp. 1–7.

[55] R. Varga, A. Costea, H. Florea, I. Giosan, and S. Nedevschi, "Super-sensor for 360-degree environment perception: Point cloud segmentation using image features," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–8.

[56] D. Göhring, M. Wang, M. Schnürmacher, and T. Ganjineh, "Radar/Lidar sensor fusion for car-following on highways," *The 5th International Conference on Automation, Robotics and Applications*, pp. 407–412, 2011.

[57] S. Chadwick, W. Maddern, and P. Newman, "Distant Vehicle Detection Using Radar and Vision," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8311–8317.

[58] F. Garcia, D. Martin, A. de la Escalera, and J. M. Armingol, "Sensor Fusion Methodology for Vehicle Detection," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 123–133, 2017.

[59] A. García-Moreno, J. Gonzalez-Barbosa, J. B. Hurtado-Ramos, and F. Ornelas-Rodriguez, "Mobile remote sensing platform: An uncertainty calibration analysis," in *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2014, pp. 64–69.

[60] Z. Wang, Y. Wu, and Q. Niu, "Multi-Sensor Fusion in Automated Driving: A Survey," *IEEE Access*, vol. 8, pp. 2847–2868, 2020.

[61] P. Radecki, M. Campbell, and K. Matzen, "All Weather Perception: Joint Data Association, Tracking, and Classification for Autonomous Ground Vehicles," *arXiv preprint*, 2016.

[62] O. Ludwig, C. Premebida, U. Nunes, and R. Araújo, "Evaluation of Boosting-SVM and SRM-SVM cascade classifiers in laser and vision-based pedestrian detection," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 1574–1579.

[63] N. Engelhardt, R. Pérez, and Q. Rao, "Occupancy Grids Generation Using Deep Radar Network for Autonomous Driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2866–2871.

[64] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint*, 2018.

[65] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[66] C. Gálvez del Postigo Fernández, "Grid-based multi-sensor fusion for on-road obstacle detection: Application to autonomous driving," *Master's thesis*, 2015.

[67] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications," vol. 5, no. 2, pp. 829–846, 2018.

[68] T.-N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M.-M. Meinecke, "Stereo-camera-based urban environment perception using occupancy grid and object tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 154–165, 2011.

[69] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool, "Dynamic 3D Scene Analysis from a Moving Vehicle," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[70] H. Xue, H. Fu, and B. Dai, "IMU-Aided High-Frequency Lidar Odometry for Autonomous Driving," *Applied Sciences*, vol. 9, no. 7, 2019.

[71] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[72] D. Wakabayashi, "Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam," *The New York Times*, March 2018, https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html.

[73] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[74] L. Zhang, L. Lin, X. Liang, and K. He, "Is Faster R-CNN Doing Well for Pedestrian Detection?" pp. 443–457, 2016.

[75] L. Pang, Z. Cao, J. Yu, S. Liang, X. Chen, and W. Zhang, "An Efficient 3D Pedestrian Detector with Calibrated RGB Camera and 3D LiDAR," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 2902–2907.

[76] M. Lindner, "Calibration and Realtime Processing of Time-of-Flight Range Data," PhD Thesis, Computer Graphics Group, University of Siegen, December 2010.

[77] T. Möller, H. Kraft, J. Frey, M. Albrecht, and R. Lange, "Robust 3D Measurement with PMD Sensors," *Range Imaging Day, Zürich*, vol. 7, no. 8, 2005.

[78] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *IEEE Journal of Quantum Electronics*, vol. 37, no. 3, pp. 390–397, March 2001.

[79] A. P. P. Jongenelen, D. G. Bailey, A. D. Payne, D. A. Carnegie, and A. A. Dorrington, "Efficient FPGA implementation of homodyne-based time-of-flight range imaging," *Journal of Real-Time Image Processing*, vol. 7, no. 1, pp. 21–29, March 2012.

[80] Sense Photonics, "Sense Photonics Introduces Osprey, the First Modular FLASH LiDAR for Autonomous Vehicles," Press Release, January 2020, https://sensephotonics.com/osprey-modular-flash-lidar-press-release.

[81] Infineon Technologies, "State-of-the-art photography results and immersive AR experiences: Infineon and pmd offer 3D-imager with longest range in the market," Press Release, October 2020, https://www.infineon.com/cms/en/about-infineon/press/press-releases/2020/INFPSS202010-005.html.

[82] J. Mure-Dubois and H. Hügli, "Real-time scattering compensation for time-of-flight camera," in *Proceedings of the ICVS Workshop on Camera Calibration Methods for Computer Vision Systems*, 2007.

