Master Thesis

Efficient Implementation of Novel Time-of-Flight Signal Processing Algorithms

Daniela Freidl

Graz University of Technology

Institute for Technical Informatics
Head: Univ.-Prof. Dipl.-Inform. Dr.sc.ETH Kay Uwe Römer

in cooperation with

Infineon Technologies Austria AG
Graz, Austria

Advisor: Dipl.-Ing. Dr. techn. Christoph Steiner
Dipl.-Ing. Dr. techn. Markus Dielacher
Univ.-Ass. Dipl.-Ing. Dr. techn. Christian Steger

Assessor: Univ.-Ass. Dipl.-Ing. Dr. techn. Christian Steger

Graz, December 2013

Zusammenfassung

In den vergangenen Jahren hat sich der Markt für 3D-Sensoren stark vergrößert. Dies ist vorallem durch die vermehrte Automatisierung und den verstärkten Fokus auf einfach zu bedienende Eingabemethoden für Computer und sonstige elektronische Geräte bedingt.

Die Vermessung der 3D-Szenen ist mit unterschiedlichen Methoden möglich, wobei im Zuge dieser Masterarbeit das Time-of-Flight-Verfahren herangezogen wird. Diese Methode verwendet eine Beleuchtungseinheit, welche moduliertes Licht aussendet, und einem Bildsensor, welcher die Laufzeit des Lichtes misst. Aufgrund der unterschiedlichen Distanzen einer Szene kommt es je Pixel zu unterschiedlichen Laufzeiten. Mit Hilfe dieser Laufzeiten wird anschließend auf die Distanz einer Szene zurückgerechnet. Die gesamte Szene wird mithilfe eines Pixel-Arrays zur selben Zeit aufgenommen. Es werden jedoch mehrere Einzelaufnahmen benötigt, mindestens zwei, um die Tiefenkarte der Szene zu berechnen.

Die Berechnung der Tiefenkarte erfolgt am PC. Aufgrund der hohen Auflösung muss daher eine erhebliche Datenmenge vom Bildsensor zum PC übertragen werden. Die hohen Datenmengen führen einerseits zu einer Überlastung der Schnittstelle und andererseits zu einem nicht vernachlässigbaren Rechenaufwand. Weiters benötigt die Berechnung am PC zusätzliche Rechenzeit, was zu einer merklichen Latenz führt.

Ziel dieser Masterarbeit ist es, ein System zu entwerfen und zu implementieren, welches die effiziente Tiefenberechnung einer aufgenommenen Szene in Hardware durchführt. Das System soll zwei Algorithmen unterstützen: den 4-Phasen Stand der Technik Algorithmus und den vor Kurzem erarbeiteten 3-Phasen-Algorithmus. Das System ist bezüglich Laufzeit optimiert worden, um einen maximalen Datendurchsatz zu ermöglichen.

Zusätzlich ist das System in der Lage, mit unterschiedlichen Frame-Dimensionen umzugehen, solange vorgegebene Kriterien erfüllt sind. Diese Skalierbarkeit ist ein Alleinstellungsmerkmal und unterscheidet das entworfene und implementierte System klar von ähnlichen Projekten.

Die praktische Implementierung zeigt, dass das entwickelte System einen geringen Distanzfehler aufweist, welcher auf die verminderte Berechnungsgenauigkeit in Hardware zurückzuführen ist. Das Ergebnis ermöglicht außerdem eine starke Erhöhung der lieferbaren Tiefenbilder mit geringer Latenzzeit. Insbesondere der 3-Phasen-Algorithmus liefert sehr genaue Ergebnisse und ist weniger anfällig für bekannte Störungen.

Abstract

Over the last years, the market for 3D imaging has grown strongly. This is a result of increased automation and stronger focus on easy-to-handle input methods for computers and other electronic devices.

Different methods have been developed to measure 3D scenes. In the course of this master thesis, the Time-of-Flight method will be examined. This method uses an illumination unit, which emits modulated light and an image sensor, which measures the time of flight of the modulated light. The different distances of a captured scene result in a difference in the measured time of flight for each pixel. The elapsed time is afterwards used to calculate the distance of scene. The whole scene is captured at the same time by an array of pixels. Multiple raw images, at least two, are needed to calculate the depth map of a scene.

The calculation of the depth map is done on a PC. Due to the high resolution, a high amount of data has to be transferred from the image sensor to the PC. This mass of data leads on the one hand to an overloading of the interface and on the other hand to a significant processing effort. Furthermore, the calculation on the PC consumes computing time, which leads to an observable latency.

The goal of this master thesis was to design and implement a system which performs the efficient depth calculation of a captured scene in hardware. The system aims to support two algorithms: a 4-phase state-of-the-art algorithm and a newly designed 3-phase algorithm. The system has been optimized regarding its performance in order to facilitate a maximal data through-put.

Additionally, the system is flexible regarding arbitrary frame dimensions, as long as certain criterias are fulfilled. This scalability is a unique feature and distinguishes the designed and implemented system from related projects.

The practical implementation shows, that the developed system has a small distance error, which is due to the limited precision at the hardware. The solution enables a significant increase of delivered depth images at small latency. Especially the 3-phase algorithm returns precise distance results and is less prone to known noise and errors.

Statutory Declaration

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 marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Acknowledgements

This thesis was composed at the Institute for Technical Informatics at the Technical University Graz in 2013. At the beginning of this thesis I want to seize the chance to thank all the people, who supported me during my studies and especially during my thesis.

I would like to extend my thanks to the Institute of Technical Informatics, particularly Ass. Prof. Dipl.-Ing. Dr. Christian Steger for his tireless efforts and assistance. Furthermore, I want to thank my team from Infineon, department "Automotive Sense and Control". My special thanks go to my advisors Dr. Christoph Steiner and Dr. Markus Dielacher for their patience and assistence to improve the quality of my work.

Special thanks to my family, who were a bastion of calm and provided constant aid during my academic studies. Without them I would not have reached this mile stone in my life.

I also would like to thank my dear friends, who supported me through rough times and shared my laughter during gentle times, namely Kathy and Vroni.

Graz, December 2013 Daniela Freidl

Contents

List of Figures

List of Tables

Nomenclature

ARM	Advanced RISC Machine
ASIC	Application-Specific Integrated Circuit
CLPD	Complex Programmable Logic Device
CORDIC	COordinate Rotation DIgital Computer
CSI	Camera Serial Interface
DDR SDRAM	Double Data Rate Synchronous Dynamic Random Access Memory
DMA	Direct Memory Access
FIFO	First-In, First-Out
FPGA	Field Programable Gate Array
FPN	Fixed Pattern Noise
FPPN	Fixed Pattern Phase Noise
fps	frames per second
f_{mod}	Modulation Frequency
GPIF	General Programable Interface
GUI	Graphical User Interface
HDL	Hardware Description Language
IDE	Integrated Development Environment
IO	Input/Output
IP	Intellectual Property
IR	Infra-Red
ISE	Integrated Software Environment
I^2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
LUT	Look-Up Table
MCB	Memory Controller Block
MIG	Memory Interface Generator
PIF	Parallel InterFace
PLL	Phase-Locked Loop
PMD	Photonic Mixing Device
RGB	Red Green Blue
RISC	Reduced Instruction Set Computing
RX	Receive
SNR	Signal Noise Ratio
SPI	Serial Peripheral Interface
t_{ill}	Illumination Time
ToF	Time-of-Flight
TX	Transmit
USB	Universal Serial bus
VHDL	Very-High-Speed Integrated Circuits HDL
3D	3-Dimensional

— CHAPTER ONE —

Introduction

In the last decade 3D imaging has gained in importance: various applications require the accurate determination of depth maps. Especially automation processes require precise distance information for an improved cycle. Previously 2D imaging was frequently used to recognize objects and features. Nowadays 3D imaging aims to facilitate or replace 2D imaging in different areas. The depth information gained by 3D imaging simplifies the interpretation of scenes.

Often when referring to 3D images, only 2.5D is achieved. This limitation is due to the fixed position of the camera system: only distance information of the front side of the object can be obtained.

In order to fulfill these increasing demands for 3D representations, various techniques have evolved. Figure 1.1 shows a categorization of different methods. Contact-less 3D imaging can be done with the aid of microwaves, ultrasonic waves, and light waves. Ultrasonic waves and micro waves are used in GPS and radar applications and will not be examined in the further investigation. This thesis will focus on optical waves with a wave length of 0.5 - 1 μm. The advantage of optical waves is their wide angular resolution. The following optical imaging methods are current subjects of research and will be analyzed regarding their advantages and disadvantages:

- Triangulation,
- Interferometry, and
- Time-of-Flight (ToF).

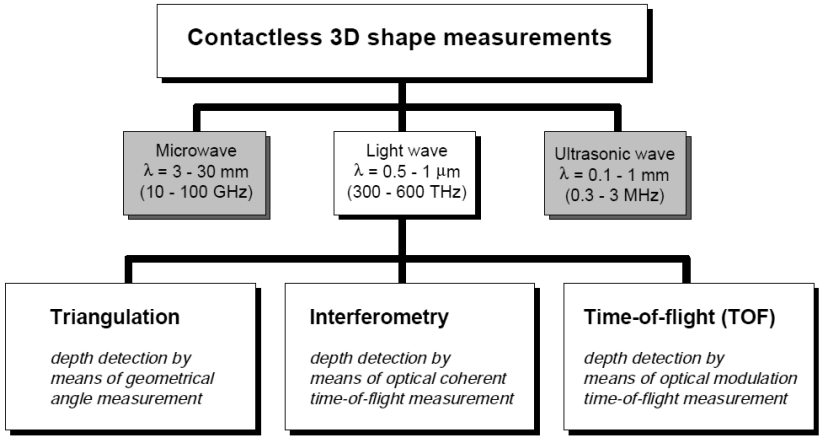


Figure 1.1: 3D measurement methods [Schwarte et al., 1999].

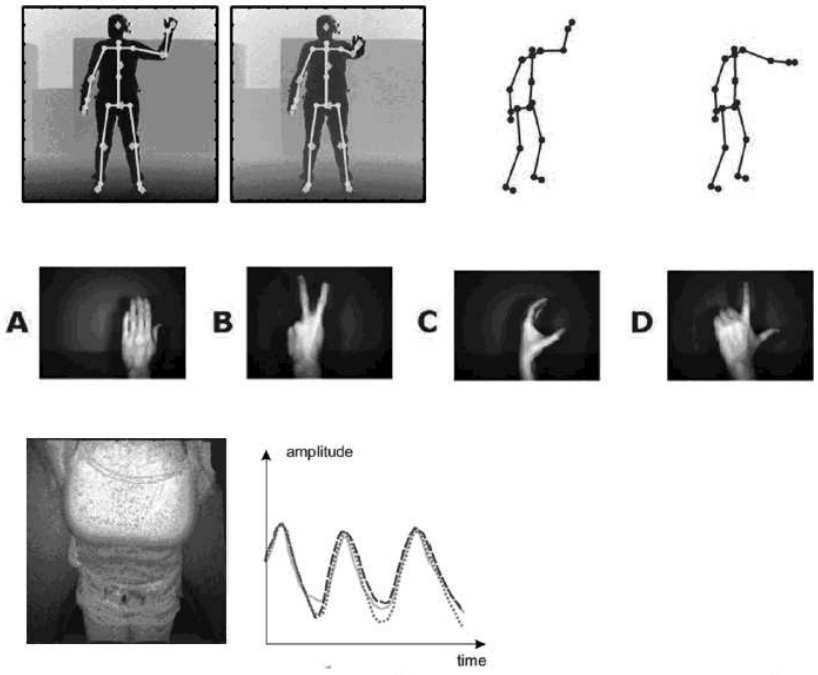


Figure 1.2: Examples of 3D imaging applications [Schwarz et al., 2011], [Kollorz et al., 2008], [Penne et al., 2008].

In general, the field of applications stretches from industrial to private applications, e.g. medical applications, human-computer interfaces, and automotive security. Figure 1.2 shows three examples for 3D imaging applications: (a) people tracking, (b) gesture recognition, and (c) medical applications like respiration monitoring.

The following chapter will shortly list the motivation for this research, define the goals and objectives and give an outline of the document's structure.

1.1 Motivation

The demand for real-time 3D applications has increased, which leads to the challenge to deliver highly accurate distance images within a continuously shrinking time slot. The current speed of 3D system is around 20 frames per second (fps). It is almost needless to say that this speed is highly limiting the possibility to capture motion. Within two captured images 2 ms pass. Considering a human gesture, which is performed with 10m/s, this will lead to a traveled distance of 2 cm just within two images. The speed-up of the 3D capturing procedure is necessary to provide real-time information.

With this goal in mind, this thesis will focus on ToF solutions. ToF cameras can deliver 3D images with high speed and are relatively cheap to fabricate. Their characteristics promise an adequate solution for fast applications and are only limited by the relatively low resolution.

The ToF system requires the capturing of multiple raw frames - in general four raw frames for one resulting distance image. The conjunction to receive the distance image is generally done in software. This system design leads to multiple disadvantages:

- all raw frames have to be transmitted to the PC, where the depth calculation is done in software,

- the interfaces between camera and PC are highly occupied, since all the raw data has to be proceeded within a certain time,
- the calculation on the PC requires processor resources and thus leads to a slow-down of other applications, and
- the calculation in software is time-consuming and leads to latency.

The proposed solution will focus on an implementation in hardware in order to eliminate or decrease the above mentioned disadvantages: within a smaller latency time an higher rate of distance images can be delivered. The goals and objectives of this thesis are listed in the following section.

1.2 Objectives

This thesis and its corresponding practical work deals with the research field of novel ToF algorithms and aims to improve their realization. The main focus is the efficient implementation of two ToF algorithms on a Field-Programmable Gate Array (FPGA) and their comparison in terms of speed, precision and error rate. Therefore, the work packages will include the following:

- research of state-of-the-art projects,
- definition of the system requirements,
- design of an appropriate system,
- implementation of ToF algorithms in hardware,
- verification of the implementation,
- improvement of the algorithms in terms of run-time and area consumption, and
- performing of various experiments to give a qualitative comparison of the two focused algorithms.

The realized system shall be implemented modularly to increase the degree of re-usability. The system has to be a general solution, which can deal with multiple settings. The initiation and start of the system is performed by the PC and the calculation can be done as soon as raw images are received from the camera. The hardware system acts as interface between the camera and the PC. The structure of this document is explained in the following section.

1.3 Outline

After this brief introduction regarding the motivation and objectives of this thesis, an overview of the theoretical background of ToF, the necessary components and a comparison of different 3D imaging techniques is given in Chapter 2. Furthermore some state-of-the-art solutions for similar 2D and 3D imaging projects are shown in detail. Chapter 3 contains details regarding the requirements of the current system and explains the design of the new system. The system is analyzed regarding its environment, the static and dynamic design and further non-functional requirements. In Chapter 4 the implementation details are shown: the necessary interfaces and the system implementation. Furthermore, an overview of the performed feasibility study is given and the development environment is explained. Moreover, the verification process is explained and the implementation results are shown. Chapter 5 lists the executed experimental results for the implemented and verified system. The conclusion and future work is summarized in Chapter 6.

— CHAPTER TWO —

Literature

2.1 Time-of-Flight Fundamentals

ToF is, as mentioned in Chapter 1, an optical 3D imaging technique. A very basic direct ToF system can be seen in Figure 2.1. The main principle works as follows: A light source emits modulated light, which is reflected by an object and afterwards captured by a sensor.

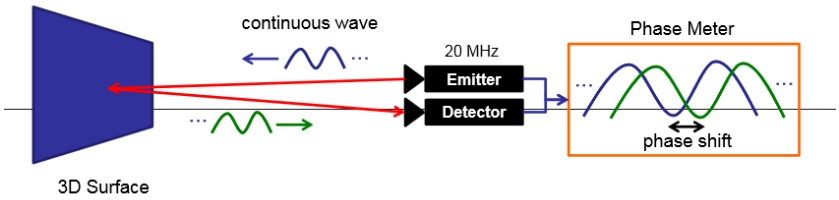


Figure 2.1: General principle of the direct ToF algorithm [Lange, 2000].

The ToF measurement principle can be sub-divided into direct ToF and indirect ToF. While direct ToF algorithms are measuring the elapsed time from emitting the signal until receiving its reflection, indirect ToF algorithms are measuring the phase difference between emitted and received signal for the distance calculation. For the sake of convenience, this thesis refers to the indirect ToF procedure as ToF. Other 3D imaging techniques can be found in Chapter 2.1.2.

2.1.1 Principle

In direct ToF systems the time delay τ, which represents the time needed by the light to cover the distance, is measured and used to calculate the true distance d (compare Equation 2.1) [Schneider, 2003]. c_0 represents hereby the speed of light:

$$d = \frac{\tau \cdot c_0}{2} \tag{2.1}$$

In indirect ToF systems the principle is adapted to use the phase difference $\Delta\varphi$ between emitted and received signal for the distance calculation (Figure 2.2). Researchers [Lange, 2000] have shown that the measuring devices do not need the same accuracy level as direct ToF measuring systems.

In Equation 2.2 the relation between phase difference $\Delta\varphi$ and time delay τ is shown, f_M represents the modulation frequency.

$$\tau = \frac{\Delta\varphi}{2\pi \cdot f_M} \tag{2.2}$$

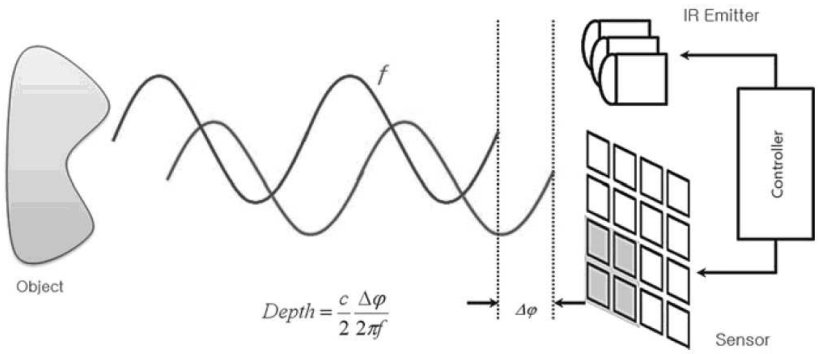


Figure 2.2: General principle of the indirect ToF algorithm [Hansard et al., 2012].

An outline of the detailed procedure can be found below.

ToF Capturing Procedure

As already mentioned, the system contains a light emitter which emits modulated light with a frequency around 5-130 MHz [Buttgen et al., 2006], [Microsoft, 2013]. These emitters are in general Light-Emitting Diodes (LEDs) or lasers which are radiating light in the infra-red (IR) range. The reason for using the IR range is that it is not visible to the human eye. The capturing of the reflected signal is done by a two-dimensional array of sensing pixels, which are working simultaneously. Each pixel measures the arriving irradiation during a defined time window, which is a multiple of the modulation period. The number of captured periods leads to a trade-off between accuracy and speed. The union of all pixels results in a raw image Y.

In a captured scene not only the distance d is unknown, but also the reflectivity and intensity of the object, which results in a number of scene unknowns. In order to unambiguously determine the distance, the number of images needs to be equal or higher than the number of scene unknowns. Theoretically two images would be sufficient to unambiguously determine the distance, but most of the state-of-the-art methods are using four images [Schmidt, 2011].

The four pictures are determined through equidistant phase shifting of the reference signal. In Figure 2.3 [Hansard et al., 2012] the ToF process from the generation of the signal until the final processing to achieve the depth value of a pixel can be seen.

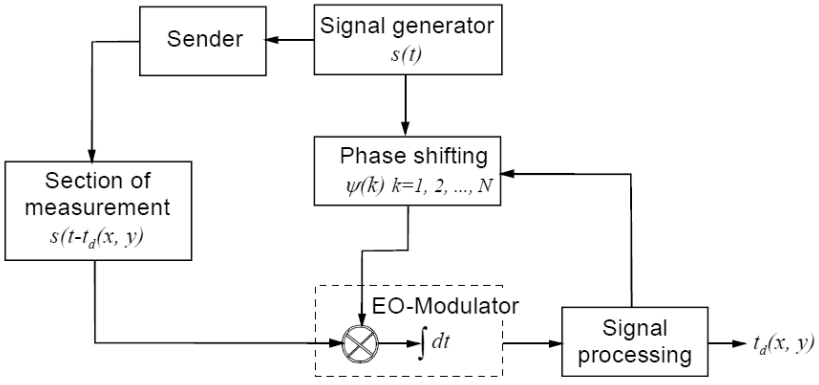


Figure 2.3: Simplified block diagram of the ToF algorithm [Luan, 2001].

The determination of the correlated signal is done for each pixel separately. Each pixel of the pixel-array represents a sensors, in the considered case with Photonic Mixing Devices (PMD) [PMDTechnologies, 2012]. Details about the behavior of this devices can be found in Section 2.1.1.

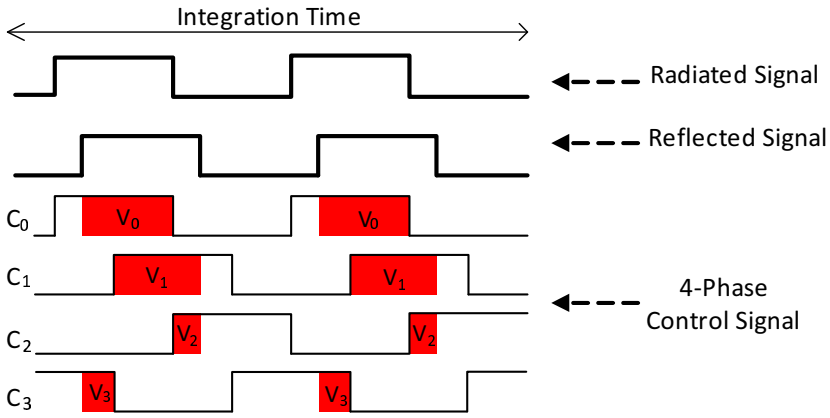


Figure 2.4: Representation of the ToF 4-phase capturing [Hansard et al., 2012].

The correlation value differs depending on the phase shifted reference signal (compare Figure 2.4). Every phase shift ψ_i of the reference signal produces a corresponding correlation value $V_i \; \forall i = 0, .., N-1$. These values are used to calculate the phase delay φ with the Equation 2.3.

$$\varphi = \arctan\left(\frac{V_3 - V_1}{V_2 - V_0}\right) \tag{2.3}$$

The sensor values can also be used to calculate the amplitude A (compare Equation 2.4).

$$A = \frac{\sqrt{(V_3 - V_1)^2 + (V_2 - V_0)^2}}{2} \tag{2.4}$$

PMD

One possibility for sensing the reflected irradiation and correlating it with the original signal is the PMD. In this chapter an overview of the principles is given, details can be found in literature [Luan, 2001].

The main advantage of the PMD technology is that it does not need complex mixing circuits to calculate the exact phase delay φ. Figure 2.5 shows its channel structure, including the light sensitive zone, the inner layers and the readout circuit. The PMD is a semiconductor device which detects and mixes the phase delay at the same time. For this purpose it uses the inner photo-electronic effect: At the surface arriving photons are split up into electron-hole pairs. The corresponding electrons are moving into either bucket a or bucket b. The ruling which bucket is selected is managed by the modulation electrons. These are modulated with the frequency of the emitted signal. Through this potential gradient the electrons are added up in the zones a and b. The frequent shift of the potential leads to a spreading of the electrons which represents the correlation of the emitted and the received signal. After a certain amount of cycles the final photo current is read out through the readout electronic circuit. This value represents the cross-correlation and varies depending on the phase difference φ. The higher the integration time (the more cycles), the higher is the Signal-Noise-Ratio (SNR).

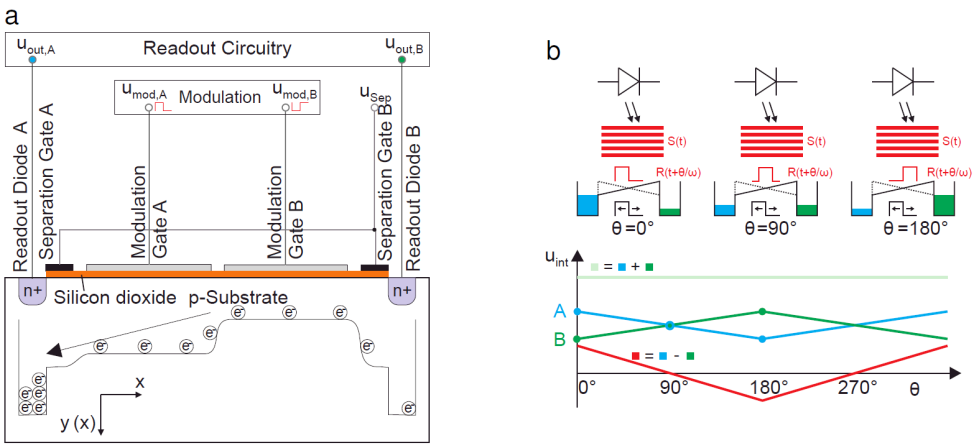


Figure 2.5: The simplified channel structure of the PMD [Schmidt, 2008].

Besides ToF, multiple other approaches of 3D imaging exist. A extract of the most common methods is given in the next section.

2.1.2 Further Optical 3D Imaging Techniques

Researchers have done investigations regarding the different techniques and compared these with focus on possible fields of applications [Foix et al., 2010], [Ghobadi et al., 2006] and [Einramhof et al., 2007].

Stereo Vision

Stereo vision systems are imitating the human eye: the systems use two cameras to capture two images from different perspectives. Poggio and Marr investigated the stereo vision concept in detail [Marr and Poggio, 1979], [Poggio and Poggio, 1984]. These images are combined afterwards and used to calculate the 3D values of an object. The advantage is that no active light source is necessary and a high resolution is possible. The main disadvantage is that the solving of the correspondence problem is computationally intensive and requires fast and powerful processing units. Also a baseline between the two cameras is needed to increase the accuracy. Therefore, the size of the camera cannot fall below a certain level. Other disadvantages are shadowing effects due to the different camera angle and triangulation correspondence problems.

Laser

Laser-based systems are using a laser to illuminate the object, capture its reflection and use it for the distance calculation. The benefits of this system are its high resolution and accuracy. The drawbacks are high costs and low speed due to moving parts which have to control the laser beam to cover the whole scene. The capturing is done line by line until the whole scene is captured. The drawback of this method is that the distance measurement is executed for one pixel after another. Complex mechanics is is needed to perform the capturing. Movement of the object during scanning leads to erroneous results. Furthermore, the device is sensitive against vibration and eye safety is not given, which makes it not suitable for many human involving applications.

Structured Light

This method uses a light source to project patterns or lines onto an object. A camera is capturing an image afterwards and interpreting the intensity gradient. The object-specific distraction of the pattern is used to gain the depth information. The accuracy of this method is limited by the high-precision projection unit [Spickermann, 2010].

ToF

The working principle of ToF has already been explained in the above sections, further details are explained in literature [Lange et al., 1999], [Kolb et al., 2008], and [Heinol, 2001]. Its main advantage is speed and decreasing manufacturing costs. A current disadvantage is the low resolution and the necessary calibration.

Differences	**ToF cameras**	**Stereo vision**	**Structured light**
Correspondence problem	No	Yes	Yes
Extrinsic calibration	No, when used alone	Yes	Yes
Auto illumination	Yes	No	Yes
Untextured surfaces	Good performance	Bad performance	Good performance
Depth range	0.3 ÷ 7.5 m	Base-line dependent	Light-power dependent
Image resolution	Up to 288 x 352	High resolution. Camera dependent	
Frame rate	Up to 100 fps.	Typically 25 fps. Camera dependent	

Table 2.1: An overview of the different optical 3D imaging techniques and their benefits and disadvantages [Foix et al., 2011], [Infineon, 2013], [Schaller, 2011].

In Table 2.1 a summary of the benefits and disadvantages of each technique is given. In stereo vision systems complex calculations have to be done to match the two captured images. A similar calculation is necessary for structured light, since the emitted light pattern has to be matched to the corresponding captured image. This computations are summed up in the terminus "Correspondence problem". In ToF systems, this mapping is not necessary. Illumination refers to the need for light emitting devices, which are necessary in ToF and structured light, but not in stereo vision cameras. A main limitation of stereo vision is the capturing of objects with untextured surfaces, since the correspondence problem is even harder to solve.

3D imaging methods have to deal with multiple errors and limitations. The following section aims to give a partial overview of ToF related challenges.

2.1.3 Challenges

ToF cameras have a restricted **distance range**, due to errors and noise. Multiple researchers are investigating errors, their reasons and possible remedies [Rapp, 2007], [Frank et al., 2009]. The range of ToF camera is variable and is limited by the illumination and the unambiguity range. Current PMD system support a range measurement up to 50 m [PMDTechnologies, 2012].

One main limitation are known **errors**, either systematic or stochastic, which are handled in Section 2.2.

The **resolution** is still quite low compared to stereoscopic systems. Actual cameras provide a maximum resolution of 288 x 352 pixels [Infineon, 2013]. This resolution is sufficient for present applications, but still leaves room for improvement. The low resolution is caused by the complex correlation pixels, which need a certain area.

Another limitation of current ToF cameras are **motion** effects. These are results of the multiple correlations which are not performed at the same time. This error has high impact, since many applications will use ToF to capture motion of the target (e.g. gesture). Motion can refer to moving objects and a fix-placed camera or the capturing of immobile objects while the camera is moving.

The field of applications for 3D imaging methods and especially ToF is manifold and an extract of its applications are shown in the next section.

2.1.4 Applications

The possibilities for ToF applications are miscellaneous:

- Medical researchers are focusing on ToF and its applicability for medical applications, especially patient positioning and respiration monitoring [Schaller, 2011].
- Various car manufacturers support the research in automotive applications, e.g. automatic braking, pedestrian recognition, convenience services inside the car [Hsu et al., 2006].
- Another interesting application is Human-Computer-Interaction with ToF camera assistance to interact for example only with gestures with the electronic device [Van den Bergh and Van Gool, 2011].
- Researchers also study the possibilities of ToF supported robot control in labyrinths to gain important depth data for navigation [Fuchs, 2012].

Over the last years, multiple sources of depth errors in ToF have been identified. An extract of known errors is illustrated in the next section.

2.2 Systematic Errors and Countermeasures

Errors of the ToF procedure have been subject to intensely research to identify countermeasures [Rapp, 2007], [Fuchs and Hirzinger, 2008]. The common procedure is to perform complex calibration for each sensor and for each setting. The next sections aim to give a basic understanding of the main errors.

2.2.1 Circular Distance Error

The circular distance error, also known as wiggling error, is a known systematic sinusoidal depth distortion. A graphical representation can be seen in Figure 2.6. The error varies with the distance between scene and sensor and the common approach is to calibrate it [Foix et al., 2011]. Recent studies have focused on methods to reduce this error, besides the costly calibration. The state-of-the-art depth calculation is to use 4-phase capturing, although detailed examination has shown that algorithms with an odd number of considered phase shifts introduce less distance related error. One approach is to capture only three phase images with an equidistant phase shift of 120 degrees, eg. 0°, 120° and 240°. The magnitude of the wiggling error can be reduced by a factor of seven. Another topic of research is to vary the phase shift minimal by each setup.

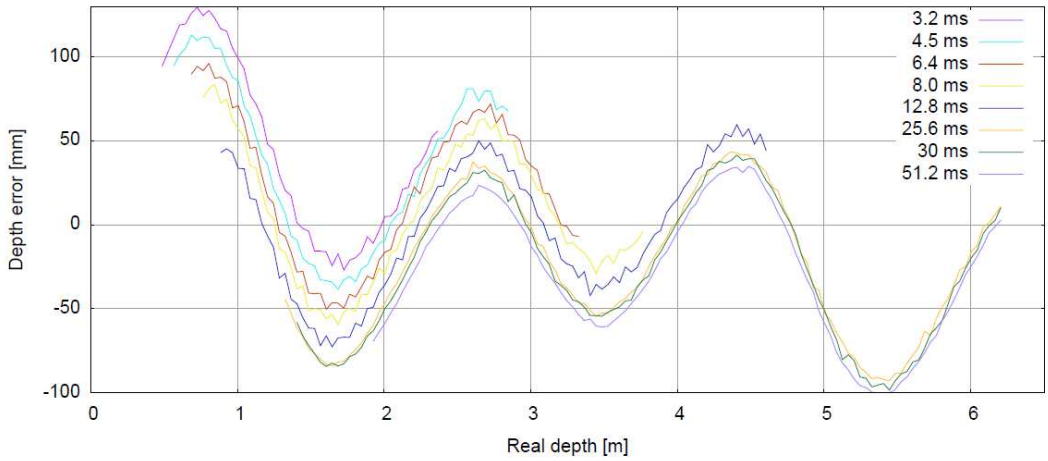
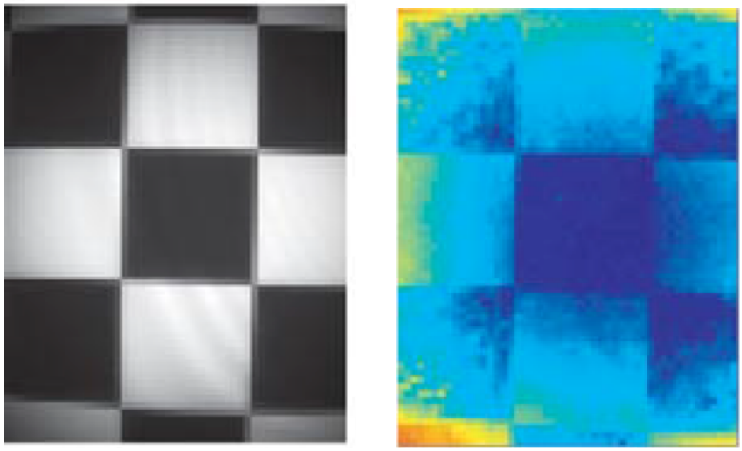


Figure 2.6: Wiggling error [Rapp, 2007].

This approach is called Adaptive-Super-Resolution (ASR) and promises a significantly higher SNR [Grünwald, 2013].

2.2.2 Amplitude Related Error

The Amplitude Related Error is caused by intensity differences. Dark surfaces appear to have a different distance than light surfaces. Especially in outdoor scenes with moving objects the impact of this error increases. Figure 2.7(a) shows an equidistant plane with a checkerboard pattern. In Figure 2.7(b) the depth image can bee seen. Although the distance to the whole plane is identical, the pattern can be observed in the depth image. This error also occurs for surfaces with differing reflectivity. Non-linearities in the sensor are most probably causing this error.



(a) Greyscale image of the plane. (b) Distance image of the plane.

Figure 2.7: Amplitude-Related-Error demonstrated on a checkerboard pattern [May et al., 2009].

2.2.3 Pixel Dependent Errors

In contrast to the above mentioned errors, pixel dependent errors do not occur for the whole image in the same manner, but differ from pixel to pixel. Two known errors are discussed in the following sections.

Fixed Pattern Noise

A non-temporal and spatial noise exists in 3D cameras. This noise also exists in 2D cameras and is individual per pixel. This noise is called Fixed Pattern Noise (FPN). When examining all pixels together, the offset appears like a pattern. The FPN consists of a coherent and a non-coherent part.

The non-coherent FPN is the characteristic of each pixel and the result of dark current due to leakage current at the photo diodes, varying material properties and differences in fabrication processes. This noise increases with the illumination time. One possibility to reduce this noise is to use low-leakage diodes and perform dark-frame subtraction. A dark frame is gathered at deactivated illumination and minimal integration time and is later subtracted from every raw image.

FPN is also linked to row and column artifacts due to discrepancies in the signal path. The pixel array is divided into multiple identical blocks, which are controlled and read concurrently. Due to slight differences in the controlling components, block-wise offsets occur. Since the pixels are connected serially, they behave like RC circuits and cause phase offsets and minor reference signal levels. This can be observed in continuous offsets in the rows and columns of each block.

Non-coherent FPN is considered a random error, whereas coherent FPN is considered a systematic error. The labeling of these latency related effects as "noise" in the classical understanding is badly elected, but due to its fixed occurrence, this nomenclature has been established.

The coherent FPN is a result of the position of a pixel on the chip and the entailing discrepancy in the signal path length. Coherent FPN is also known as Fixed Pattern Phase Noise (FPPN) [Albrecht, 2007].

FPN in general is frequency independent and calibratable. During the last years, research regarding different calibration algorithms has been done [May et al., 2009], [Drayton, 2013], [Fuchs, 2012], and [Haker, 2010].

Fixed Pattern Phase Noise

FPPN is not based on the pixel itself, but its position on the chip due to the differing signal path to each pixel. It cannot be eliminated through dark-frame subtraction, since the effect is overlapped by systematic errors. Amplitude related offsets due to different intensities, materials and surface characteristics can add an erroneous ratio to the true distance. Another side effect is that it causes edges in the depth images, which leads to segmentation faults. The state-of-the-art approach is to estimate the FPPN with the aid of a plane and approximate the error with a polynom of third degree. Other calibration methods can be found in [Karel and Pfeifer, 2009], and [Fuchs and Hirzinger, 2008].

Supposing that the matrix is control by a signal path, the FPPN can be described with the a linear function $E_c(v) = b_0 + v \cdot b_1 + u \cdot b_2$. The parameter set b_0, b_1, b_2 has to be determined for every pixel block. (u, v) are the row and column number in the block. Therefore, the number of FPPN parameters is $3 \times$ number of blocks.

2.3 State-of-the-Art

In the following sections, projects with similar background and focus of research are shown. First of all, an arc tangent realization is illustrated explicitly. In the next section an FPGA implementation of a ToF system is explained in detail. At last a hardware calibration for a 2D camera is shown, which shows several calibration approaches, which might also be useful in 3D imaging.

2.3.1 Coordinate Rotation Digital Computer

In Section 2.1.2 the basic principle of ToF was shown. In order to calculate the depth out of the phase images, an arc tangent is applied. One possible algorithm to calculate the arc tangent is the COordinate Rotation DIgital Computer (CORDIC) procedure. CORDIC algorithms are linear convergence methods and are well suited to calculate trigonometric functions, logarithm, and square root. The algorithm was invented in 1959 by Volder [Volder, 1959] and is mainly used in real-time digital computers. Its main advantage is its low consumption of resources, since only shifts, additions, comparisons and LUTs are needed. Over the years, many researchers have evolved multiple architectures and implementations of the algorithm. Andraka [Andraka, 1998] implemented an FPGA specific solution, while Vladimirova and Tiggeler [Vladimirova and Tiggeler, 1998] compared iterative and cascaded solutions. Giannakopoulou [Giannakopoulou and Masselos, 2012] performed a hardware analysis of a parametric CORDIC IP.

Principle

CORDIC is an iterative fixed-point technique which converges bit-wise towards the solution. In Figure 2.8 a graphical demonstration is shown. Starting with an angle of 45 degrees, the present angle approaches towards the goal angle. Each phase rotation is half of the preceding phase rotation. The difference between real phase Θ and sub-total of the iterative phases are controlling the sign of the consecutive phase rotation. The resulting phase Θ is a sum of all positive and negative angles.

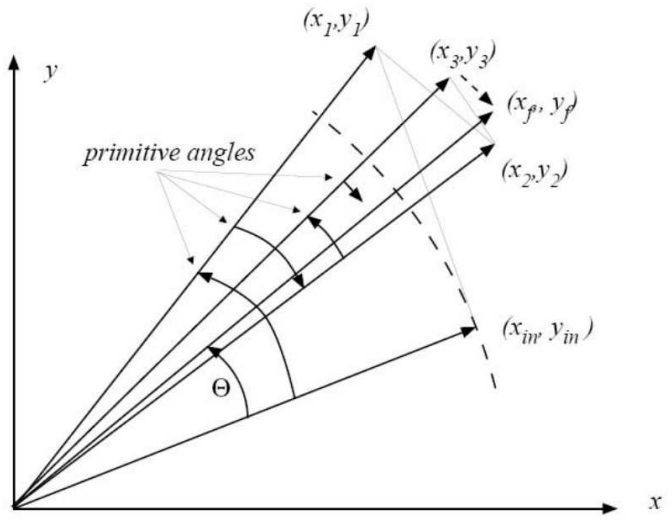


Figure 2.8: Graphical demonstration of the CORDIC algorithm [Ercegovac and Lang, 2004].

The mathematical rule for a counter clockwise phase rotation of Θ degrees is:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} cos\Theta & -sin\Theta \\ sin\Theta & cos\Theta \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2.5}$$

$[x_n, y_n]^T$ represents the resulting vector, $[x_0, y_0]^T$ is the starting vector and Θ the rotation angle. To perform a clockwise rotation, it is only necessary to change the sign of Θ. Multiplications are costly, therefore the mathematical equation $cos\Theta = \frac{1}{1+tan^2\Theta}$ is used and employed in Equation 2.5. The resulting equation is shown below:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{\sqrt{1+tan^2\Theta}} \begin{bmatrix} 1 & -tan\Theta \\ tan\Theta & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2.6}$$

Since the rotation angles are identical for every calculation, the pre-calculation of the tangent values and storage in a look-up table (LUT) is the next logical improvement. Volders key pro was the consideration of digital computers, which are storing information in multiples of 2. When examining the tangent values of the angles, his idea was to use only angles, whose tangent value was a divisor of 2. With this improvement no extra storage is needed, but only shifting of the vector's x and y coordinate.

$$\begin{aligned} x_{i+1} &= x_i - y_i\sigma_i 2^{-i} \\ y_{i+1} &= y_i + x_i\sigma_i 2^{-i} \\ z_{i+1} &= z_i - \sigma_i \cdot \arctan 2^{-i} \end{aligned} \tag{2.7}$$

In Equation 2.7 the simplified iterative steps are shown. [x_i, y_i] is the vector; σ represents the direction of the phase rotation (counter clockwise vs. clockwise); the imaginary value y_i is used to determine the direction of the next phase rotation, see Equation 2.8.

$$\sigma_i = \begin{cases} -1, & \text{if } y_i > 0 \\ +1, & \text{if } y_i \leq 0 \end{cases} \quad \text{and i = 0,1,2,..., n-1} \tag{2.8}$$

The resulting phase is the sum of all positive and negative performed phase rotations, see Equation 2.9.

$$\Theta = \sum_{i=0}^{n-1} \Theta_i \sigma_i, \quad \sigma_i \in -1, 1 \tag{2.9}$$

A realization of the working principle can be seen in Figure 2.9: the diagram shows the structure of the iterative CORDIC. As can be observed, only multiplexer, shifter, adder and a small LUT are needed to perform the calculation. The decision function d_i is driven by the sign of the imaginary part of the coordinate y_i. In the beginning, the initial coordinates $[x_0, y_0]$ are loaded into the register. Depending on the decision function, the coordinates are rotated clockwise or counter clockwise, which corresponds to addition or subtraction of the bit-shifted variable. The final calculation of the resulting phase is not shown in this block diagram. The higher the number of iterations, the more precise is the result.

The next section will present a ToF project which deals with the challenge of real-time 3D image acquisition.

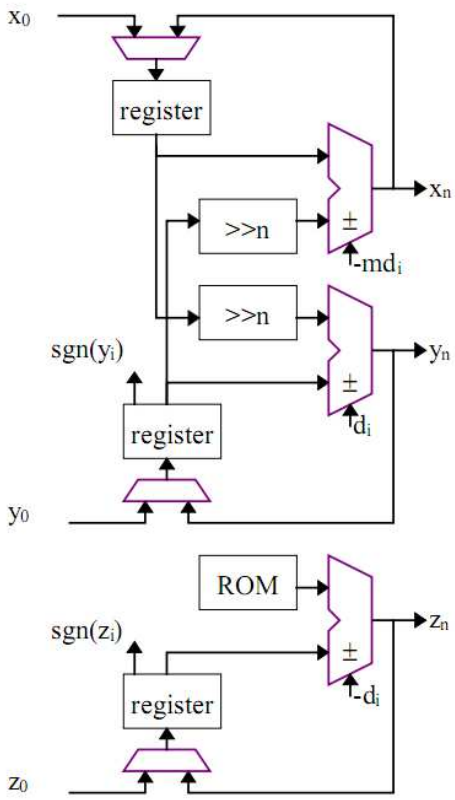


Figure 2.9: Iterative CORDIC structure [Andraka, 1998].

2.3.2 Heterodyne Range Imaging in Real-Time

3D imaging techniques have become an interesting and relevant topic of research over the last decade. Hand in hand with their evolution and enhancement arises the demand for real-time imaging. In this sub-chapter, a promising approach to deal with this challenge is presented. [Jongenelen et al., 2008] have been focusing on a ToF solution with help of an FPGA board.

Framework

The applied method is a state-of-the-art ToF procedure, which projects modulated light onto a scene, captures the reflected light, and mixes it with the modulation signal. The modulation frequency f_{mod} is known and variable. Laser diodes with a power of 80 mW are used to illuminate the scene in a 90 degree angle. The used camera is a Dalsa Pantera 1M60 digital video camera with a maximal frame rate of 220 Hz. The supported resolution is 128 $\times$ 128 pixel and the high frequent shuttering is done with a Photek MCP 125 image intensifier. A block diagram of the existing Ranger system hardware can be seen in Figure 2.10.

In order to calculate the distance value, the difference between reflected and emitted light has to be known accurately. This critical requirement is achieved with a Direct Digital Synthesizer (DDS) circuit board with high-precision analog devices with a maximal frequency of 400 MHz, driven by a 20 MHz temperature-compensated oscillator. One of the main advantages of the DDS board is that it only needs a single ended clock to create the control signals. Its task is to control the emitted light, the reference signal for the reflected signal as well as the FPGA. Although the analog ICs work up to a 400 MHz, the maximal frequency of the whole system is limited by a low pass at the output stage and therefore is 160 MHz.

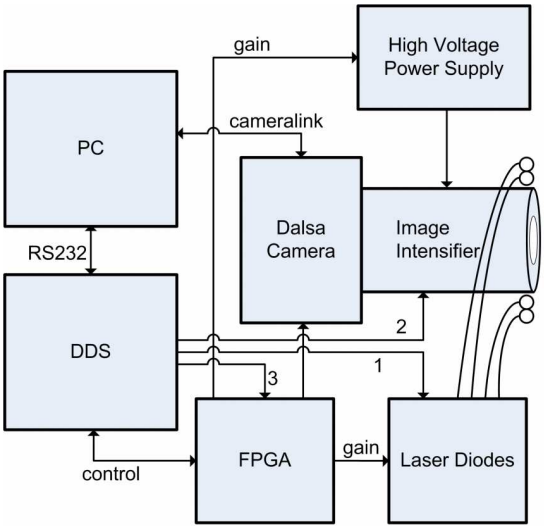


Figure 2.10: Existing ranger system hardware [Jongenelen et al., 2008].

The system is controlled by the Xilinx Spartan 2 FPGA, a micro controller and the DDS board. The frames are transmitted over a cameralink serial interface standard to the PC and the calculation is carried out offline. Five raw frames are used to calculate the depth image. The developers focus on a sliding window approach, which calculates for every received raw frame a new depth frame. This method increases the frame rate, but in order to notice a step change in the depth image, still five raw frames are needed.

Procedure

The proposed system is routing the raw data directly to the FPGA, instead of the PC. The pixel clock is 40 MHz per tap and the system consists of two taps to parallelize the read-out process. The depth calculation is operated on the FPGA and the depth images are not only sent over Ethernet to the PC, but also displayed on a VGA monitor. The developed system can be seen in Figure 2.11 and consists of an additional Altera Nios II Stratix II Development Kit with a Stratix II EP2S60 FPGA and a Nios CPU.

Software

The Hardware Description Language (HDL) synthesis was done with the Altera Quartus II 7.1 software and Nios II 7.1 IDE was used for programming the CPU. The programming and debugging of the FPGA has been performed with a Joint Test Action Group (JTAG) connection. The needed resources are listed in Table 2.2.

Results

The system has been evaluated with moving targets, to receive results for changes in distance. The evaluation has shown that the system works well and that the resulting error is around 4cm. The system is limited by the Ethernet interface to the PC, which only supports a frame rate up to 8 frames per second (fps). The full data rate of approximately 60 Hz can only be achieved when using the VGA monitor.

The next presented project deals with distortion correction in 2D images using FPGA based systems. It has been selected since many calibration methods apply similarily for 2D and 3D applications.