[83] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Next generation radar sensors in automotive sensor fusion systems," in *2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2017, pp. 1–6.

[84] F. Roos, J. Bechter, C. Knill, B. Schweizer, and C. Waldschmidt, "Radar Sensors for Autonomous Driving: Modulation Schemes and Interference Mitigation," *IEEE Microwave Magazine*, vol. 20, no. 9, pp. 58–72, 2019.

[85] I. Bilik, O. Longman, S. Villeval, and J. Tabrikian, "The Rise of Radar for Autonomous Vehicles: Signal Processing Solutions and Future Research Directions," *IEEE Signal Processing Magazine*, vol. 36, no. 5, pp. 20–31, 2019.

[86] M. Parker, "Automotive Radar," in *Digital Signal Processing 101*, second edition ed., M. Parker, Ed. Newnes, 2017, ch. 20, pp. 253–276.

[87] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Design of a Low-Level Radar and Time-of-Flight Sensor Fusion Framework," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 268–275.

[88] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Short-range FMCW monopulse radar for hand-gesture sensing," *IEEE National Radar Conference - Proceedings*, no. June, pp. 1491–1496, 2015.

[89] Infineon Technologies, "Infineon 3D Image Sensor, IRS10x0C," Product Brief, May 2013.

[90] J. Steinbaeck, N. Druml, A. Tengg, C. Steger, and B. Hillbrand, "Time-of-Flight Cameras for Parking Assistance: A Feasibility Study," in *2018 12th International Conference on Advanced Semiconductor Devices and Microsystems (ASDAM)*, 2018, pp. 1–4.

[91] L. P. Peláez, M. E. V. Recalde, E. D. M. Muñóz, J. M. Larrauri, J. M. P. Rastelli, N. Druml, and B. Hillbrand, "Car parking assistance based on Time-or-Flight camera," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, June 2019, pp. 1753–1759.

[92] H. Plank, J. Steinbaeck, N. Druml, C. Steger, and G. Holweg, *Localization and Context Determination for Cyber-physical Systems based on 3D Imaging.* IGI Global, 2018, pp. 1–26.

[93] X. Gao, G. Xing, S. Roy, and H. Liu, "Experiments with mmWave Automotive Radar Test-bed," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 1–6.

[94] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, February 1992.

[95] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[96] L. Stanislas and T. Peynot, "Characterisation of the Delphi Electronically Scanning Radar for Robotics Applications," in *Australasian Conference on Robotics and Automation (ACRA 2015)*, vol. 2, 2015, p. 4.

[97] J. Peršić, I. Marković, and I. Petrović, "Extrinsic 6DoF calibration of 3D LiDAR and radar," in *2017 European Conference on Mobile Robots (ECMR)*, September 2017, pp. 1–6.

[98] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[99] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[100] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," 2001.

[101] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13).* Springer, July 2014.

[102] J. Steinbaeck, C. Steger, E. Brenner, and N. Druml, "A Hybrid Timestamping Approach for Multi-Sensor Perception Systems," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 447–454.

[103] A. Koubaa, *Robot Operating System (ROS): The Complete Reference.* Springer, 2017, vol. 1.

[104] J. Steinbaeck, "Integration of a Time-of-Flight 3D Camera into a Mobile Densing Platform," Master's thesis, Graz University of Technology, 2016.

[105] J. Steinbaeck *et al.*, "ACTIVE - Autonomous Car to Infrastructure Communication Mastering Adverse Environments," in *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2019, pp. 1–6.

[106] J. Steinbaeck, A. Tengg, G. Holweg, and N. Druml, "A 3D Time-of-Flight Mixed-Criticality System for Environment Perception," in *2017 Euromicro Conference on Digital System Design (DSD)*, 2017, pp. 368–374.

[107] J. Steinbaeck, A. Strasser, C. Steger, E. Brenner, G. Holweg, and N. Druml, "Context-Aware Sensor Adaption of a Radar and Time-of-Flight Based Perception Platform," in *2020 IEEE Sensors Applications Symposium (SAS)*, 2020, pp. 1–6.

[108] J. Steinbaeck, H. Plank, and A. Schoenlieb, "Time of Flight Sensor Module, Method, Apparatus and Computer Program for Determining Distance Information based on Time of Flight Sensor Data," European Patent Application EP 3 663 801 A1, 2018.

[109] V. C. Chen, F. Li, S. . Ho, and H. Wechsler, "Micro-Doppler effect in radar: phenomenon, model, and simulation study," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 2–21, 2006.