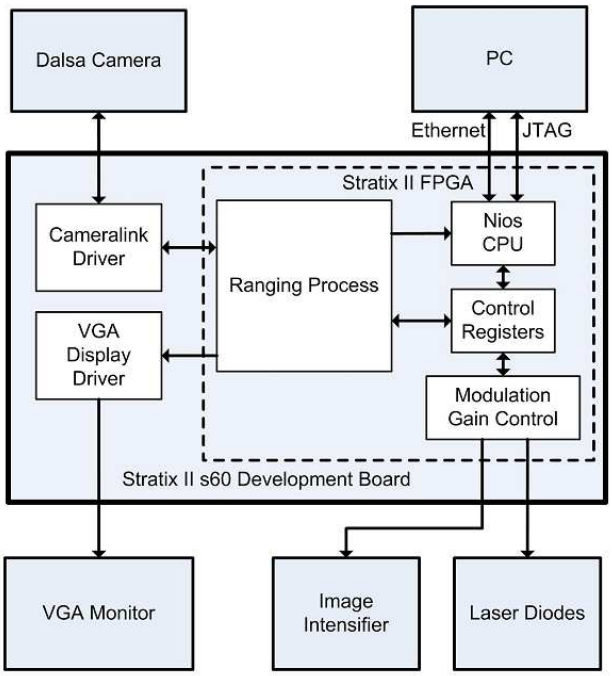


Figure 2.11: Altera Stratix II Development Kit [Jongenelen et al., 2008].

Resource	ALMs	DSP9x9	BlockRAM (bits)
Ranger Process	1529	8	1,163,476
Nios CPU	2285	8	48,128
Other	417	0	181,248
Total	4231	16	1,392,852
Total as % of available	9 %	6 %	55 %

Table 2.2: Stratix II EP2S60 FPGA Resource Usage [Jongenelen et al., 2008].

2.3.3 Real-Time Image Distortion Correction using FPGA based system

This project deals with distortion correction for geometric distortions in 2D images. In order to achieve a real-time capable system, the post-processing is done on an FPGA.

The first important step is to analyze the distortion and extract the calibration parameters of the used camera and its sensor. The subsequent correction of the disturbance is performed with aid of these parameters. Each pixel of the original image is weighted and influences multiple pixels in the output image. The position of the pixel in the original image is back-calculated. Since this back-calculation will not lead to a integer position, but to a floating number, the luminance is influenced by the surrounding pixels. The weight of each pixel is determined by bi-linear interpolation, compare Figure 2.12. In the upper right part of the figure the output image and in the upper left part the original image is shown. The luminance of one output

pixel is determined through its neighboring pixel in the original image, in this case through the weighting of its four neighbor pixels. The design of the system is generic and configurable for different foci and zooms. As mentioned, the decision to perform the calculations on an FPGA is justified in order to provide an undistorted real-time video stream.

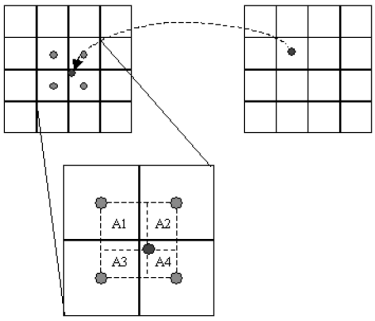


Figure 2.12: Luminance bi-linear interpolation [Hernandez et al., 2008].

The main focus of this work is to deal with radial and tangential distortions, which occur due to alignment problems between optical axis and image plane. The necessary calibration can be done as soon as the camera parameters are known. In literature there can be found multiple calibration algorithms with different advantages and disadvantages [Bailey, 2002]. In the proposed solution a pin-hole model diagram has been used. A pinhole camera uses a light-proof box without a lens, but only a small pinhole to capture an image. This procedure only captures the direct light between an object and the capturing point. The image is captured upside down on light sensitive material on the back side of the light-proof box.

The goal of the presented system is to back-calculate from an orbital image, as it is taken by the camera, into a rectangular image and remove the effect of the lens. For wide-angle cameras with a short focal distance a polynom of degree five is necessary to determine the position of one original pixel in the output image. Depending on the lens, a different number of parameters is recommended. Generally speaking, the higher the number of calibration parameters, the more accurate is the correction. Needless to say, the more parameters are used, the more complex and costly is the processing. Hence a trade-off between accuracy and complexity has to be found.

The formulated non-linear system of equations is solved with a minimization technique, in this very solution with a Levenberg-Marquadt technique. This numeric optimization method is a least square algorithm named after Kenneth Levenberg and Donald Marquardt. The luminance of the original point cannot be transferred directly to the output pixel, since it also affects the surrounding pixels. The weighting of the luminance for the surrounding pixels is determined with a bi-linear interpolation.

Architecture

The block diagram of the proposed architecture can be found in Figure 2.13.

The activity is serial and starts with the reception of the original image. In the first process step, the image is parallelized by buffering. Secondly, the pixel positions of the original image are mapped to the output image. With this knowledge the authors proceeded to elicit the influence of the original pixel to certain output pixels. The weighting and the mapping coordinates have been calculated during the foregoing calibration process and are stored in a LUT. During the actual transformation these parameters are solely applied.

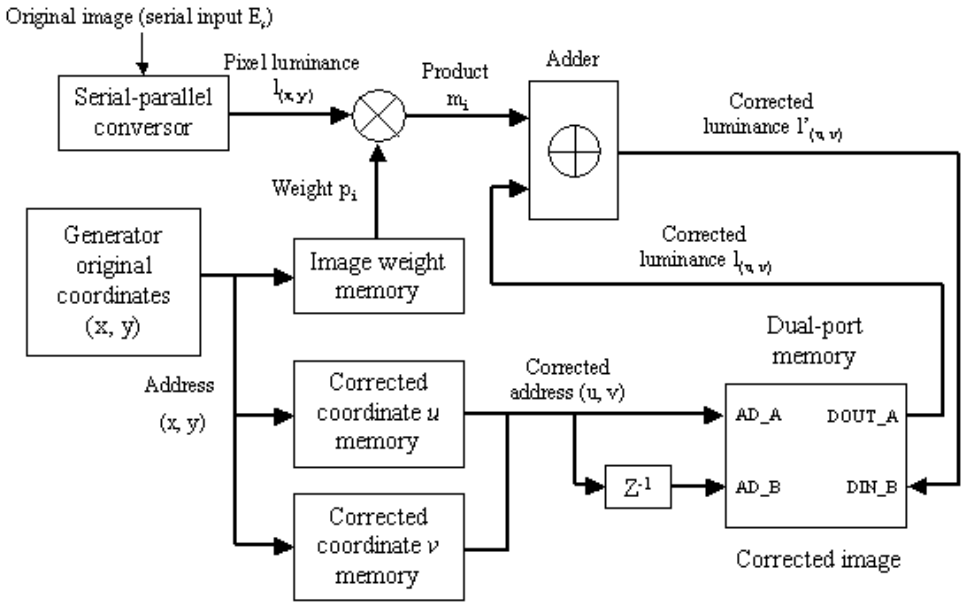


Figure 2.13: Block diagram of the distortion correction system [Hernandez et al., 2008].

The calculation rule is

$$l'_{(u_i,v_i)} = P_i \cdot l_{(x_i,y_i)} + l_{(u_i,v_i)} \tag{2.10}$$

in which $l_{(u_i,v_i)}$ is the affected luminance in the output image; $l'_{(u_i,v_i)}$ is the resulting luminance; $l'_{(x_i,y_i)}$ if the luminance in the original pixel and p_i is the corresponding weight.

The original and the resulting output image can be seen in Figure 2.14. It becomes obvious that a correction of the geometrical distortion has been performed. The difference is most evident at the tower buildings in the left part of the image. A frame rate of 50 Hz could be achieved and the design is flexible in terms of different focus and zoom settings.

(a) Before correction.

(b) After correction.

Figure 2.14: Recorded image before and after the performed distortion correction [Hernandez et al., 2008].

Applications and necessitated distance

Although the ToF principle can be applied for multiple applications, this thesis focuses primarily on short distance human computer interfaces with aid of 3D imaging. In Figure 2.15 an abstract of short distance applications and the resolution needed to distinguish certain changes are shown. A relatively small resolution with $160 \cdot 120$ pixels is sufficient to recognize gestures for interaction with a mobile device and a laptop. This interaction is expected to be done with finger motion and simple token gestures. Samsung [Oh et al., 2013] has presented its new Dynamic Vision Sensor (DVS) at the end of 2013 and sets a future market trend with this. The proposed sensor is a promising solution with gesture recognition rates above 95% for basic movements as "left", "right", "up", and "down".

Desktop PCs and All-in-One PCs on the other hand require a higher resolution since the observed distance is up to 1.5m.

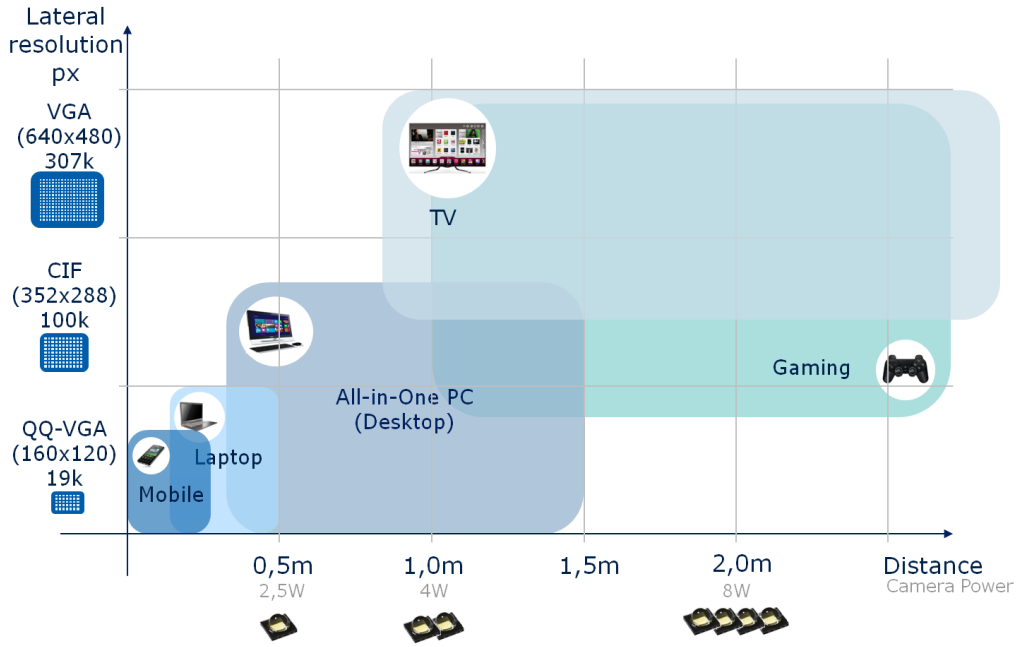


Figure 2.15: ToF user applications.

The greatest distance in this field of application is required for gaming and TV control. In this applications no finger movements can be distinguished, but body movements and hand gestures recognition are totally satisfying. Microsoft XBox Kinect [Microsoft, 2013] has revolutionized the gaming market in 2010 when introducing a motion detection capable game console. The first depth sensor was a structured light solution developed by 3DV [3DV, 2013] and Primesense [PrimeSense, 2013], although in the new Kinect a ToF sensor is used [Microsoft, 2013].

The following chapter lists system prerequisites and illustrates the design of the new system regarding its static and dynamic characteristics.

— CHAPTER THREE —

Design

This chapter focuses on the design prerequisite of the proposed system. Section 3.1 describes the system, which is currently in use and the limitations of it. The outcome of this are the system objectives. In order to clarify the problem, the system environment is sketched. Besides the static and dynamic design also non-functional requirements and the aspired quality of service are discussed.

3.1 Current System

In this section the current system, its components and the interfaces are shown. The limitations of the current system are explained and the design objectives derived.

The current system is shown in Figure 3.1 and consists of three main units:

- the sensing unit,
- the illumination unit, and
- the peripheral controller interface.

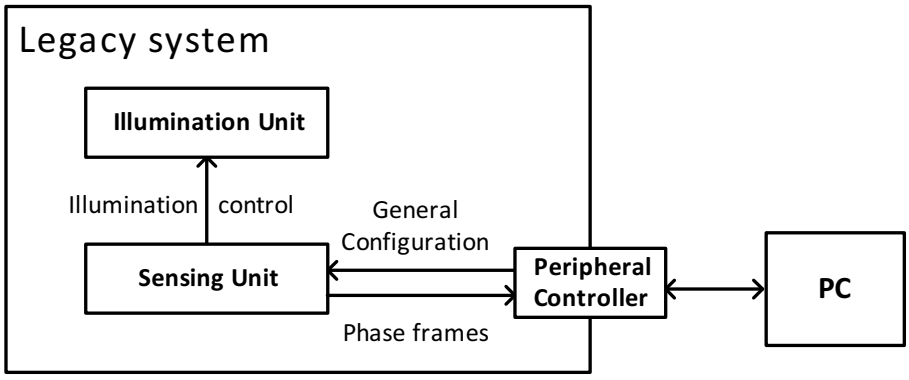


Figure 3.1: Logical representation of the current system.

The starting point of the new design is the current system, which is delivering raw data to the PC. The communication between the complete system and the PC is done via a Universal Serial Bus (USB) and a peripheral controller interface. This interface translates the configuration data from the PC to Inter Integrated Circuit (I^2C) commands and transmits these commands to the sensing unit. The sensing unit receives the commands and executes the necessary configuration steps. After the configuration process and receiving the start signal, the illumination unit transmit IR light and captures the reflected signal. The correlated signal is then transmitted to the PC.

The sensing unit, as it is referred to in this work, is an application-specific integrated circuit (ASIC) with a broad field of activities: It is receiving the configuration data from the external interface over an I²C interface. Once triggered from an external controller (PC) the sensing unit performs all necessary start-up routines according to the previously programed configuration. Furthermore, the sensing unit controls the illumination unit and performs the necessary phase shifts to capture and correlate all four phase-shifted images.

The main task is the actual sensing of the reflected light and its integration with the reference signal. The resulting analog signal is converted into a digital signal with 12-bit resolution and transmitted to the peripheral controller. The 12-bit data can be transmitted via a serial interface or a parallel interface. A detailed block diagram can be found in Figure 3.2.

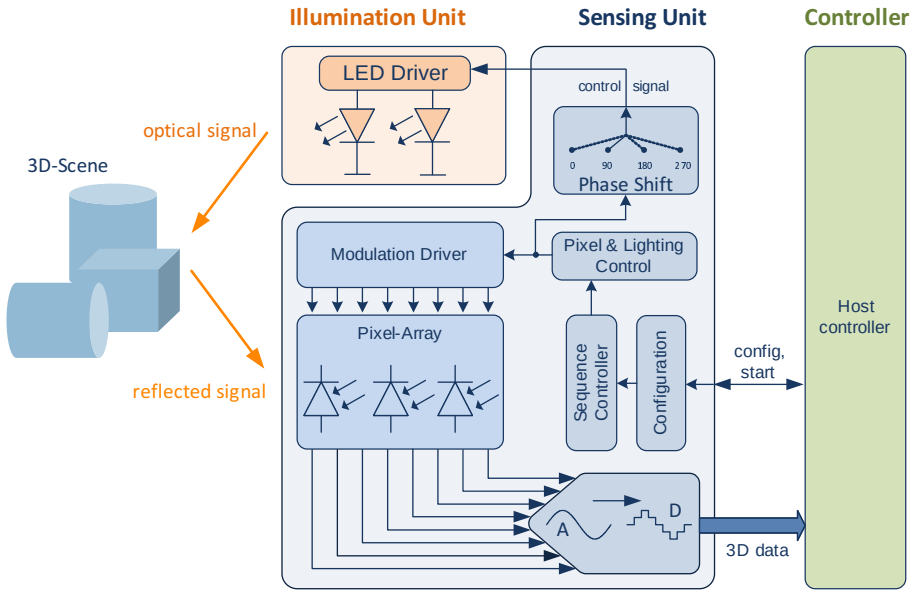


Figure 3.2: Block diagram of the sensing unit.

The illumination unit is an external light-emitting component, which just receives On-/Off-signals from the sensing unit. The illumination unit is selected regarding to the use case (necessary frequency, accuracy). For testing purposes a basic circuit with one LED is chosen.

The peripheral controller which offers an super-speed USB interface on the PC side and a configurable multi-purpose in interface on the peripheral side. This multi-purpose interface is configured to provide a I²C command interface and a 12-bit parallel data interface. The USB interface was chosen, since it is one of the most common interface standards. The peripheral controller collects the data from the sensing unit and delivers it as video stream to the PC. The configuration is done via an I²C bus and is stored in the internal memory of the sensing unit.

3.1.1 Limitations

One limitation of this design is the illumination unit: its accuracy and maximum frequency are limiting the illumination frequency. The current default illumination frequency is set to 20 MHz for general application. Depending on the luminance intensity an appropriate illumination time has to be selected. The illumination time is directly influencing the maximum number of fps: a longer illumination time leads to a lower frame rate. The maximum number of fps of the described system is around 100 [Infineon, 2013].

In the Equations 3.1 and 3.2 the relationship between the frame rate, the pixel array consisting of 352 × 288 pixel, an ADC resolution of 12 bit and the data rates of the serial and the parallel

interface is shown. The illumination time, which is also a big factor in the overall time from starting the measurement until receiving the frame data, is not considered in this equation. The frame rates for the serial and parallel interfaces are gained from the data sheet.

$$f_{pix\_parallel} = \frac{f_{pif}}{A_{array}} = \frac{36\ MHz}{352 \times 288} \approx 355.1\ fps \tag{3.1}$$

$$f_{pix\_serial} = \frac{f_{csi/2}}{A_{array}} \cdot \frac{1}{res_{ACD}} = \frac{1.6\ GHz}{352 \times 288} \cdot \frac{1}{12\ bit} \approx 1315.2\ fps \tag{3.2}$$

The aspired data rate is around 1.6 Gbps, which can only be achieved with the serial interface. The parallel interface allows a maximum data rate of 580 Mbps.

Another limiting factor is the transmission to the PC: although USB 3.0 has a maximal transmission rate of 5 Gbps a processing at maximum rate is not guaranteed. Since the selected operating system and the USB interface are not real-time capable and the task management is done automatically, the result is frame dropping at the maximum frame rate. Since all four phase images are needed for the correct calculation of depth, the loss of one-out-of-four frames will lead to the loss of one z-frame.

3.1.2 Objectives

The objective of the new system is to enhance the overall system performance.

The aim is on the one hand to provide an all-in-one system, which delivers not only the raw data (in this case the four phase images, one for each phase shift) but the already computed depth value (z-frame). This step will lighten the load of the requesting device. Especially in devices with less available processing power, it is important to reduce the necessary computations. The pixel array has the dimensions 288 times 352 and the depth calculation has to be done for every individual pixel. This will lead at least to 304,000 calculation steps, presuming that the arc tangent calculation is done within one process step per pixel, not considering the read and write overhead. Depending on the clock frequency, an iterative software-calculation on the device will not only lead to a significant stress of the CPU, but also result in an essential delay.

Moreover, the early depth calculation can significantly decrease the data rate to the computer by a constant factor of 4. This is due to the fact that not all four phase frames, but only the z-frame has to be transmitted. This data reduction will relieve the interface between system and computer. The goal is to simultaneously omit an input buffer overflow on the phase frame acquisition and boost the z-frames through-put.

The depth calculation shall be performed with two different algorithms:

- the state of the art 4-phase algorithm with equidistant phase shifts of 90 degrees, respectively $\pi/4$ radiant, and
- the recently formulated 3-phase algorithm with phase shifts of 120 degrees, respectively $2\pi/3$ radiant [Grünwald, 2013].

Subsequent to the hardware implementation of the depth computation algorithms, these two algorithms shall be compared regarding their errors. The expectations regarding literature [Grünwald, 2013], are significant improvements of the wiggling error for the 3-phase algorithm. Another expectation is the diminishing of the FPPN and to facilitate its prediction.

The design requirements can be achieved with different system architectures. An excerpt of possible architectures is given in the following section.

3.2 System Architectures for Imaging Applications

In this subsection multiple architectures for imaging applications are presented. Starting from a basic architecture with a sensing and computation unit until a completely integrated solution.

3.2.1 Basic Architecture

In Figure 3.2.1 the basic system design can be seen. The raw data is transmitted directly to the computation unit, where the further processing is done. The processing consists of noise rejection, elimination of known errors, conversion from radial to orthogonal distance and, depending on the imaging technique, specific process steps. In a structured light solution, the reflected pattern has to be mapped to the emitted pattern and calculated back to the scene. In this technique the two captured images have to be merged to receive the depth image. Since the focus of this thesis is on the ToF procedure, further discussion will concentrate on ToF-specific process steps.

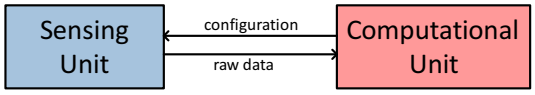


Figure 3.3: Basic architecture with a sensing and a computation unit.

As noted above, ToF needs multiple phase images to calculate the depth information of a scene. This means, that the four phase frames are captured on the sensing device, processed to the computation unit and the arctangent conjunction is done on the computation unit. The main advantage of this approach is the comfort of implementation: only the sensing unit is needed in hardware and the post-processing can be implemented in software, which is swifter than any hardware implementation. The disadvantage in this design is the remaining high working load on the PC. Many researchers have investigated this topic during the last decade [Jongenelen et al., 2008].

The next presented architectures aims to relieve work-load from the computational unit by installing a pre-processing unit.

3.2.2 Advanced Architecture with a Pre-Processor

In order to achieve higher data rates and relieve the computation unit, parts of the post processing can be outsourced to a pre-processor, for example an FPGA. The design principle is shown in Figure 3.4. The pre-processor takes a big part of the work load from the computation unit and relieves it thereby.

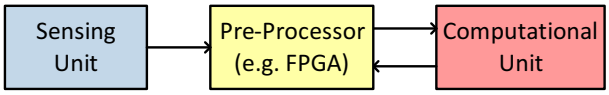


Figure 3.4: Advanced architecture: the distance calculation is performed on a pre-processor and the resulting distance data is transferred to the computation unit.

The current market trend is towards mobile systems for imaging applications and human computer interaction. Therefore, the next presented architectures is composed for mobile devices.

3.2.3 Architecture for Mobile Devices

The next intermediate step is to remove the connection to high-power computational units and move towards embedded devices with only limited CPU instead. One scenario can be seen in Figure 3.5: The sensing unit is sending raw data to an FPGA, which is doing advanced post processing and carries out the whole necessary calibration, noise and error correction. The CPU only receives corrected depth images, which can be used directly in its applications.

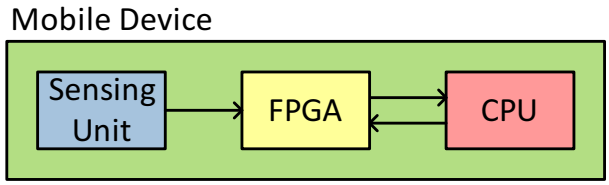


Figure 3.5: Architecture for mobile devices.

Especially in mobile systems, size and cost of all system components play a significant role. The next architecture is a solution which integrates sensing and distance calculation onto a single chip.

3.2.4 Integrated Architecture for Mobile Devices

The last scenario shows an architecture, which could be used in mobile devices with limited size and resources, see Figure 3.6. Due to the high production volume and extreme size limitations in smart phones and tablet computers, only a highly integrated ASIC solution is conceivable. The power of the device is limited and especially computation time and resources on the CPU are costly. The whole necessary calculation, calibration and processing has therefore to be done on-chip. This trend will require a large amount of integration and research regarding known errors. Companies like Samsung [Samsung, 2013] and Pantech [Pantech, 2013] are market drivers in this field and already equipped their smart phones with 3D gesture recognition systems.

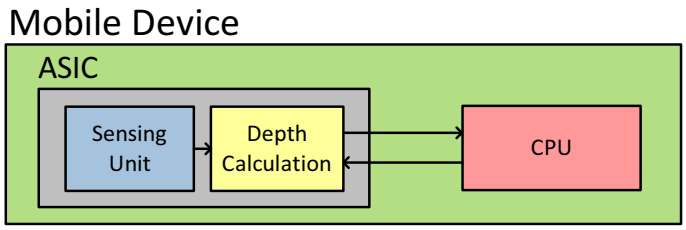


Figure 3.6: Integrated architecture for mobile devices.

The next section presents the basic architecture, which will be used to provide an adequate solution for the mentioned limitations.

3.3 Architecture

This section deals with the system environment and the static and dynamic design of the proposed system. A schematic representation of the new system is shown in Figure 3.7. In the following sections the processing unit is examined in detail.

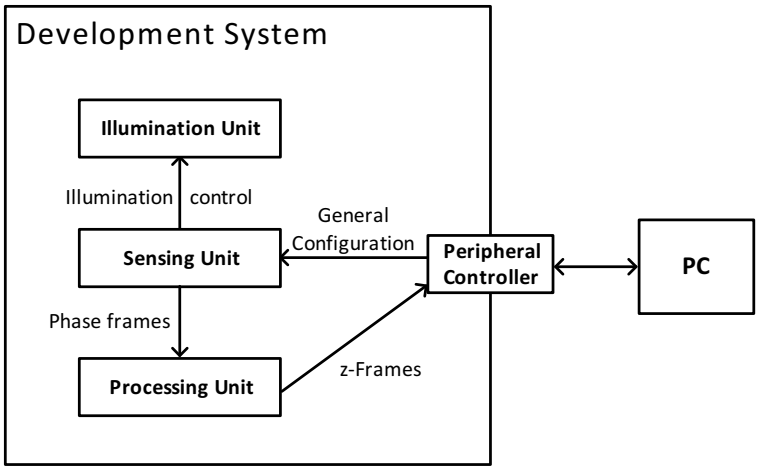


Figure 3.7: Logical representation of the new system.

A processing unit is placed between the sensing unit and the peripheral controller. The processing unit receives the phase frames, performs the calculations and finally transmits the z-frames.

In order to calculate the depth values, the data of all four phase images is needed. In Equation 3.3 the amount of memory needed to store all four frames is shown. Whatever detailed architecture will be implemented, the system has to deal with the calculated amount of data to achieve the calculation result.

$$\begin{aligned} memory_{needed} &= A_{array} \cdot (width_{data} + width_{meta\_info}) \cdot \#frames \\ &= 352 \times 288 \cdot (12\ bit + 4\ bit) \cdot 4 \approx 811\ kB \end{aligned} \tag{3.3}$$

The main interfaces of the processing unit are to the sensing unit, to the peripheral controller and to the external memory. In order to provide a detailed concept of the design, the system is discussed from different points of view. The main goal is to draw a holistic model of the overall system: on the one hand the static structure of the system and on the other hand the dynamic behavior. Moreover, the analysis is divided into different levels of detail.

3.3.1 System Environment

The overall system is shown in Figure 3.7. Hereby not only the processing unit, but also its interfaces and surrounding components can be seen. The legacy system, which is described in Chapter 3.1, has been extended with the processing unit. The communication between the triggering PC and the sensing unit is unchanged: the peripheral controller transforms the configuration commands and forwards it to the sensing unit. The sensing unit controls the illumination unit and records the reflected signals, but instead of forwarding it directly to the PC, it transmits the phase frames to the intermediate processing unit. Furthermore, the processing unit has a data interface to the peripheral controller. The system start signal is triggered by an

external user over the PC and the PC is interpreting the received data as video stream. The external user does not notice the additional components.

The data rate is configurable and defined by illumination time, interface clock frequency and the number of frames. The sensing unit provides, beside the actual data, also the clock and control signals. Since the processing unit chops the transmission line between sensing unit and peripheral controller, it has to provide on the one hand the z-frames and on the other hand the corresponding clock and control signals.

The next section deals with the static characteristics of the designed system and itemizes its sub-components.

3.3.2 Static Design

This chapter deals with the static design of the system. The system receives data from the sensing unit (either over a serial or a parallel interface), calculates the depth values, and forwards the data to the peripheral controller. The data of the sensing unit is transmitted in bursts, each burst contains one frame of raw data. The data is attached row by row. In Figure 3.8 one frame is shown. For the state-of-the-art calculation of the depth values four frames are needed and each frame consists of 287×352 12-bit pixel values plus 1×352 12-bit pseudo data.

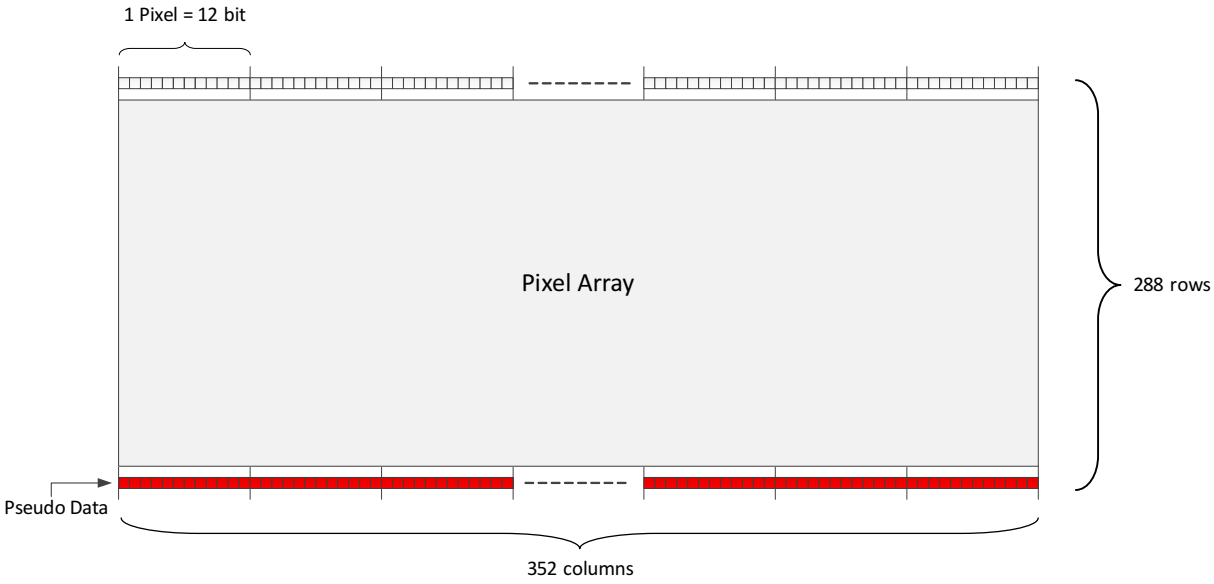


Figure 3.8: Representation of one phase image.

The system needs multiple sub-components to achieve the desired functionality. These sub-components are shown in Figure 3.9 and described below.

The purpose of the **Data Receiver** is to receive the data from the sensing unit. The communication interface is a 12-bit parallel interface, which transmits one pixel per clock cycle. The start and end of the frame is represented with a vertical sync signal, the start and end of a row is represented with a horizontal sync signal. The data integrity is assured on the falling edge of the clock signal.

The **Data Transmitter** transmits the depth data to the peripheral controller. The transmission is also done with a parallel interface, but in order to increase the calculation accuracy 16-bit are used to represent the pixel value. To fulfill the protocol, the system also has to generate the regarding horizontal and vertical sync signals, see Section 4.1.1.

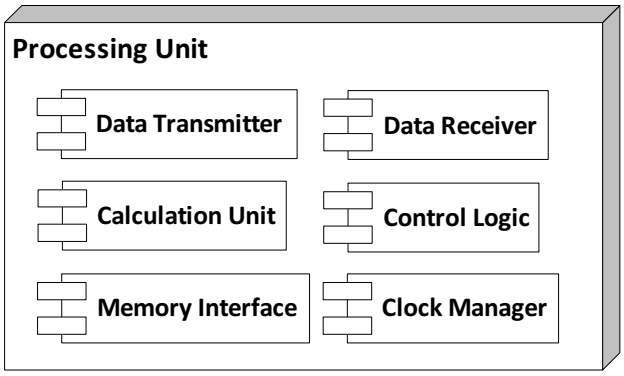


Figure 3.9: Physical view of system.

The **Calculation Unit** encapsulates the depth calculation, independent from the chosen algorithm and the type of arc tangent calculation. Details regarding the arc tangent algorithm will be discussed later.

The **Control Logic** is the core piece of the system, since it connects all sub-components and controls the data flow through the system.

The link to the memory is done with the aid of the **Memory Interface**. In order to ease the handling of the memory, it is mapping the user logic (read- and write-commands) to DDR3-conform commands.

The system consists of multiple clock domains: starting from the slow receiving clock, the fast calculation clock till the high-speed memory clock. To deal with all the different necessary frequency domains a clear clocking management is necessary. The **Clock Manager** is handling these different requirements and providing the essential clocks.

The detailed environment of the system and its interfaces can be seen in Figure 3.10: The processing unit has to provide an interface to write to the memory and read from it. The interface towards the sensing unit is only a receiver, while the interface towards the peripheral controller is a transmitter. The configuration is done over a separate connection.

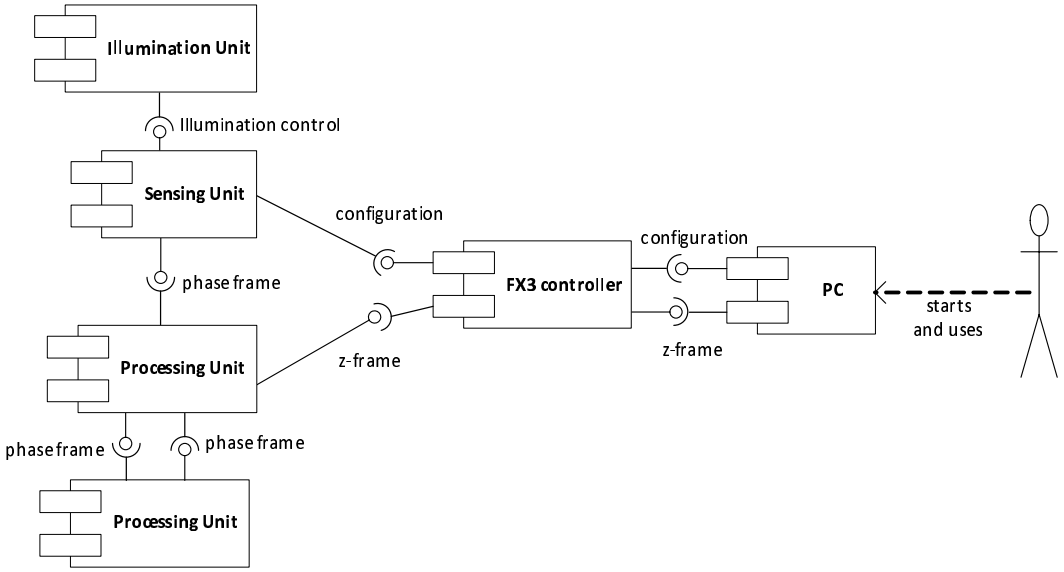


Figure 3.10: Development view of the system.

The following sub-section aims to familiarize the reader with the dynamic requirements and the elected behavioral design.

3.3.3 Dynamic Design

A sequence diagram of the overall system is shown in Figure 3.11. The four shown components are the requesting device, in general a PC, the sensor, the processing unit and an external memory.

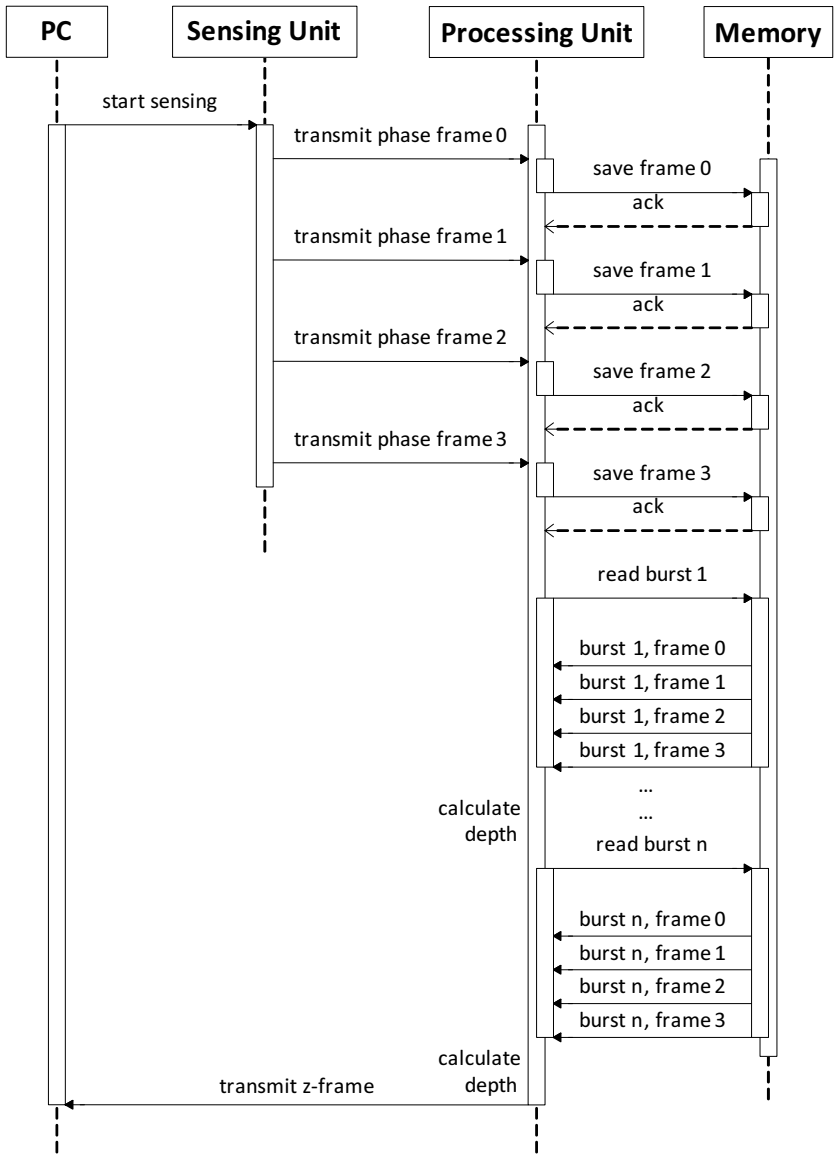


Figure 3.11: Sequence diagram of the system.

The starting of the system is triggered by the PC, which sends a start command to the sensing unit after defining all necessary properties. The sensing unit starts capturing the reflected light and transmits the raw data frame by frame to the processing unit. The processing unit receives the phase frame and stores it immediately into the external memory. This procedure is necessary, since the internal memory is not sufficient for the four raw data frames.

The saving of the raw frames has to finished before the next frame is received in order to guarantee that no data is lost. When the reception and storage of all phase frame is completed,

the actual calculation begins. For the depth calculation of one pixel, all phase frames are needed. Therefore, the raw data of each frame is read out of the memory again. For the purpose of speed-up and in order to avoid communication overhead multiple pixels are read out at the same time. These combined pixels are called a burst. As soon as all necessary raw pixel data is in the internal memory, the z-pixel calculation is performed. This procedure is repeated until the depth data of all pixels is computed. The depth pixel data, further referred as z-frame, is transmitted to the PC. This procedure is repeated while the sensing unit transmits phase frames to the processing unit, more precisely, the communication and calculation stops, when the user ends the connection. The reception, storage, calculation and transmission can be done simultaneously.

The following paragraphs show and explain the activity diagrams of the calculation component, see Figure 3.12 and Figure 3.13. The calculation unit is interchangeable to support multiple algorithms.

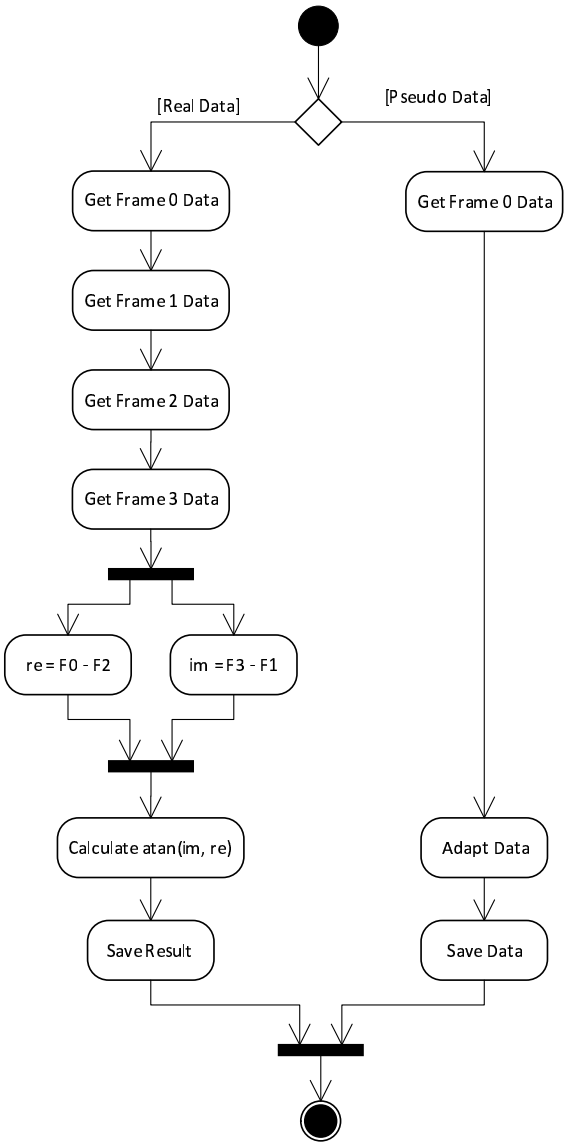


Figure 3.12: Simplified activity diagram of the 4-phase algorithm.

The considered algorithms are based on anterior research and are on the one hand a 4-phase algorithm and on the other hand a 3-phase algorithm to calculate the depth.

The combination instructions underlie the Equation 3.4 and 3.5.

$$z_{4-phase} = arctangent(F_3 - F_1, F_0 - F_2) \tag{3.4}$$

$$z_{3-phase} = arctangent(\sqrt{3} \cdot (F_2 - F_1), 2 \cdot F_0 - F_1 - F_2) \tag{3.5}$$

The activity diagram in Figure 3.12 illustrates the steps to get the depth data in case of a 4-phase algorithm. The first distinction is done between pseudo data and real data. The pseudo data is well-defined and created by the sensing unit. It is needed to gain the information regarding the phase number and other conditions. The pseudo data of the first frame is selected and most of it is just forwarded. Adaptation occurs only in information bits regarding the phase number, frame counter and depth calculation. In case of real data, all four phase values are collected and the imaginary and real input values of the arc tangent are computed. The result is in the range of $-\pi$ until $+\pi$ and is saved for further processing.

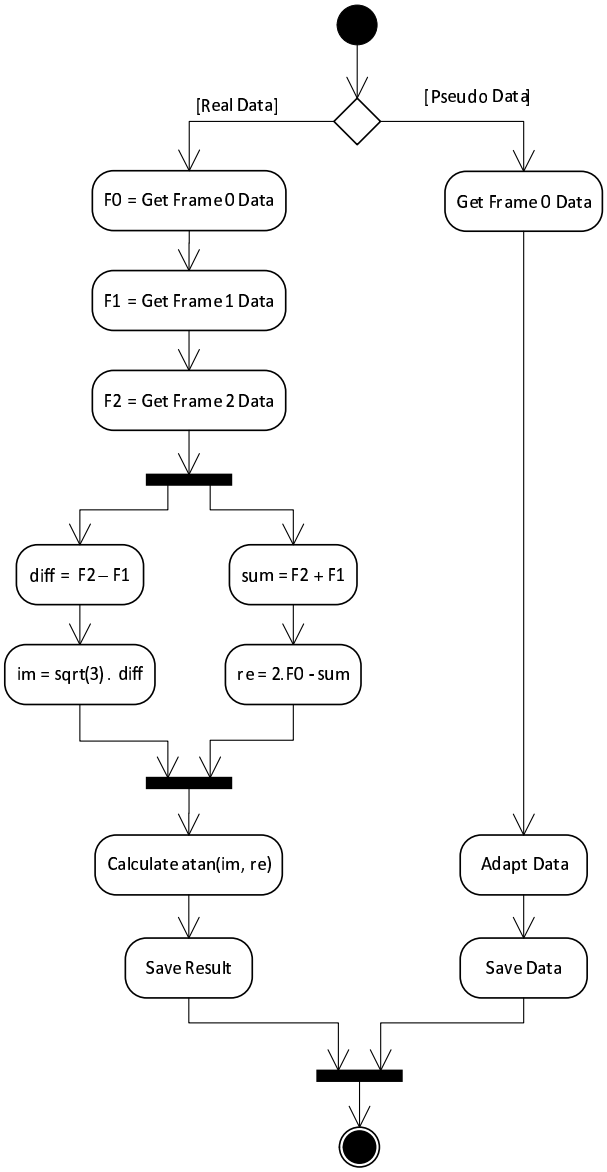


Figure 3.13: Simplified activity diagram of the 3-phase algorithm.

The 3-phase algorithm is shown in Figure 3.13. The sequence is similar to the 4-phase algorithm: the sole changes are in the pre-processing of the imaginary and real input values of the arc tangent. Since just three phase frames are needed, the fourth frame is omitted. The calculation instruction consists of one additions, two subtractions and two fixed-point multiplications. The arc tangent implementation and the post-processing stay untouched.

In order to prevent erroneous capturing and processing of data, meta information is stored in combination with every data word. The additional information can also be used to re-generate the synchronization signals for the output interface. Table 3.1 shows its possible states.

Meta Information	**Explanation**
FRAME_START	first data word of a frame
FRAME_END	last data word of a frame
LINE_START	first data word of a line, except the first row
LINE_END	last data word of a line, except the last row
TRANSMIT	arbitrary data word, which is not in the first or last position of a row
ERROR	mistakenly received data word

Table 3.1: Meta data.

Further requirements, which cannot be represented within the static and dynamic design are listed in the next section.

3.4 Further Requirements and Analysis

In this section non-functional requirements like availability, through-put and quality of service will be discussed. Later, a accuracy analysis is the system is performed.

3.4.1 Non-Functional Requirements

The calculation of the z-frames is performed on the FPGA. In order to receive the distance out of the z-values, the modulation frequency has to be considered by following the calculation steps in the Equations 2.1 and 2.2. This final step is performed on the PC with aid of a middle-ware.

The availability of the system is linked to the availability of the sensing unit, since the system is reacting on the raw data received from it. The system is responsible for the accurate depth calculation and thus has to react on every phase frame. No frame filtering is done within the system. To sum up, whilst depth sensing is done, the system has to be online and perform the computation steps.

The through-put of the system is defined by the maximal performance of the sensing unit and the input interface of the processing unit. Due to these limitations, the maximal through-put is around 1300 fps. To deal with this through-put a calculation clock frequency of minimal 140 MHz has to be provided.

The system shall be scalable to deal with several frame dimensions. The system reacts autonomously to the received dimensions, as long as its number of columns is a multiple of eight. This restriction is defined to ease the memory access, which is done in bursts. Variable turning-off of individual components is not intended, as the system needs all components for its proper

performing. Scalability in the terms of clocking frequency seems noteworthy, since the input clock frequency limits the overall system performance. At lower input clock a lower memory clock and processing clock could gain savings in terms of power.

In order to raise the degree of reuse, the design is divided into multiple blocks with clear interfaces. Further projects can for example easily reuse receiver or transmitter units or exchange and adopt the calculation steps. Since this project is part of a first research, further constitutive designs and implementations are likely. Thus, the interfaces have to be clear, comprehensible, and well-documented.

The communication interfaces are predetermined and their clocking frequencies are constrained by the system frequency of the sensing unit. The maximum clocking frequency for the parallel interface in this setting is 68 MHz. The CSI-2 protocol is implemented according to the MIPI CSI-2 specification [MIPI, 2013]. Its maximum clock rate is 800 MHz.

The system is fault-tolerant in terms of erroneous calculation, since only one individual pixel is influenced. The processing unit is able to detect missing pixels / interface errors and will abort the z-frame calculation. Furthermore, if the host controller expects a whole video frame and only one pixel is missing, the frame is dropped. Hence, the remaining procedure on the processing unit can be stopped as soon as one missing pixel is discovered. The system can start working again when it receives the next phase frame with a phase shift of 0 degrees.

The system consists of multiple sub-components which handle different types of tasks. Most of these tasks can be executed concurrently in order to build a pipeline. One limitation is the memory access, since only one read or a write access can be performed at the same time, see Figure 3.14. Concurrency is one main advantage of the implementation in hardware.

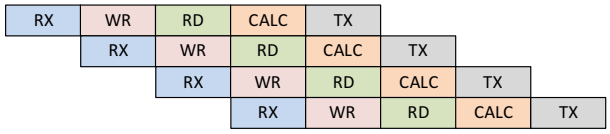


Figure 3.14: Principal architecture of a pipeline system.

The system itself is passive and only reacts on received data. An active design is also realizable, but not reasonable, since the final handling of the data is not performed on the processing device but on an external controller.

The next section deals with the question if a shut-down of particular system components is possible and reasonable.

3.4.2 Quality of Service

The design provides several possibilities to decrease the power consumption, which will be handled briefly. As known from literature, the dynamic power consumption is influenced by multiple parameters, see Equation 3.6. f is the clocking frequency, V_{dd} stands for the power supply, C_N for the output capacity of each node and β for the switching activity.

$$P_{dyn} = \frac{1}{2} \cdot \beta \cdot C_N \cdot {V_{dd}}^2 \cdot f \tag{3.6}$$

As mentioned above, after the user triggers the start signal, the sensing unit is starting-up according to its configuration and begins capturing. Therefore, a power-down condition of the processing unit is conceivable. During the power-down state the system will be supplied by a slow clock. The system has to react on the wake-up signal, which will be the start signal by the host controller. In order to guarantee the correct functionality, the wake-up has to be finished and the system has to be available when the first frame from the sensing unit is received. This optimization will influence the following parts: reception, transmission and storage of data will be turned off as well as the calculation. The remaining functionality is the reception and handling of the control signals.

The clocking frequency f directly influences the dynamic power consumption, hence another improvement can be to drop the clocking frequency. This implies that the final setting is known. The pursued approach is a general design, which aims to work for multiple settings and input frequencies. Therefore, this optimization has not been considered during development.

The mentioned calculation process was dealing with reference calculation methods with no accuracy limitations. The system will be implemented in hardware, which leads to limitations regarding its accury. An analysis of the loss of accuracy is shown in the next section.

3.4.3 Accuracy Analysis

The first step was to perform a comparison between the reference results and the results with the aspired calculation method. In this case, the computation results between floating-point and fixed-point implementation of the arc tangent are compared, see Figure 3.15.

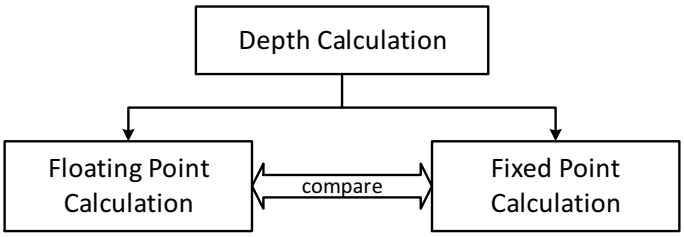
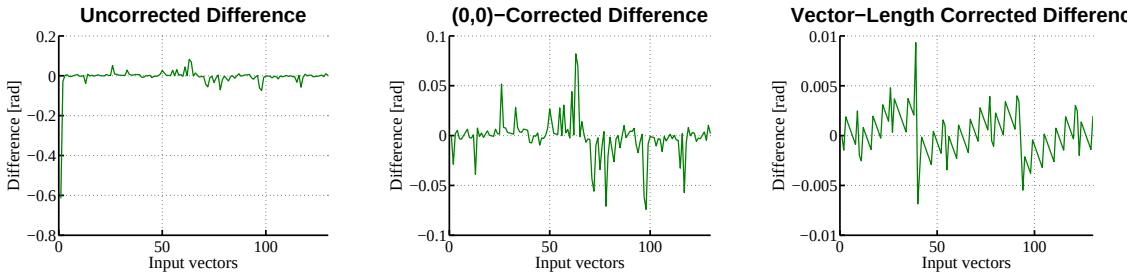


Figure 3.15: Verification of floating-point vs. fixed-point implementation.

This verification has been performed in Matlab, where the reference results were gained with the elementary trigonometric function $atan2(Y, X)$. The function accepts floating point values for real and imaginary part of the complex number. The function $atan2(Y, X)$ returns values in the range of $-\pi$ to $+\pi$ in contrast to the function $atan(Y/X)$, which returns values in the range of $-\pi/2$ to $+\pi/2$.

The fixed-point calculation has been performed with help of a software implementation of the serial cordic algorithm. The method uses the Matlab "Fixed-Point-Toolbox", where fixed point data types can be specified regarding their word length w and fraction length fl. In order to gain comparable results, the parameters have been set to match the precision of the sensor data. The input data-type shall refer to a value on the unit circle, therefore the parameters have been set as following: $w = 12\,\text{bit}$, $fl = 10\,\text{bit}$.

The calculation was divided into two steps: first the *atan* value was calculated with the explained bit-shift procedure, see Section 2.3.1. The input values of the arc tangent procedure are the absolute values of the imaginary and real part of the complex number. Due to the fact, that the input vector is in the first quadrant of the unit circle, the result is in the range of 0 to $\pi/2$ radiant. The second step is to transfer the gained result to the correct quadrant: hereby the afore neglected signs of the complex input numbers are used. The resulting range is thus $-\pi$ to $+\pi$, which results in the following fixed-point parameters $w_{out} = 12\,\text{bit}$, $fl_{out} = 9\,\text{bit}$.



(a) Uncorrected difference

(b) Difference after the removal of the not defined input vector (0,0)

(c) Difference for input vectors on the unit circle

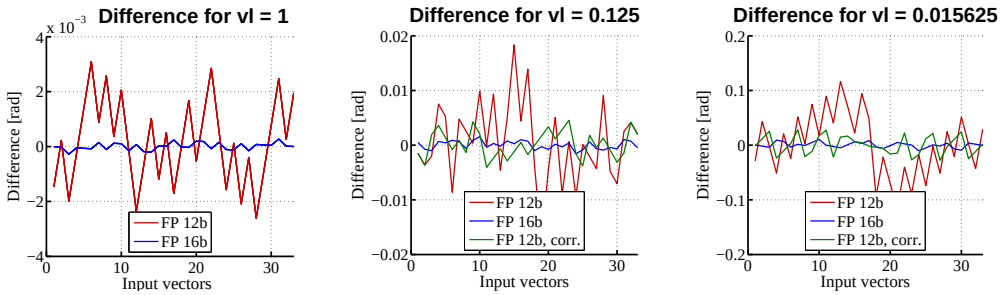
Figure 3.16: Comparison between floating point and fixed point arc tangent calculation.

The result of the difference between floating-point and fixed-point calculation is shown in Figure 3.16. The input values of the arc tangent calculation have been selected randomly. For each input pair (imaginary part, real part) a fixed-point and a floating-point calculation has been done. The figure shows the difference between the two calculation methods for a fixed-point width of 12 bit in radiant. The input values are covering the whole unambiguity range. The vector length of the input is defined as $vl = \sqrt{Re^2 + Im^2}$ and has been chosen randomly between 10% and 100% of the unit circle.

Figure 3.16(a) shows minor differences in most of the considered cases. The high differences around 0.6 radiant occur at the input values of $(0, 0)$. This is due to the fact, that the atan2 for the origin is not defined. Therefore, these discrepancies can be neglected. When removing these errors, the error plot shown in Figure 3.16(b) is received. The result shows still considerable differences between the two calculation methods. The reason for these peaks are precision errors: due to the fact, that the number of used bits and number of performed iterations is limited, the result cannot be approximated exactly for small input values. To reduce this error it has to be ensured, that the input values are approximately in a range of the unit circle.

Figure 3.16(c) shows the results after adjusting the vector size. It can be concluded, that the differences are in the range of maximal 0.01 radiant and in average around 0.00146 radiant. Another limitation is the quantization error, which increases with decreasing fixed-point length and is 1/2 least significant bit (LSB) of the output. Due to these minor discrepancies, the fixed-point implementation was considered reasonable.

In order to gain more insight about the errors due to the vector length of the input values, a comparison between a fixed-point calculation with $w = 12\,\text{bit}$, $w = 16\,\text{bit}$ and fixed-point calculation with $w = 12\,\text{bit}$ and vector stretching is shown in Figure 3.17. Figure 3.17(a) shows the calculation errors for $w = 16\,\text{bit}$ and $w = 12\,\text{bit}$ at a vector length of 1. It can be seen, that the error is smaller, the higher the precision of the fixed-point data type. Figure 3.17(b) shows the precision errors for a vector length $vl = 1/8$. The error is 5-6 times higher for both precisions. As noted before, the error is higher for the 12 bit calculation. The green line represents the



(a) l=1. (b) l=0.125. (c) l=0.015625.

Figure 3.17: Influence of the length of the input vector onto the fixed point result.

error for a 12 bit precision after stretching the vector length to approximately 1. Figure 3.17(c) supports the theory, that the stretching of a small input vector can significantly decrease the fixed-point error. Therefore, an vector stretching for small input vectors has been included into the hardware implementation.

The next chapter presents the implementation of the designed system in detail. Furthermore, the verification process is defined and the implementation results are shown.

— CHAPTER FOUR —

Implementation and Verification

In this chapter the implementation details of the system are shown. A feasibility study has been performed to obtain the best possible solution for the design. The details of the study can be found in Section 4.2. According to this study, the system was implemented with state-machines on an FPGA. Next, the necessary interfaces and their limitations are explained, since these have a strong impact on the system parameters. The following section shows the detailed architecture and implementation including some example code to illustrate the implementation. The development environment and used tools are mentioned afterwards. Finally, the verification process is described and the implementation results are listed.

4.1 Interfaces

The proposed system architecture needs interfaces to communicate with other relevant components. The two most important interfaces are explained in the following subsections.

4.1.1 Parallel Interface

The parallel interface (PIF) provides the opportunity to receive multiple bits in general. The input interface from the sensor to the processing device uses twelve data lines in parallel. The frame is transmitted row by row and pixel-wise. Besides the data lines two synchronization lines exist: a vertical synchronization line (vsync), which marks the start and stop of a frame; and a horizontal synchronization line (hsync), which marks the start and stop of a row. Per clock cycle, one pixel is transferred. The clock is generated from the transmitter side and is a single-ended clock with configurable clocking frequency. Besides sensor specific configuration, also the maximal clocking frequency is a limiting factor to the system through-put.

The protocol does not provide any hand-shake functionality or possibilities to request a new transmission. Figure 4.1 shows a PIF protocol draft.

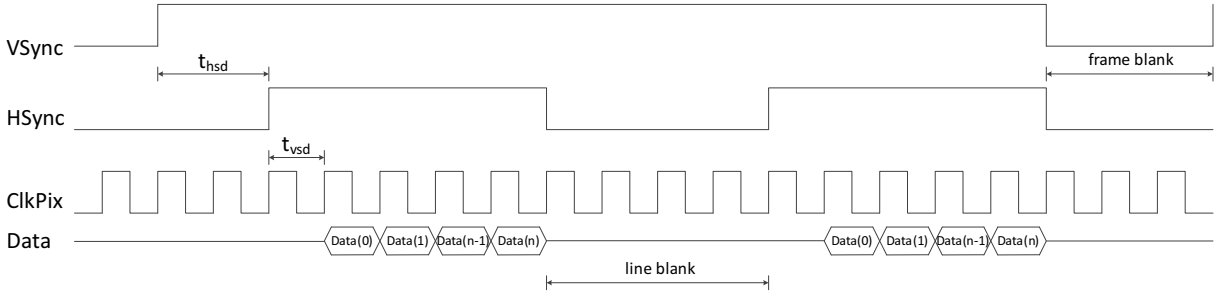


Figure 4.1: Timing of the parallel interface.

The pixel clock has to be started before the frame transmission is started. The number of parallel bits is extendable, since the peripheral controller permits a maximum data line number of 16. The active level of the sync signals can be modified, for further explanations an active-high signal is assumed.

So as to omit faults due to spikes in the sync signals and consecutively erroneous frame start or stop, only changes in the sync signals which occur on a rising clock edge are allowed. The data, in turn, is available at the falling clock edge. The time elapsed between two rows is called line blank and the elapsed time between two frames is called frame blank. Between vsync and hsync and hsync and data transmission a certain delay can be defined. This delay is used to offer the receiver the possibility to react adequately when noticing frame and line start. Per default, this delay is set to zero clock cycles.

Receiving Interface

Figure 4.2 shows the activity diagram of the parallel input interface. The receiving of the data is only performed, whilst horizontal (hsync) and vertical (vsync) signal are activated.

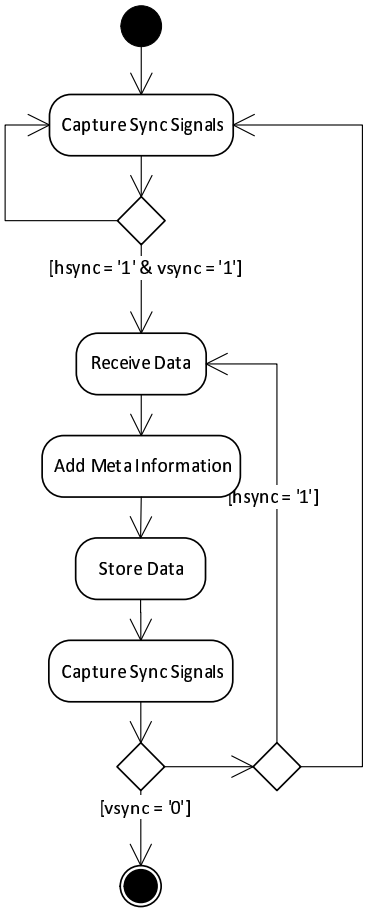


Figure 4.2: Simplified activity diagram of the data receiver.

The received 12 bit pixel data is expanded with 4 bit meta information, which holds information regarding the horizontal and vertical position of the pixel. The meta information states have been explained in section 3.3 and are binary encoded. The data is stored in the external memory. Continuously the synchronization signals are examined. When both sync signals are activated, the reception is performed. After each completed row, the hsync signal is deactivated

to indicate a line break. Once the vsync signal is also disabled, the whole frame has been received and the input activity is completed.

Transmitting Interface

The restoring of the synchronization signals necessary for the parallel interface is done in the output component. To provide a flexible solution, which is not relying on static definition of the dimensions, the meta information of each pixel value is used. The depth pixel value is taken and the meta information is filtered. In case of a frame start signal, both sync signals are enabled and data is transmitted. Further center pixel data is transmitted without changing any sync signal. In case of a line end, the hsync signal is disabled and the transmission of data is stopped until the next line start occurs. In case of a frame end, both sync signals are disabled, the data transmission is stopped and the state machine stops. See Figure 4.3 for a graphical representation.

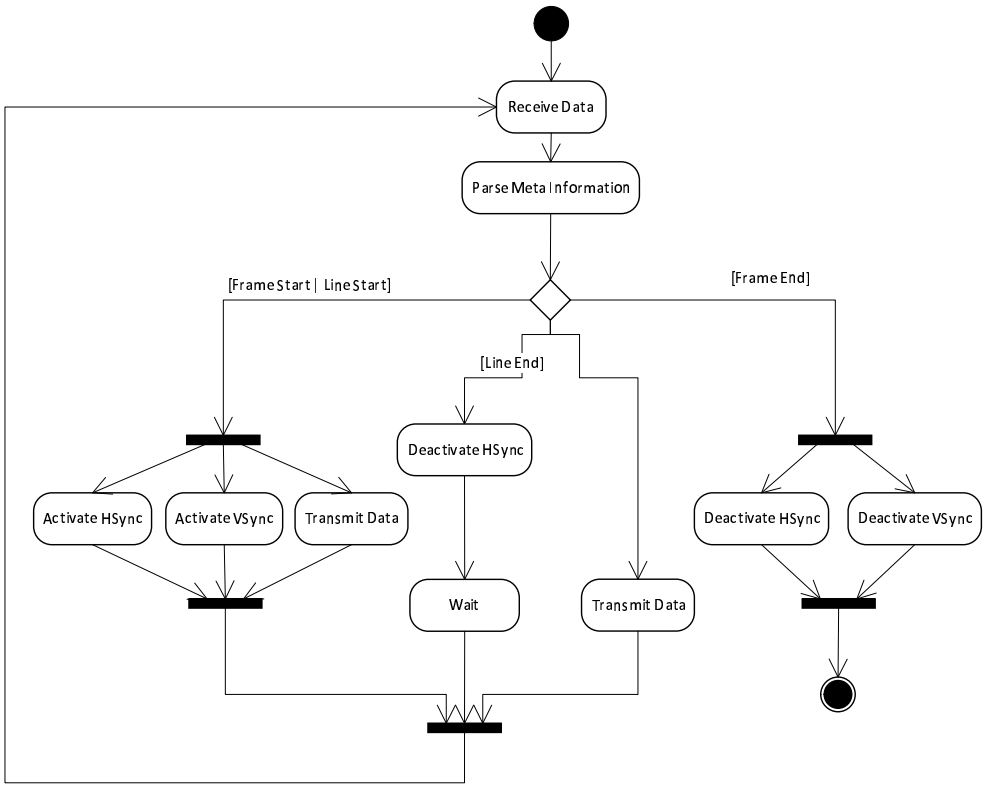


Figure 4.3: Simplified activity diagram of the data transmitter.

The next section will deal with details of the memory interface, which is an abstraction layer to ease the use of the external memory.

4.1.2 Memory Interface

The high amount of data requires an external memory and for this purpose a DDR3-RAM has been choosen. The maximal data rate of the memory is 6.4 Gbps for a 8 bit interface with a clock frequeny of 303 MHz and dual data rate.

In order to ease the use of the external memory, an intermediate layer between external memory and processing device is installed, see Figure 4.4. The aim of the layer is to decrease the complexity of the interface handling to the memory. Xilinx provides an IP core, called Memory

Interface Generator (MIG), for this purpose. The MIG takes care of all user-related signals and their logic and converts them to memory specific commands. The MIG is principally divided into a command and a data path, which can be configured regarding the specific requirement. The read, write, and command clock can be chosen independently, although it is suggested to use the same clock for every user interface port.

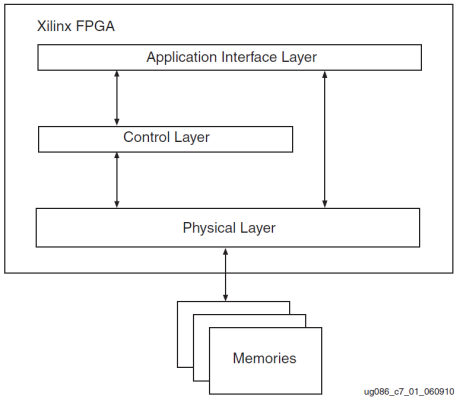


Figure 4.4: Memory Interface Representation [Xilinx, 2010a].

Each user port can be defined as

- an unidirectional read port,
- an unidirectional write port, or
- a bidirectional read and write port.

The whole communication is done with multiple command and hand-shake signals, which inform about defined conditions like buffer full or data available. The memory has to be initialized and calibrated before using it after a system start. The calibration is started automatically by the MIG as soon as the reset signal is released.

Each signal has to be changed at the rising edge of the particular user memory clock (either command, read, or write clock). The final commands to the memory itself are generated by the MIG, and the MIG also takes care of the external memory's timing specifications. The read or write access is started with the initiating of the regarding command, see Figure 4.5.

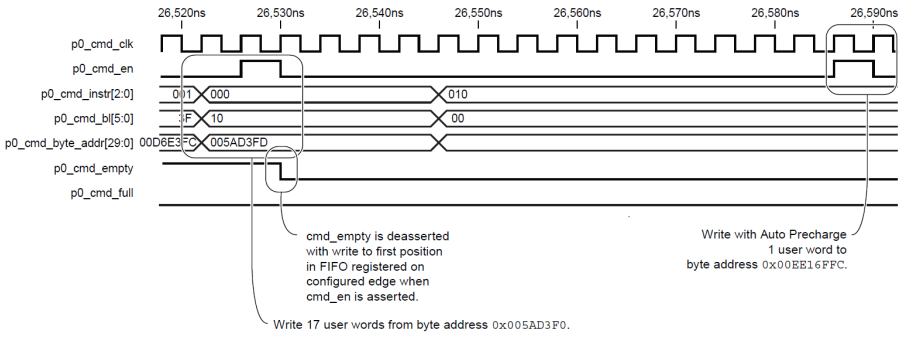


Figure 4.5: Memory Command Timing [Xilinx, 2010b].

The signal *cmd_en* indicates the initiating of a command, while in the *cmd_instr* bus the command is specified (read, write, blocking read, blocking write). The read or write address is as well transmitted as the burst length. As soon as the MIG notices the command, the *cmd_empty* signal is de-asserted.

Figure 4.6 demonstrates the timing of a read access to the memory. After the successful transmitting of the read command, the memory starts to return the ordered data. The signal *rd_empty* marks the availabilty of data at the read port. The *rd_count* bus shows the number of currently available data words. The activating of the *rd_en* signal starts the read process and due to the fall-through algorithm the first data is available, as soon as the enable signal is set. Until the *rd_en* signal is deasserted, one data word is available per clock cycle. As soon as all available data is read from the buffer, the *rd_empty* signal rises again.

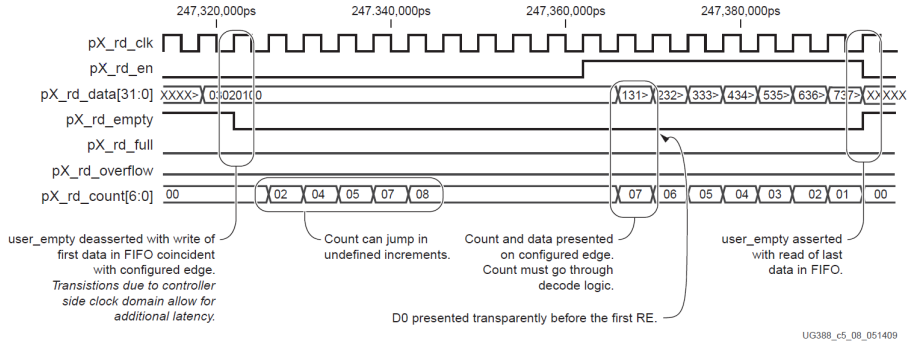


Figure 4.6: Memory Command Timing [Xilinx, 2010b].

The MIG needs a relatively large area, since it provides calibration logic and system logic for multiple settings. The parameter setting is explained in Section 4.2.3. In order to reduce the needed area, the unnecessary logic can be removed.

4.2 Feasibility Study

In order to find the best fitting architecture, configuration and parametrization of each submodules a feasibility study has been performed. The alternatives are discussed and compared regarding their impact onto the system.

4.2.1 Processing Unit

This chapter shows the advantages and disadvantages of each architecture and argues afterwards which solution fits best. Due to the fact that a FPGA is available, two of the treated designs of the processing unit will be using the FPGA. The other solution will be handled with an ASIC realization.

MicroBlaze on an FPGA

As mentioned in Chapter 3.3, one way of implementation is to use an FPGA. Due to its low cost, low power and availability, the widely-used Xilinx Spartan-6 FPGA has been chosen. Another advantage is the included design suite. The Xilinx ISE Design suite provides multiple free intellectual property (IP) cores, e.g. the free micro controller system MicroBlaze. MicroBlaze is a general micro controller system, which is based on the MicroBlaze processor. The micro controller provides the possibility to implement the necessary functionality in software instead of

hardware. This reduces the implementation time, since software development in general is less time-consuming than hardware development. The code can be reused easily if similar demands occur in the future. The cost of the MicroBlaze solution is low, since no extra components nor software (besides the necessary Xilinx ISE Design suite) nor fabrication is needed. In general, the MicroBlaze is an attractive alternative to implement the necessary tasks.

The amount of internal available memory is around 250 kB random access memory (RAM) blocks and far from the necessary amount calculated in Equation 3.3. Therefore, the system has to buffer the four frames in an external memory. To guarantee a real-time calculation the memory interface frequency has to be higher than the maximum input frequency per pixel, which is around 133 MHz for serial transmission. The memory used is a double data rate (DDR)-3-RAM, with a storage capacity of 1 Gb and a maximal frequency of 667 MHz.

The maximal clocking frequency of the MicroBlaze is 120 MHz. Due to clocking limitations and further overhead the MicroBlaze soft-core as an entire system cannot meet the design requirements.

State Machine on an FPGA

Another design possibility also includes the FPGA, but instead of the synthesized micro controller, a plain hardware solution is analyzed. The calculation will be realized in a hardware description language (HDL) with multiple logic blocks and state machines. This design provides a higher clock frequency (maximal 1 GHz) and therefore permits faster processing without limiting the through-put. The same DDR3-RAM as described in Chapter 4.2.1 is necessary to fulfill the memory requirements, but in this design the clock frequencies can be met.

Many parts can be parallelized and pipe-lined in hardware and thus speed-up the processing. The implementation time is, compared to the micro controller, higher since hardware development is more time-consuming than software programing. Compared to an ASIC, the implementation time is still quite acceptable. In order to get the most benefit, the system can and should be encapsulated to small and reusable parts. Furthermore, this design can use free IP blocks from Xilinx. These IP blocks are frequently used pieces of code, which ease the implementation, e.g. a serial and parallel implementation of the CORDIC algorithm.

ASIC

The last examined solution for the requirement is related to the state machine design: a fully customized ASIC, which performs the necessary calculations. The main advantage is the high accomplishable speed and its possible use in mass-production. The disadvantage of this solution are the immense amount of implementation time and high fixed costs, which are not justifiable for a prototype. The re-usability of an ASIC is also lot smaller than that of an FPGA, although in outlook to future mobile devices an integration of the depth calculation in the overall system would be an enormous step ahead.

In Table 4.1 a clearly arranged summary of the different architectures can be seen. Due to the mentioned advantages and disadvantages of the architectures, the state machine system architecture on an FPGA has been chosen; the exact static and dynamic design is expound in the following section.

	FPGA: MicroBlaze	FPGA: VHDL State Machines	ASIC
Implementation time	small	medium	high
Re-usability	medium	medium	low
Processing speed	slow	fast	fast
Cost	low	low	high

Table 4.1: Comparison of the possible architectures.

4.2.2 CORDIC

The main principle of the CORDIC algorithm has been explained in Chapter 2.3.1, now several implementation possibilities are explored regarding their feasibility. The first survey dealt with the question of using an IP core or implementing the algorithm from the scratch. Since [Giannakopoulou and Masselos, 2012] already analyzed the Xilinx CORDIC IP cores and came to the conclusion that it is a practicable possibility, this thesis focuses on a feasibility study regarding the parametrization of the IP core. The IP core provides a coarse rotation to extend the range from 90 degrees to 360 degrees. Depending on the requirements, the size of the data ports is configurable from 8 to 48 bits. The number of iterations can be configured as well, although the bus size of the data is limiting the preciseness, so a further increment of the iterations, but the bus size is ineffective. The availability of the result is marked with a valid flag. Per clock cycle one input value can be loaded. A comparison between the realizations can be found in Table 4.2.

	Serial Implementation	Parallel Implementation
LUT Flip-Flop Pairs	278	1005
Calculation Time (cycles)	12	1
Latency (cycles)	20	20
Error LSB	50 %	50 %

Table 4.2: An comparison between different CORDIC realizations.

The serial implementation is especially suitable for designs with hard size requirements. In contrast, the parallel implementation focuses on high through-put and therefore requires more on-chip size. Since a high through-put is required and the area needed is second-ranked.

The following section deals with the configuration of the MIG respective port configuration, bus width, and burst length.

4.2.3 MIG

The MIG provides various configuration properties. In order to find an adequate setting, various settings were compared, starting with the bus width.

Bus Width and Burst Length

Regarding the system requirements, the parallel interface will work with a maximum clocking frequency of 68 MHz and receive a 12 bit word in parallel. This frequency accords to a clock cycle of 14.71 ns. In the above section, the principle of the interface and the additional storage

of 4 bits meta information has been explained. The data amount, which has to be handled is shown in Equation 4.1.

$$DR = f_{rcv} \cdot d_{word} = 68MHz \cdot 16bit = 1.088Gbps \tag{4.1}$$

Table 4.3 shows a comparison of the available port width and burst length with the resulting number of pixels needed for one memory access and the resulting waiting time.

No.	Port Width	Data rate	Word per burst	Pixel per memory access	t_{burst}
1.	32 bit	19.39 Gbps	4	8	117.65 ns
2.	32 bit	19.39 Gbps	8	16	235.29 ns
3.	64 bit	38.78 Gbps	4	16	235.29 ns
4.			8	32	370.59 ns
5.	128 bit	77.57 Gbps	4	32	370.59 ns
6.			8	64	941.18 ns

Table 4.3: Port configurations and their resulting time.

The least number of pixels per memory access and the shortest waiting time until one access can be performed is with configuration no. 1.: a burst length of eight pixels leads to a waiting time of 117.65 ns, until all necessary data is collected and the memory can be accessed. The highest number of pixels per access and highest waiting time is at configuration no. 6.: 64 pixels are needed for a burst, which results in a waiting time of 941.18 ns. The waiting time t_{burst} results from Equation 4.2, where w_{MIG} is the port width of the MIG, w_{pixel} is numer of bits per pixel, l_{burst} is the number of words per burst and t_{pixel} the time needed to receive one pixel.

$$t_{burst} = \frac{w_{mig}}{w_{pixel}} \cdot \frac{l_{burst}}{t_{pixel}} \tag{4.2}$$

The number of pixels per memory access defines the size of the needed buffer to buffer the received pixels with the slow external clock for further processing with the fast internal clock. The higher the number of pixels, the bigger is the needed first-in, first-out (FIFO) buffer. In order to reduce the size of the buffer, setting no. 1 has been selected for the implementation.

Port Direction

The next problem was to find the appropriate bus direction for the communication. The available settings are regarding the bus width, its direction and the number of available ports. The settings are shown in Table 4.4.

No.	Bus Width	Direction	Number of ports
1.	128 bit	bidirectional	1
2.	64 bit	bidirectional	2
3.	32 bit	bidirectional	4
4.	32 bit	unidirectional	4+2

Table 4.4: Possible port configurations for the MIG.

Since the anterior investigation resulted in a port width of 32 bits, the direction of the port interfaces has to be defined. The only interaction between internal system and external memory

happens during the write access of the receiver and the read access of the calculation block. The command interface yet is shared between the read and write ports. A control logic to prevent simultaneous writing to the command port is required. The unidirectional port can only perform one operation: either write or read. The advantage of two uni-directional ports is that no attention has to be paid to the command interfaces. As enough ports are available, one read and one write port were selected. With this setting, read and write commands are completely seperated.

The design and implementation of the system was done in dependence on a development life cycle, which is explained in the next section.

4.3 Design and Implementation Process

The whole project was done with a development life cycle, see Figure 4.7. Starting from the rough idea, a literature research has been done to get an insight into the topic and to familiarize oneself with the state-of-the-art technology. Next the system was analyzed and the dynamic and static requirements were defined. After defining the specification, the new system has generally been designed. The design does not include any implementation specific information. Subsequently, a feasibility study of the system has been conducted. In this study multiple implementation possibilities for different blocks and interfaces have been studied and evaluated: the design options regarding the implementation platform (FPGA, MC, ...) were exploited and the IP cores for the memory block and the arctangent calculation were evaluated.

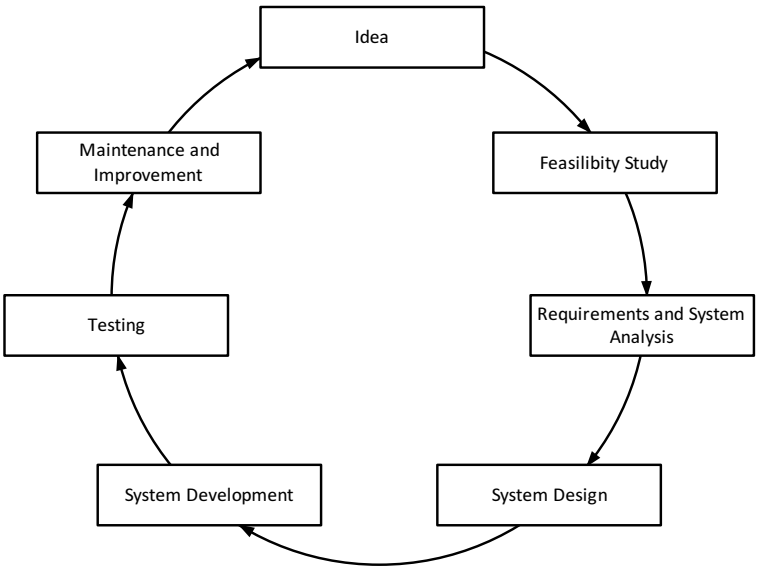


Figure 4.7: Hardware development life cycle.

This evaluation yielded the final implementation decision, which has been realized afterwards. The implementation was first done with artificial data and led over to real data. The whole system was extensively tested and verified, compare Chapter 4.6. The tests were not only done during the development, but also after the development to verify the design requirements. The detailed procedure can be found in Section 4.5. After the verification, multiple measurements have been performed and interpreted. Although the system fulfills all requirements, it still provides improvement possibilities, which can be achieved in further projects.

Afterwards the system was implemented regarding the results of the feasibility study. Implementation details can be found in the following paragraphs.

4.4 Implementation Details

In Section 3.3 the different designs have been compared and the decision regarding the architecture has been made: this thesis focuses on a state machine solution on an FPGA. The main interfaces of the FPGA are to the sensing unit, the peripheral controller and to the external memory. The main focus in the detailed architecture is the speed, since the necessary data rate has to be achieved. Furthermore, an efficient implementation in terms of power is aspired. In this section, the selected system is explained in detail.

The implemented system can be seen in Figure 4.8. As in the old system, sensing of the image is done in the sensing unit which consists of 288 times 352 individual pixels. The illumination of the scene is done with aid of the illumination unit, which is controlled by the sensor. The configuration and start signal is transmitted from the PC over the peripheral controller to the sensor. The captured and correlated phase frames are transferred to an installed FPGA. The whole calculation process is done on the FPGA and the intermediate data storage is done on a connected DDR3-RAM. The calculated z-frame with the distance information is transferred from the FPGA to the peripheral controller and the PC.

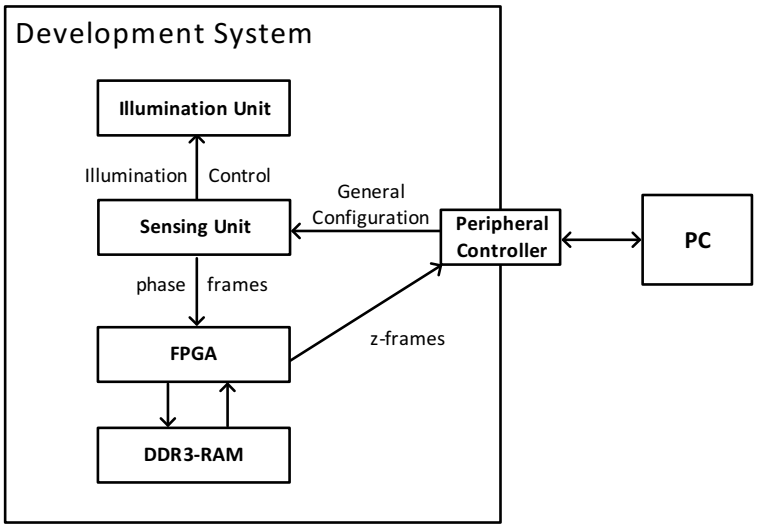


Figure 4.8: Overall system and its interfaces.

The FPGA and its components will be explained in the following paragraphs: the feasibility study suggested the use of an FPGA and a Xilinx Spartan-6 was selected to realize the necessary procedure. The configuration of the memory interface can be found below:

- port direction: uni-directional (one read port + one write port),
- bus width: 32 bit,
- burst length: 4 bursts per memory access, and
- selected bank: 3.

As interface between the sensor and the FPGA the parallel interface was selected: the receiver module is constucted to deal with an 12 bit interface. The synchronization settings have been limited: the interface does not support synchronization delay between the synchronization signals hsync and vsync and the data transmission, see Figure 4.1. The frame and line delay can be configured arbitrarily. The correct interpretation of the data signal is guaranteed with an edge synchronized gripping.

The calculation module has been divided into four sub-modules, which can be seen in Figure 4.9. The first module, called *Collect_Data* is receiving the raw data and buffering it for further proceeding. In the $Pre - Processing$ modul, the pixel-wise subtraction, multiplication and conjunction regarding the design steps in Figure 3.13 and 3.12 are performed. The $\sqrt{3}$ correction factor is approximated by a fixed-point representation.

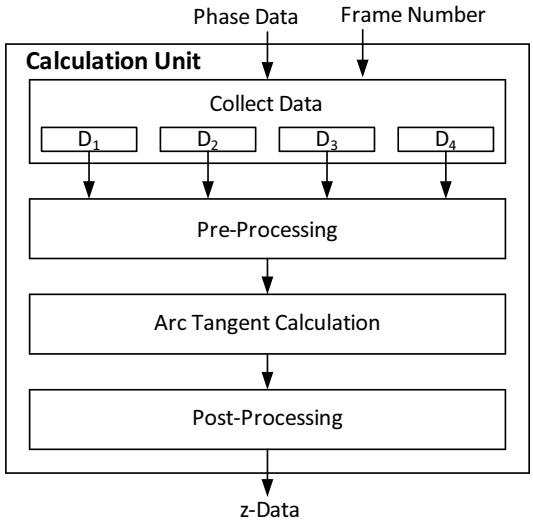


Figure 4.9: Detailed view of the calculation unit.

The arc tangent calculation, which is done with help of the Cordic IP core is done in the next sub-module. After the arc tangent calculation, the adaptation regarding the output interface settings is performed in sub-module $Post - Processing$. The Cordic core was configured as follows: the calculation is done in parallel, which leads to a speed increase at the expense of the needed area. The calculation depth was set to 16 iterations and the latency between feed-in of data and reception of a result is 20 cycles. Later one, one result per clock cycle is received.

The multiple clock domains of the system have been selected regarding the requirements and can be seen in Table 4.5. The input clock is controlled by the sensing unit. Its frequency can be configured and was set to 68 MHz. The output clock is generated onto the FPGA and necessary for the PIF to the PC. The clock is matched with the input clock. The reason for this is that especially during verification a forwarding of the phase data was necessary. This can only be achieved when receiving and transmitting happens at the same frequency. The memory clock was selected regarding its limits and set to 303 MHz. The user clock was set to 75 MHz since the limiting factor of the system performance is the received data. Any further speed-up of the user clock would lead to increased power consumption and no enhancement in the through-put.

Clock name	f_{clk}
Input clock	68 MHz
User clock	75 MHz
Memory clock	303 MHz
Output clock	68 MHz

Table 4.5: Clocking frequencies of the system.

The communication between the external memory and the FPGA has been abstracted with multiple layers.

- The MIG translates the user signals and commands into memory compatible commands.
- The writing of the memory is done with the sub-module $Memory\_Write$. The algorithm for a memory write is shown in Algorithm 1. The data, which is written into the memory, is stored into a buffer in the MIG. As long as the buffer is not full, the writing can be performed. The buffer is filled with data words, until the number of words per burst is reached. The writing process checks, if a write-command can be started and if it is possible, the command $'write'$ is written into the command buffer.
- The read out of the memory is done with the sub-module $Memory\_Read$. The algorithm works similar: A read-command for a specified address is put into the command pipeline. The MIG translates the command and receives the data from the memory. The data is stored in an internal buffer and as soon as a data word is availble, a notification signal is triggered. The read-out process of the buffer can be started then.

Algorithm 1 Write data to external memory

Summary: Receives data words and writes them at the specified address into the memory

```
 1: procedure writeToMemory(data, addr)
 2:
 3:     if writeBufferFull = false then
 4:         dataWriteEnable ←' 1'
 5:         for i ≤ numBurstItems do
 6:             writeBuffer ← data(i)
 7:         end for
 8:
 9:         while commandBufferFull = TRUE do
10:             wait
11:         end while
12:
13:         command ←' write'
14:         writeAddress ← addr
15:         commandEnable ←' 1'
16:         while commandDone = false do
17:             wait
18:         end while
19:     end if
20: end procedure
```

The data transmitter has to restore the correct synchronization signals. The easiest approach would have been to trigger the sync signals with help of a counter after one row respectively one frame. The disadvantage is the implicit knowledge of the frame size. The selected approach uses meta information, which is generated during the reception. This meta data is used in the transmitter to generate the adequate sync signals. The algorithm is shown in Algorithm 2. The code sniplet shows a simplified procedure. The transmission starts as soon as enough data has been calculated and is available to proceed.

Algorithm 2 Constructs the parallel interface signals

Summary: Decodes the internal data and generates the signals for a parallel communication

```
1: procedure constructPIF(data)
2:     while dataAvailable = false do
3:         wait
4:     end while
5:
6:     if (metaData = FRAME_START) then
7:         vsync ←′ 1′
8:         hsync ←′ 1′
9:         dataOutput ← data(1)
10:        i ← 2
11:        while (dataAvailable = true & metaData ≠ FRAME_END) do
12:            if metaData = LINE_END then
13:                hsync ←′ 0′
14:                wait numFrameBlankCycles
15:                hsync ←′ 1′
16:            end if
17:            dataOutput ← data(i)
18:            i ← i + 1
19:        end while
20:        vsync ←′ 0′
21:        hsync ←′ 0′
22:    end if
23: end procedure
```

The next section lists the software tools, which were needed to design, implement and verify the mentioned system.

4.5 Development Environment

Software tools facilitated the research, design and implementation of the proposed solution. The most important tools have been selected and are shortly explained below, starting with the Xilinx ISE Design Suite.

4.5.1 Xilinx ISE Design Suite

The Xilinx Integrated Software Environment (ISE) Design Suite [Xilinx, 2013] is a software tool, which offers the possibility to analyze and synthesize a hardware design. One of its features is that it generates Register-Transfer Level (RTL) diagrams out of written code to review the design manually. The free Web Edition supports the whole Spartan series and is therefore highly interesting for researchers and students. Within the Design Suite, multiple tools were used throughout the design, development and analysis of the project. The ISE eases the generation of multiple Intellectual Property (IP) cores, which shorten the development time, as already verified components can be used and adopted to the personal needs of the designer. In addition, it provides the opportunity to synthesize FPGA specific code and generate a program file out of it. As a result of the combination of multiple software tools, the integration of the overall system is eased.

The following steps have been performed with the ISE:

- Synthesis: during this step the HDL design is taken and translated into a device-specific netlist. The HDL code is transferred into blocks like MUX, adders, etc. and irrelevant blocks and nets are removed. Furthermore, a configuration file with timing and other constraints can be incorporated. The result can be seen in a report file and a schematic file.
- Translate: the netlist file from synthesis and further constraint files are mapped.
- Map: the resulting file from translation is used to map the translated design to a specified device.
- Placement and Routing: place all components and perform an iterative routing process to maximize the system quality by reducing the timing delays and provides an equal signal distribution. Furthermore, the generation of a post-place & route simulation model is possible. The tools support specific routing settings, if required.
- Generation of the bit file: in order to configure the target device a bit file is necessary. This process step generates a bit file out of the place and route result.

4.5.2 ModelSim

Another important tool in the development process is ModelSim by Mentor Graphics [Mentor-Graphics, 2013]: ModelSim is a verification and simulation tool for HDL, as Very-High-Speed Integrated Circuits HDL (VHDL) and Verilog. The necessary steps to run a simulation are:

1. create a working library,
2. compile the design files,
3. load the compiled files,
4. run the simulation, and
5. debug the results.

ModelSim was used during the implementation to verify the working principles of the design. Although nowadays simulation is convenient, its results have to be examined critically: when performing a pre-synthesis simulation timing, violations are not identified. Thus, multiple simulations during the implementation process have to be run. The simulation is complex and time-consuming, thus only time spans of several milliseconds are observed in general. ModelSim provides an easy to handle interface with many configuration possibilities and is therefore suitable for beginners and advanced designers. ModelSim requires a project simulation file where all necessary configurations can be set. This is especially important when Mixed-Mode-Simulations are run. In this matter, this was the case: the IP cores provided Verilog code for simulation, whereas the other implementation was done in VHDL. In order to simulate the design, a proper test-bench is required. Details about the verifiction and the test-benches can be found in Chapter 4.6.

The following code shows an example simulation file. Firstly, Verilog and VHDL code files are included and their simulation parameters are defined. Secondly, the simulation is started with specified options and libraries. Finally, the simulation result is represented with aid of a waveform.

```
% include Verilog files
vlog "verilog files.v"

% set parameters
vlog "mig.v" +define+x2Gb + define+sg187E +define+x16

% include VHDL files
vcom -explicit -93 "files.vhd"

% simulate with the defined properties and libraries
vsim -voptargs -L library1 -L lib2 glbl

% show waveform
do wave.do
```

The next section deals with a webcam viewer software which was used to start the system and show a real-time video stream of the results.

4.5.3 WebcamViewer

In order to perform a very basic check of the functionality, a webcam viewer [BustATech, 2013] has been used: the imaging system is recognized by the computer as camera. Due to this fact, the start and stop signals for the system can be generated by every webcam viewer. The Webcam Viewer software has been selected, since it is a freeware, provides the necessary functionality and is an executable, so no further installations are necessary.

The software starts the video stream and receives the image data delivered by the system. The software interpretes the received data as a video data format, known as YCbCr or YUY2. This color space needs only 16 bit per pixel to represent it. The basic idea is that two adjacent pixels share the same U and V information, only the Y value is individual per pixel. Y represents the brightness and U and V the chrominance or color components. The detailed alignment is shown in Figure 4.10. Obviously, this compression leads to information loss.

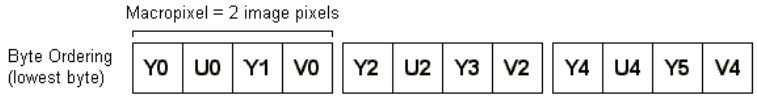


Figure 4.10: YUY2 data format.

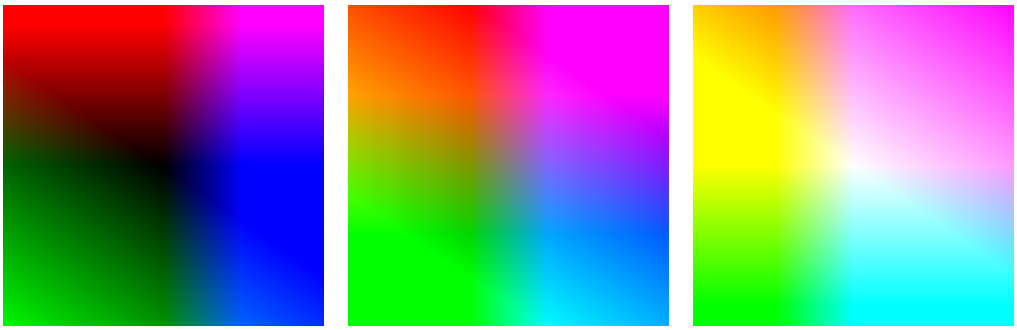
During the feasibility study the idea of sending depth information in form of color decoded pixels was born. The advantage would be that the depth information can be seen directly in a webcam viewer. No further processing would be needed to perform a basic inspection of the scene distance. Obviously this conversion leads to worsening of the Signal Noise Ratio (SNR).

The principle process steps for this conversion are:

1. calculation of the depth value,
2. determination of the regarding RGB image,
3. transformation into YUY2 color space, and
4. transmission to the PC.

The difficulties start with the determination of the RGB image. The result of the phase calculation has to be mapped to a distinct RGB value. The result of the depth calculation is an angle between $-\pi$ and $+\pi$. Most likely not the whole range of angles is containing relevant information. Therefore, an elimination of irrelevant pixels should be performed to obtain a better result.

The resulting RGB value needs to be converted to a YUY2 image, which can be achieved with a basic conjunction. Figure 4.11 shows an illustration of the YUY2 color map, where U can be seen in the x-axis, V in the y-axis and Y is varying in the sub figures. The information loss hereby occurs due to the shared chroma information between two pixels. Pixels, which do not necessaryly share the same chroma value are unitized. Especially for the low supported resolution of 288×352 the consequent error is obvious. Resulting from these disadvantages, this feature has not been realized.



(a) $i_1 = 0$ (b) $i_2 = 0.5$ (c) $i_3 = 1$

Figure 4.11: Illustration of the YUY2 color map with the illuminances i_1, i_2, and i_3.

The qualitative evaluation of the results, the accuracy analysis and the verification process has been performed with the tool Matlab, which is mentioned in the following section.

4.5.4 Matlab

Matlab by [MathWorks, 2013] is a software to solve mathematical problems and represent them graphically. Its main application area is the collection and analysis of large amounts of data. Matlab has an enormous number of application-specific toolboxes and eases the development of own scripts. Another advantage is its access to hardware interfaces to directly access external devices.

In this project Matlab was used to verify the mathematical solution of the depth calculation. The following code shows the starting procedure of system, which is specified as video interface.

```
videoDevice = imaqhwinfo('winvideo');

if (videoDevice.DeviceName =  'FX3')
  device = videoinput('winvideo','YUY2_352x288', 'Tag', 'FX3');
  device.FramesAcquiredFcn =   @processFrame;
  device.FramesPerTrigger  = NUM_ACQ;
  start(device);
end if
```

The method looks for available image acquisition hardware and opens a video connection with the desired data format and resolution to the device. After defining the setting for the handling of the acquired frames, the device is started.

A specified number of frames is collected and stored to perform a qualitative comparison between reference data and received data. Matlab also receives the frames in YUY2 format, but does not modify the frames and interprets pixel information as raw data.

After receiving the frames and storing them, the qualitative verification of the algorithm was performed. Although the code was verified during simulation, a check of the algorithm with real data can only be done when integrated in the overall system. As mentioned in Chapter 4.6, in the verification setup transmits not only z-frames, but also the regarding phase frames. The information, if the received frame is a phase frame or a z-frame is stored in defined pseudo data. Matlab calculates a reference z-frame out of the phase frames and compares it with the received z-frame. Discrepancies entailed further checks and necessary changes in the implementation.

The systematic comparison between different setups and the two algorithms has also been performed with Matlab. The results of this survey have been analyzed and represented graphically. The final results were stored in files to ease further examinations.

A peripheral controller was used handle the communication between devices with different interfaces. The selected controller is illustrated in the next section.

4.5.5 Peripheral Controller

As peripheral controller, the Cypress FX3 controller was used. It serves as bridge between devices with different interfaces. The connection to the external device is configureable with a General Programmable Interface (GPIF). The schematic representation of the FX3 controller is shown in Figure 4.12. In the implemented system, the FX3 controller serves as interface between sensor and PC. Hence, the controller was adjusted to facilitate the communication between the parallel interface and the USB interface. The controller is configurable with a graphical user interface (GUI), where a state machine for the interface can be created [Semiconductor, 2012]. Several state machines for common interfaces are already available in libraries and can be parameterized regarding the requirements. The controller supports synchronous and asynchronous as well as master and slave interfaces. After the generation of the state machine, the corresponding header file can be integrated in the developed firmware framework. The ARM core manages the efficient data processing and enables the data access for the FX3, whereas the DMA controller enables flexible Direct Memory Access. An adapted Eclipse IDE can be used to create and build firmware code. The configuration of the FX3 can be done with a provided control center software over several interfaces - SPI, USB, or I^2C..

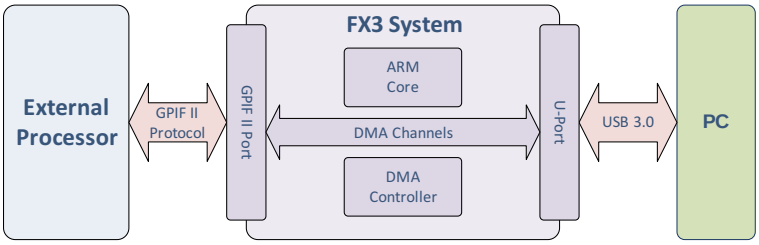


Figure 4.12: General programmable interface [Semiconductor, 2012].

The Xilinx tool iMPACT was used to program the FPGA. Details regarding the tool can be found in the following section.

4.5.6 iMPACT

The tool iMPACT is also part of the Xilinx Design Suite and used to program the synthesized bit file onto the FPGA. Besides checking the status of the device and detecting connection or

supply problems, it also reads the device ID and compares it with the device specified in the bit file. If the status of the device is alright and the device IDs match, it programs the file onto the device. After the configuration a check is performed, to ensure, that the configuration worked out properly. The configuration of the device is done with a JTAG interface. JTAG is the common name of the IEEE (Institute of Electrical and Electronic Engineers) Standard Test Access Port and Boundary Scan Architecture [IEEE, 2001]. This serial interface needs only an input pin (TDI), an output pin (TDO), a clock signal (TCK) and a test mode (TMS) besides the supply and reset signals (TRST). This standard interface can be used to configure not only FPGAs, but also CPLDs (Complex Programmable Logic Devices) and Programable ROMs.

The logic analysis during run-time has been performed with the tool ChipScope, which is listed in the next section.

4.5.7 ChipScope

ChipScope is part of the XILINX toolbox and is a logic analyzer used to view internal signals or nodes of an FPGA at run-time. The monitoring has to be considered during design, since a logic analyzer core has to be inserted. The tool provides the possibility to monitor signals with configurable clock frequency. Due to the fact that in an FPGA specific clocking nets are used to route a clock signal, these signals cannot be monitored. The monitoring can be triggered by aid of multiple trigger signals which can be interrelated. The analyzing core can be configured regarding the specific needs, e.g. number of signals, number of triggers, number of samples, definition of the clock signal. ChipScope uses the JTAG interface to monitor the signal, so no further pins are needed. The monitoring does not influence the functionality and is used to verify the implementation in the overall system.

After the implementation of the system, it has to be verified to guarantee the correctness of its result. The detailed verification process is described in the following section.

4.6 Verification

Verification is a major part in the design and implementation process of hardware: the results have to be compared with reference data to guarantee the accuracy. The verification process can be divided into simulation verification and on-board verification. While in the simulation only self-generated data is used to test the functionality, the on-board verification deals with external data. Both verification steps are necessary to achieve an efficient and correct result.

4.6.1 Simulation Verification

Since the evaluation of the fixed-point calculation returned promising results, the hardware implementation has been focused next. The verification has been done module by module to ensure the proper functionality of every part, before starting the integration of the whole system.

The following modules have been verified on their own, before starting the system verification:

- First of all the **calculation** module has been created according to the design concepts. The input data and the reference results have been stored before. The results of the calculation have been compared with the reference results.
- The next intermediate step was to check if the **data receiver** works properly. To ensure that, the parallel interface had to be implemented. To guarantee the system functionality, it has to be ensured that no data is lost during the reception. Only if every single data word is received, the algorithm will work properly.

- The **data transmitter** uses the meta information to create the necessary signals for the parallel interface. To ensure its functionality, the sync signals have to be activated correctly and output data has to be injected accurately.
- The **memory** is simulated with the aid of a memory model. The model facilitates configuration parameters like speed, size, and bus width. Details about the memory and interfacing can be found in Section 4.1.2. The first test of the memory has been performed with a provided memory sample project. Before the memory can be used, it needs to be calibrated. The calibration is done with help of the MIG module and after successful calibration the memory can be accessed with read- and write-operations.

After ensuring the correct functionality of all system components, the integration and verification of the overall system was performed. Figure 4.13 shows the test-bench of the design top. The stimulus system and the design top are served with the same clock and reset signals. The stimulus system generates random data and transmits the data frame by frame to the tested system. The format of the frame (number of rows and columns) is known. The system receives the phase frames, calculates the z-frame and returns it to the stimulus system.

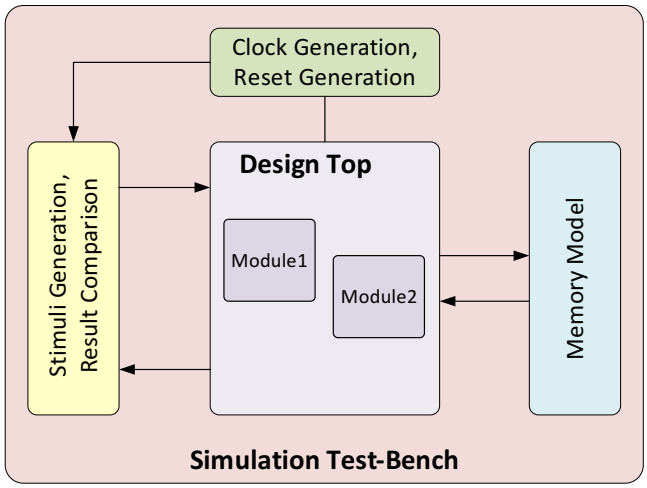


Figure 4.13: Simulation test-bench.

The simulation verification is followed by the system verification where the functionality is tested in its final environment. Details can be found in the next section.

4.6.2 System Verification

The system verification examines the whole system including the sensor, the FPGA, the external memory and the external oscillator, which provides the clock. The goal is to compare the hardware result with the Matlab reference results, see Figure 4.14.

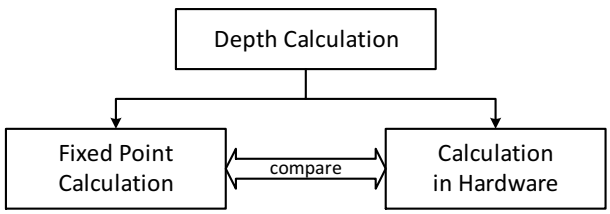


Figure 4.14: Verification hardware calculated results vs. Matlab calculated results.

Figure 4.15 shows the test-bench for the system verification. The PC is triggering the start signals for the sensing unit and the processing device. The data is transmitted from the sensor to the processing device. The clock signal for the processing device is generated by an external oscillator. The FPGA performs the necessary process steps with aid of the DDR3-memory to store the single phase frames. After receiving all phase frames, the calculation is performed and the z-frame is transmitted to the PC. The triggering and data evaluation is done in Matlab.

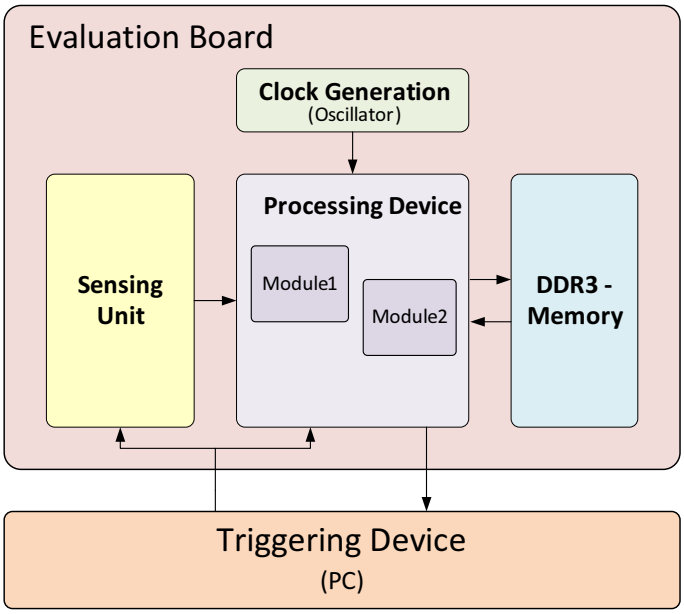


Figure 4.15: Synthesized test-bench.

The setup of the verification procedure is shown in the next section.

4.6.3 Verification Setup

The verification has been performed during different development steps. During the computation verification, the setup has been a simple test-bench in Matlab, which performs checks for the whole input range and compares the results. The next verification has been performed after the hardware development by simulation of the system. After the calculation, the result is passed back to the test-bench, where the result inspection is performed. The final verification is done with real data. Accordingly the implementation is synthesized, mapped, placed and routed and programed onto the FPGA. The sensor is connected to the FPGA to deliver the phase frames and and the FPGA is connected to the FX3 controller to transmit the z-frames to the PC.

In Figure 4.16 a schematic representation of the received and transmitted frames is shown. In the upper part of the picture, the raw data is shown. The frames are received in equidistant timing intervals. The graphic shows the setup for a 4-phase algorithm. Four consecutive frames form a whole z-frame, where the first frame delivers the result for a 0° phase shift, the second frame a 90° phase shift, and so on. During the verification process the four phase frames have been sampled and forwarded to the PC and the resulting z-frame has been transmitted as fifth frame. This procedure permits the checking of the results at the PC. Since the transmission of an additional frame requires time and the necessary delays have to be met, a frame dropping of phase frames cannot be avoided. As mentioned before, only a set of four phase frames leads to a proper calculated z-frame, the next four raw frames are dropped. The reception and forwarding starts again with the next phase frame with a phase shift of 0°. The middle part of the figure shows this timing.

After the successful verification of the results, the transmission of the phase frames is canceled completely and only the resulting z-frame is transmitted to the PC. In this final case, no frame dropping is performed: all phase frames are used to calculate the resulting z-frame. The timing of the final solution is shown in the lower part of Figure 4.16.

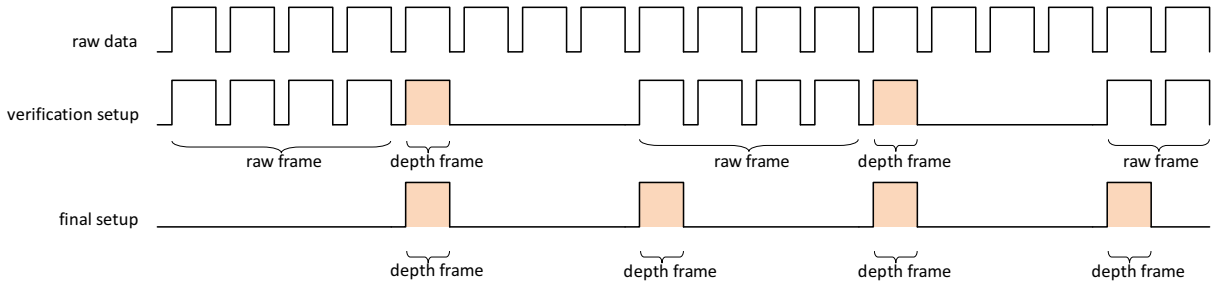


Figure 4.16: Timing diagram of the frame transmission.

The results of the practical implementation can be seen in the next section. Here, the area results and the power consumption is analyzed.

4.7 Implementation Results

In this section, the results of the implementation process are shown: the occupied area in general and each module as well as the maximum system frequency are analyzed.

4.7.1 Area

The area consumption for the implementation hardware can be seen in Figure 4.6. The number of used slice registers is 2,572 which corresponds to 4% of the disposable slice registers. 2,545 slice LUT have been used, which corresponds to 9%. The used area is adequate and its detailed usage is listed below.

	Used Slices	Available Slices	Percentage
Number of Slice Registers:	2,572	54,576	4%
Number of Slice LUTs:	2,545	27,288	9%
Number used as logic:	2,389	27,288	8%
Number used as Memory:	4	6,408	1%
Number used exclusively as route-thrus:	152		

Table 4.6: Slice logic utilization.

In Figure 4.17 the result of the first implementation, without any optimization is shown. The graphic shows the percentual distribution of the slice usage of each module. A main part of the used slices is occupied by the the calculation unit. The calculation unit consists of the CORDIC IP core and the pre- and post-processing of the necessary data. The modules ICON and ILA are the IP cores Integrated Logic Analyzer (ILA) and the Integrated Controller (ICON).

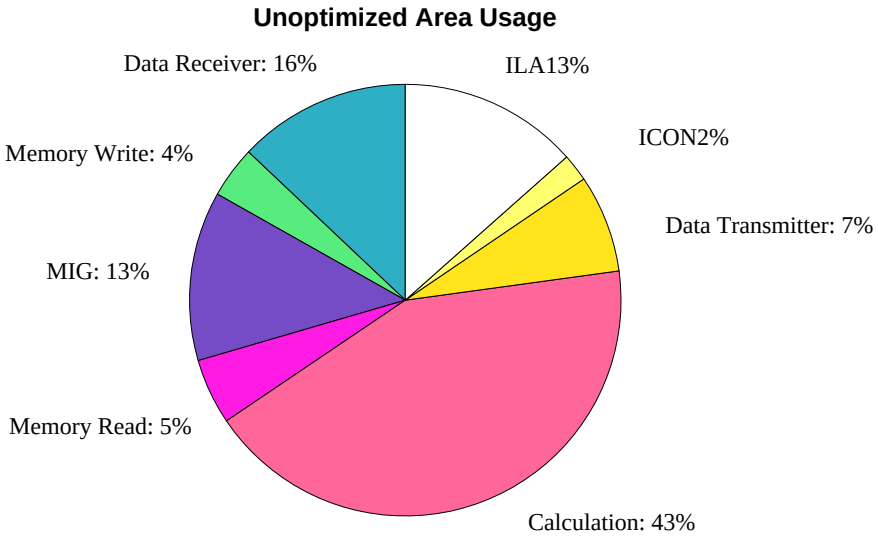


Figure 4.17: Percental distribution of the used slices for the un-optimized solution.

As an optimization for the real system, the logic analyzer cores were removed. These were no longer needed, as soon as the verification succeeded. Furthermore, the buffers were optimized for the implementation and therefore scaled down. In Figure 4.18 the new percentual distribution is shown. An numeric listing is explained below. Due to the reduction of needed slices in some modules, the percentage needed by the calculation unit increased significantly.

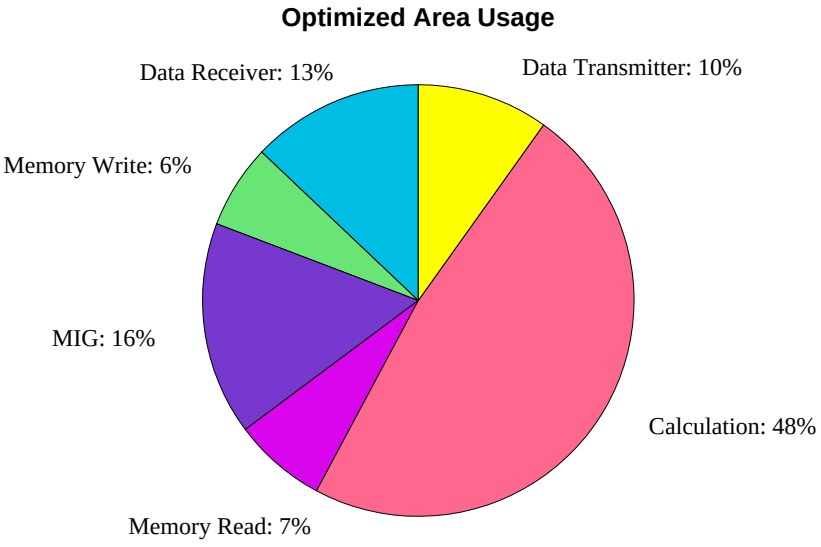


Figure 4.18: Percental distribution of the used slices for the optimized solution.

Table 4.7 shows the numeric area usage of the implemented system divided into its sub-blocks. As can be seen, the biggest part of the whole implementation is the calculation module.

Module	Slices	Slice Reg	LUT
Data Receiver	148	255	269
Memory Write	72	132	127
MIG	183	315	462
Memory Read	80	164	181
Calculation	549	1449	1248
Data Transmitter	113	254	239
ALL	1157	2572	2545

Table 4.7: Area usage of the implemented system.

In order to clarify the area consumption, a detailed list of its sub-components is analyzed in Table 4.8. The main part of the needed slices and LUT is used by the CORDIC core. The core cannot be changed and the input and output sizes are specified, thus no further optimizations are possible. The next big area consumer are buffers. These buffers are needed to hold the pixel data for each phase shift until the calculation can be done. Further analyzing and specifying of the implementation for one setting could reduce this amount. The other sub-modules are the state-machines for each sub-process. The differences in the needed area for the 3-phase and the 4-phase algorithms are negligible, since the other necessary modules are identical.

Sub-Module	Slices	Slice Reg	LUT
CORDIC	290	869	916
Collect Data	6	0	6
Buffer	227	513	270
Post-processing	3	13	6
Pre-processing	10	33	28
Calculation	549	1449	1248

Table 4.8: Area usage of the calculation module.

The next shown results are the power consumption of the FPGA. The consumption has been analyzed with aid of the tool xPower and can be seen in the next section.

4.7.2 Power

The FPGA has been analyzed regarding its used power with the tool XPower from the Xilinx Design Suite. The results can be seen in Table 4.9. The overall power consumption is around 755 mW, in which the main power consumers are the Phase-Locked Loops (PLL). These are necessary to provide all the necessary clocking signals and consume around one third of the necessary power. Three PLLs are used in the whole system: one PLL is needed by the MIG and two PLLs for the generation of the internal clocks. In order to improve the power consumption of the system, a combination of clock signals should be considered. The next big power consumer is the Memory Controller Block (MCB), which handles the communication between the external

memory and the Spartan-6 FPGA. The MCB consumes a fourth of the overall consumed power. Further main power consumers are the Input and Output Blocks (IO) and the clocks. In order to decrease the used power, especially clocking should be investigated in detail.

Resource Type	**Power [mW]**
Clocks	86
Logic	2
Signals	7
BRAMs	6
MCB	189
PLL	278
IOs	135
Leakage	52
Total	755

Table 4.9: Power consumption per resource.

Table 4.10 shows the power consumption of each module. The main power consumer is again the MIG, which requires 284 mW. Another notable power consumer is the design top: the big impact is caused by the used PLLs of the design top. The calculation of the depth image only requires 6.37 mW, which is - compared to the overall consumption - minimal.

Module	**Power [mW]**
Design Top	188.47
Data Receiver	2.46
Memory Write	0.05
MIG	284.55
Memory Read	0.03
Calculation	6.37
Data Transmitter	0.58
Total	481.5

Table 4.10: Power consumption per module.

In order to analyze the system performance the maximal through-put has to be evaluated. The corresponding results can be seen in the next section.

4.7.3 Through-Put

The maximal through-put is limited by the sensing unit, which can deliver maximal 100 phase fps. The effective data rate is therefore 54.0672 Mbps. The maximal achievable data rate regarding the processing unit is 1.2 Gbps. The desired data rate of 1.6 Gbps could not be achieved, although increasing the clocking frequency could lead to more processed data. But since the sensor cannot provide a higher data rate, further investigation to speed-up the processing unit are omitted.

The calculation time for one frame is 1.352 ms, which is done in parallel with receiving and transmission. At maximal clocking frequency a calculation time of 0.3379 ms can be achieved on the FPGA. This time still provides improvement possibilities when considering parallel calculation of pixels. The calculation is performed parallel with the reception and transmission. Therefore, no further limitations occur due to the needed calculation time.

The performed experiments, their pre-conditions and the obtained results can be found in the following chapter.

— CHAPTER FIVE —

Experimental Results

This chapter presents the executed experiments, which were performed after the successful verification of the implemented system. In order to make the experiments reproducible, certain pre-conditions have to be met. The aim of the experiments was to show up the behavior and results of the two implemented algorithms.

5.1 Pre-Conditions

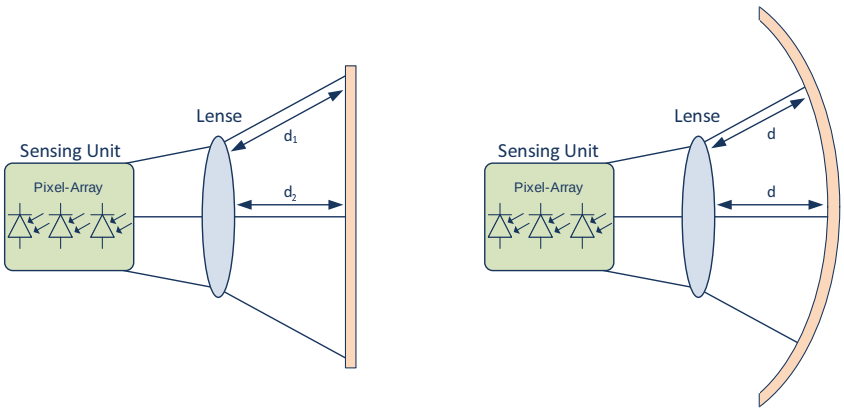
In order to make the experiments reproducible, certain aspects have to be taken into account.

- The influence of ambient light has to be minimized: the easiest way to do this, is to perform the experiment in a shaded measurement chamber.
- Multi-path reflection shall be omitted, which can be done by backing the measurement chamber with light-absorbing material and ensuring, that no other objects are close to the measured scene and influencing the result.
- The temperature influence of the ambience should be constant. Since no measuring cell with temperature compensation is available, this factor cannot be guaranteed to be constant.
- The true distance of the scene shall be constant and known.
- The reflectivity of the captured scene must be identical to avoid errors due to different reflection features.
- Device drifts due to manufactoring processes shall be avoided by using the same measurement setup and devices for all experiments.

In order to meet the pre-conditions, all experiments were performed with the same FPGA board, the same sensing unit and the same illumination unit. The measurements were performed in a shaded box with an equidistant plane. Figure 5.1 shows the resulting radial distance for a straight plane.

Due to the different traveled distances between the center and the outside of the plane, the outer areas appear to be more distant. This distortion is known and can be calibrated by a radial-to-orthogonal conversion. The distance of the plane was known and within the unambiguity range of the sensor. The use of light-absorbing material was dispensed.

The performed experiments are listed in the following section.



(a) Captured Distance of a straight plane. (b) Captured Distance of a parabolic plane.

Figure 5.1: Representation of the calculated distance result for a straight plane and a parabolic plane. The light has to travel a longer distance for the outer parts of the straight plane: the result is an radial blurred distance.

5.2 Experiments

The reason for the performance of the specified experiments was to receive further knowledge about the accuracy and limitation of the algorithms. The settings of each experiment and the results will be explained in this section. All experiments have been performed with the 3-phase-algorithm and the 4-phase algorithm. The results will be discussed regarding known errors. All measurements have been repeated 25 times to receive a statistically representive result. All shown figures are averaged over the performed 25 measurements.

5.2.1 Measurement of an Equidistant Plane

The start setup for the further measurements was the measurement of an equidistant plane for the 3-phase algorithm and the 4-phase algorithm.

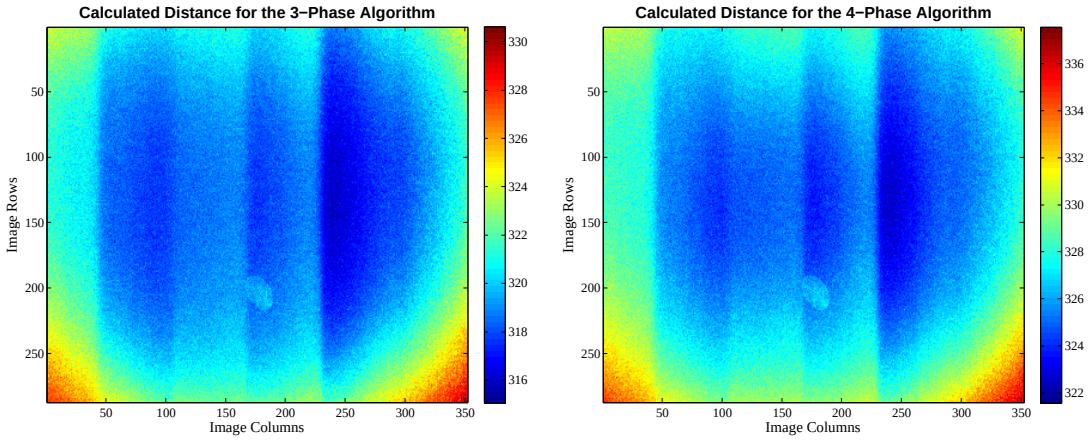
Setting

The ambient light was oppressed with a box. Both measurements were performed with the same modulation frequency f_{mod} and same illumination time t_{ill}. The scene was a white plane with constant reflectivity. The parameters were set to $f_{mod} = 20\,\text{MHz}$ and $t_{ill} = 1.8\,\text{ms}$. The result was received in radiant and back-calculated into a distance with Equation 2.2.

Results and Interpretation

Figure 5.2(a) shows the captured distance image for the 3-phase algorithm and Figure 5.2(b) the result of the 4-phase algorithm. Both results are showing the calculated distance in centimeters. The shown figures have not been calibrated regarding row and column offsets, nor was the radial result converted into orthogonal distance.

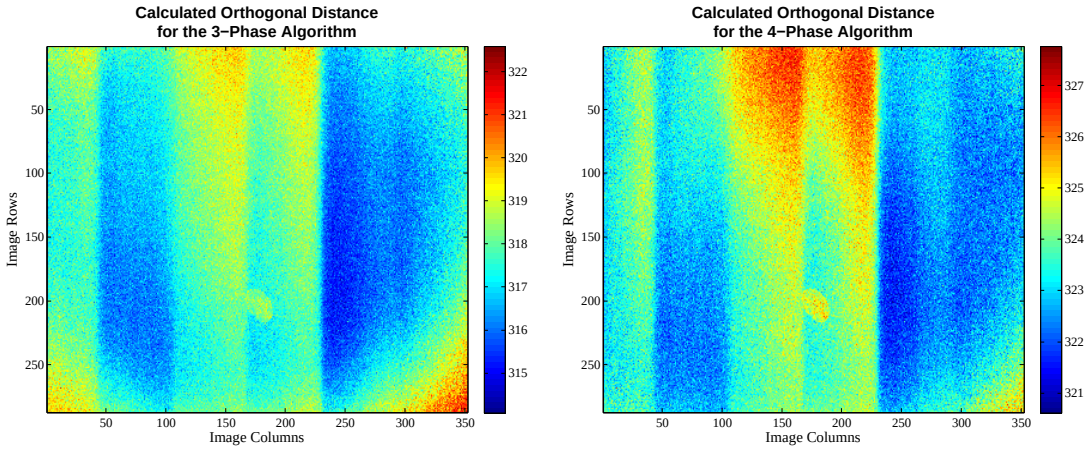
Both figures look quite similar. But at close examination, it can be seen that the 4-phase result differs from the 3-phase result by an almost identical offset of around 6 cm. The expected result is an image with an identical distance value for each pixel. The estimated distance differs between 315 cm and 330 cm for the 3-phase algorithm, which is an 15 cm difference within one image. The most conspicuous differences are listed below:



(a) 3-Phase Algorithm. (b) 4-Phase Algorithm.

Figure 5.2: Calculated distance values for an equidistant plane, without any calibration and transformation. The figure shows the whole captured image, where the color indicates the distance.

One feature is the radial deformation: this is a result of the sensing unit, which captures the radial distance of a scene. The distance covered by the light for the middle area of the image is smaller than for the outer areas. This leads to the impression, that the outer areas are more distant than the central areas. Figure 5.3(a) shows the captured image after a radial to orthogonal transformation. The center has been set to (126, 176). The absolute pixel difference has shrunk from 15 cm to approximately 7 cm within one captured image. Concluding, a radial-to-orthogonal transformation will lead to a more accurate result and decrease the distance error within one received depth image.



(a) 3-Phase Algorithm, whole distance image. (b) 4-Phase Algorithm, whole distance image.

Figure 5.3: Calculated distances values for an equidistant plane after a radial-to-orthogonal conversion. The figure shows the whole captured image, where the color indicates the distance.

In the above figures a distance variation which seems to occur in six equally sized horizontal blocks can be observed. Detailed investigation of the sensing unit led to the reason for the distance variation within one image: the pixel array, which is performing the correlation between emitted and received signal is regulated by six control blocks. The characteristic of the six blocks

is identical and can be observed in Figure 5.4. The x-axis represents the column number and the y-axis the calculated distance centimeters. The lines represent one selected row of the captured image. Each block shows a different offset, but the trend of the distance error is the same within each block: the distance increases constantly by 2 cm from left to right.

Between the 3-phase algorithm and the 4-phase algorithm occurs a significant distance discrepancy of around 6 cm. The assumption is that this discrepancy occurs due to the different impact of the wiggling error. Anterior research presumes that the wiggling error is significantly lower when using the 3-phase algorithm. In order to clarify this theory, further experiments were performed and can be seen in the following section. At this modulation frequency and the measured distance, the wiggling error for the 4-phase algorithm is around 4 cm and the wiggling error for the 3-phase algorithm is around 0.1 cm [Grünwald, 2013]. The remaining difference is subject of further research.

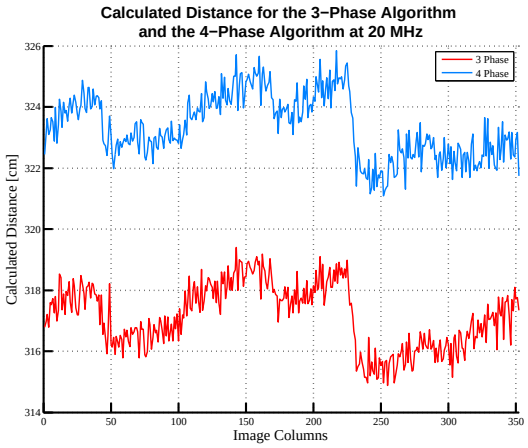


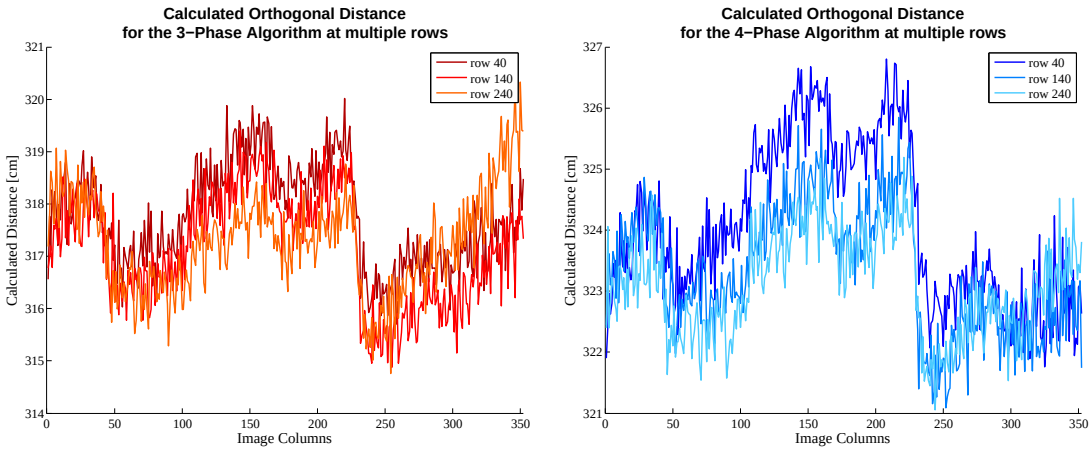
Figure 5.4: Distance value for the 3-phase algorithm and the 4-phase algorithm after radial-to-orthogonal transformation for pixel row 140.

The examination of different rows shows that the offset of the blocks is differing between the rows, see Figure 5.5. It can be observed, that the lower rows appear to be more distant, than the upper rows. This behavior can be observed within the four inner blocks for the 3-phase and the 4-phase algorithm. The outer blocks show a different behavior for the 3-phase and the 4-phase algorithm. While in the 4-phase algorithm the row-wise difference is comparatively low, the 3-phase algorithm shows a discrepancy of around 3 cm within the rows.

Figure 5.6 shows multiple rows of one captured image. As can be seen, the calculated distance of the exterior rows is larger than in the distance of the central row. This difference occurs at both algorithms, although the 3-phase algorithm result shows significantly less variation between each row. The explanation for this difference is the differing row behavior.

Another conspicuity is the oval spot in the image center at around (205, 170), see Figure 5.3. The reason for this inconsistence is a contamination of the sensing array, see Figure 5.7. Its purification has been considered, but due to the sensitivity of the sensor, it was decided to leave it untouched and calibrate it for further processing.

The next performed measurement was performed with varying illumination time and can be found in the follow section.



(a) 3-Phase Algorithm. (b) 4-Phase Algorithm.

Figure 5.5: Calculated distances values for multiple image rows for an equidistant plane after a radial-to-orthogonal conversion.

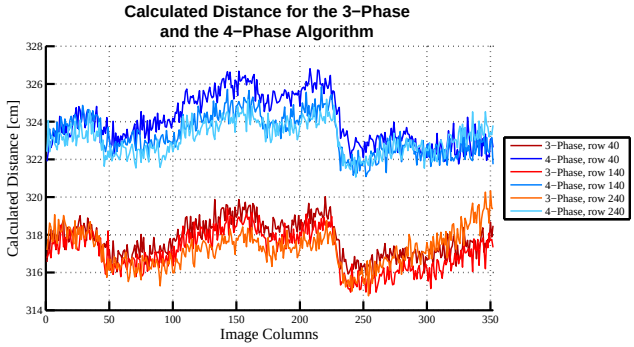


Figure 5.6: Calculated orthogonal distances values for multiple rows of a constant setting with $f_{mod} = 20\,\text{MHz}$ and $t_{ill} = 2.5\,\text{ms}$.

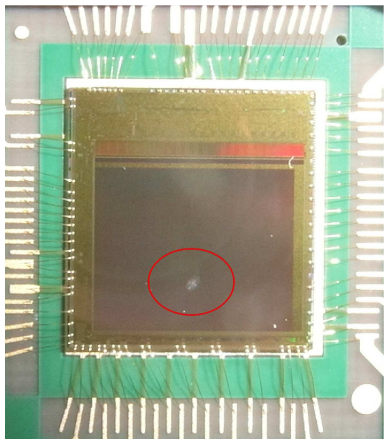


Figure 5.7: Picture of the used sensing unit. The mentioned contamination of the sensor is marked with a circle.

5.2.2 Measurement of a Scene with Differing Integration Time

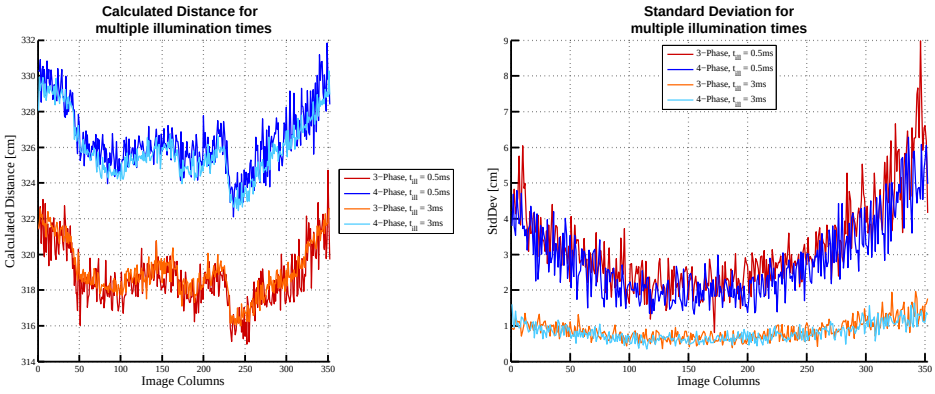
The illumination time influences the accuracy of the calculated depth data: the higher the illumination time is, the more photons are emitted. This increase leads to a higher captured signal strength and further to a superior signal-to-noise ratio (SNR). In general, the higher the illumination time, the higher is the accuracy of the resulting distance. This measurement aims to examine the influence of the illumination time onto the calculated distance.

Setting

The scene is once more a flat plane with a constant distance. The scene is captured within a measurement box to eliminate background illumination. The modulation frequency has been set to the constant value of 20 MHz and the measurement was untouched throughout the capturing of all data. The illumination time t_{ill} varies in the range of several ms and 25 samples have been captured to receive a quantitatively representative result.

Results

In Figure 5.8(a) the calculated distance for $t_{ill} = 0.5\,\text{ms}$ and $t_{ill} = 3\,\text{ms}$ is shown. It can be observed, that the expected distance trend within a row is visible for both illumination times. The further presented images have not been calibrated regarding the true orthogonal distance, since the behavior regarding the varying settings can also be observed in the uncorrected radial distance. Similar to the first measurement, a constant offset between the 3-phase and 4-phase algorithm can be seen.



(a) Result for row 40.

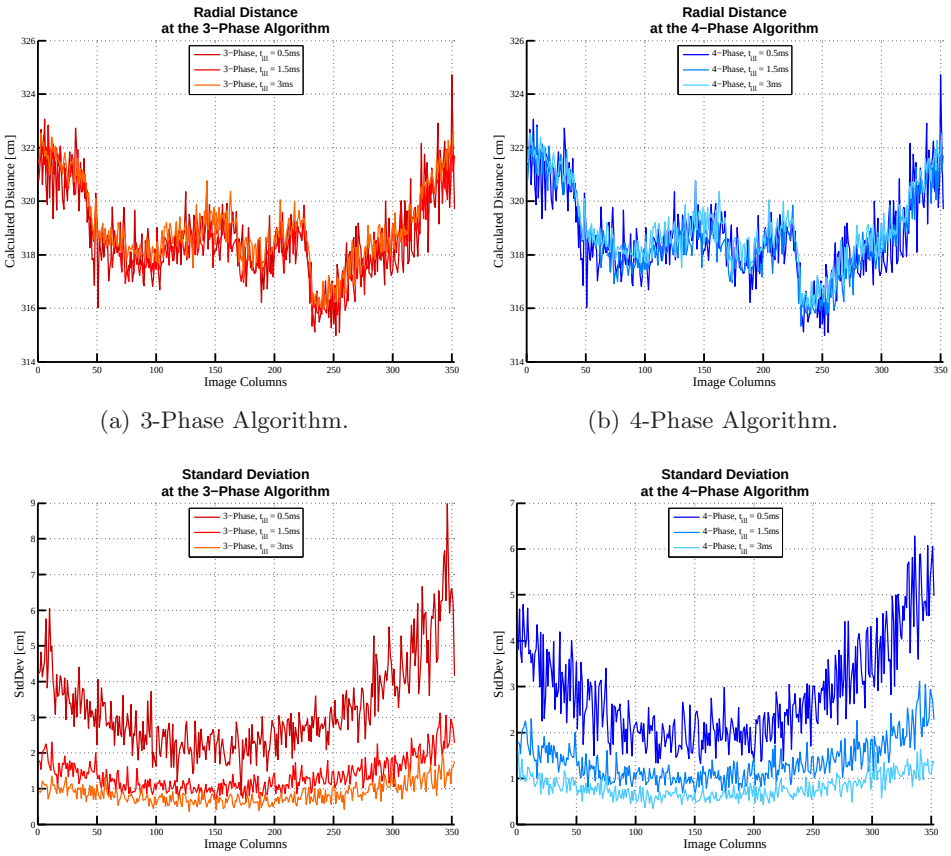
(b) StdDev.

Figure 5.8: Calculated distance and standard deviation for the 3-phase and 4-phase algorithm with varying illumination time.

It can be concluded that the measured distance is not influenced by the illumination time, although the result for a short t_{ill} shows higher pixel discrepancies. Figure 5.8(b) shows the standard deviation for the 25 captured images. It can be observed, that the standard deviation for the higher illumination time is considerably lower than for the low illumination time. The 3-phase and the 4-phase algorithm lead to a similar standard deviation for the same illumination times.

The results for varying illumination times can be seen in Figure 5.9. For both algorithms it can be concluded that the result is more precise, the higher the selected illumination time is. Halving t_{ill} from 3 ms to 1.5 ms leads to a maximal standard deviation of around 2.5 cm. A further dropping of the illumination time to 0.5 ms results in a maximal standard deviation

of around 6 cm. It can be summed up that the illumination indeed influences the accuracy, but no differences between the 3-phase and 4-phase algorithms can be observed. The standard deviation increases constantly from the inner towards the outer columns. The explanation for this behavior is the spot effect of the LED. Due to this fact, the outer areas receive less light.



(a) 3-Phase Algorithm.

(b) 4-Phase Algorithm.

(c) StdDev 3-Phase Algorith.m

(d) StdDev 4-Phase Algorithm.

Figure 5.9: Calculated distances and standard deviations for the 3-phase and 4-phase algorithm with varying t_{ill}.

The characteristic features within one row are visible throughout all illumination times: the influence of the illumination time is therefore significantly less than the wiggling error and the sensor characteristics.

The next performed measurement investigates the influence of differnt modulation frequencies onto the result and can be seen in the following section.

5.2.3 Measurement of a Scene with Differing Modulation Frequency

The system facilitates the use of various modulation frequencies, which can be selected regarding the use case. An examination for different modulation frequencies has been done to specify the behavior of the system.

Setting

The following measurement has been performed with an equidistant plane with a constant distance and varying modulation frequency. The illumination time t_{ill} has been set to 2.5 ms.

The background illumination has been shaded and the measurements were performed with the 3-phase algorithm and the 4-phase algorithm.

Results

The result of the measurement can be seen in Figure 5.10. The figure shows the calculated distance for one row (in this case for row 40) for the modulation frequencies of $f_{mod} = 20\,\text{MHz}$, $f_{mod} = 30\,\text{MHz}$, and $f_{mod} = 40\,\text{MHz}$ and both algorithms. It can be seen that both algorithms lead to different distances dependent on the modulation frequency. The expected result would be the same distance for all modulation frequencies and both algorithms. The variation of the frequency results in 6 cm difference for the 4-phase algorithm and in 3 cm difference for the 3-phase algorithm, where $\Delta d = d_{3-phase} - d_{4-phase}$. When comparing the difference between 3-phase and 4-phase algorithm for the same frequency, it can be seen that the differences Δd_2 and Δd_3 do not have the same absolute value nor the same sign. The result of the 3-phase and the 4-phase algorithm are the same for a modulation frequency of 40 MHz. It can be concluded that the wiggling error is approximately zero for this distance and modulation frequency.

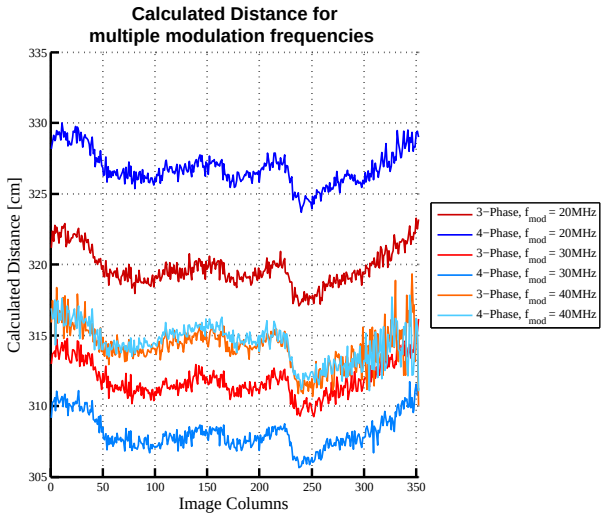


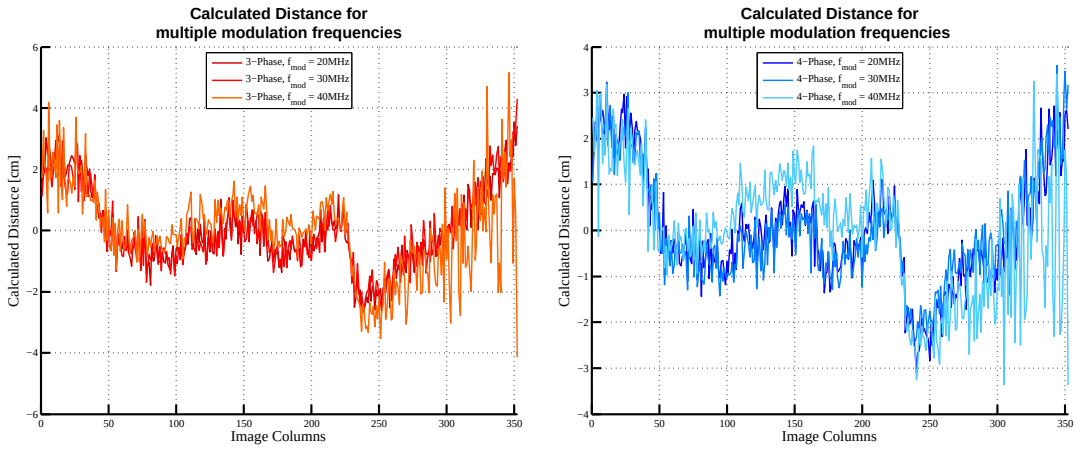
Figure 5.10: Calculated distances values for a setting with varying modulation frequency.

The explanation for this behavior partially lies in the wiggling error and its sinusoidal trend. As expected the distance variation for the 3-phase algorithms is substantially smaller than that of the 4-phase algorithm. This supports the assumption that the 3-phase algorithm results are less influenced by the wiggling error.

The resulting difference between the 3-phase algorithms for different modulation frequencies is caused my a modulation frequency dependant offset of the sensing unit. It can be concluded, that a frequency dependent calibration is necessary to receive the true distance.

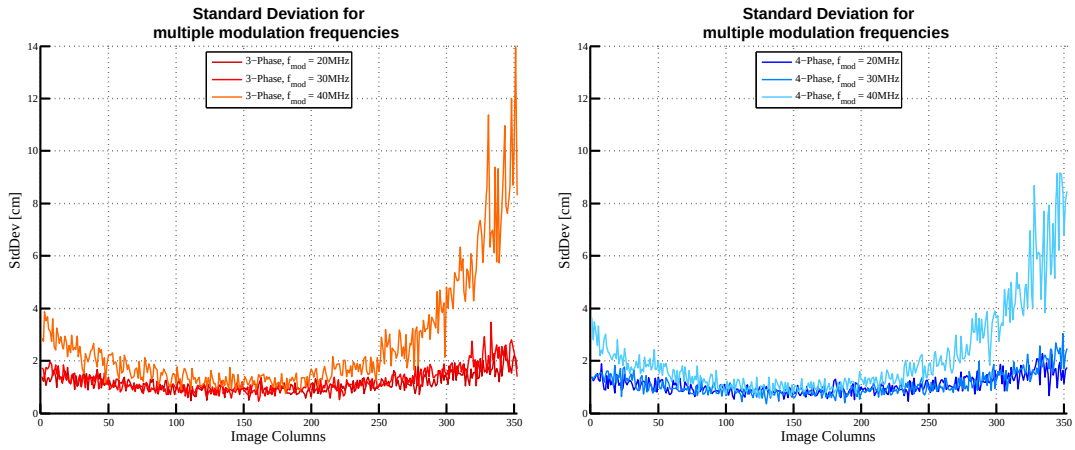
In Figure 5.11 the offset-corrected distances of three modulation frequencies are shown. The calculated distances have been calibrated with their row mean to receive only pixel/column errors. For the 4-phase algorithm it can be observed that the pixel-characteristic for high modulation frequencies drifts from the pixel-characteristic for low to medium frequencies. For the 3-phase algorithm, the result is much more consistent, although discrepancies still occur. The discrepancies are highest for the right-most columns. In this area, the result shows a very high standard deviation.

The presumption is that this deviation is caused by the side walls of the shading box, which are closest at this area. Multi-path reflections are disturbing the measurement significantly. In



(a) 3-Phase Algorithm.

(b) 4-Phase Algorithm.



(c) StdDev 3-Phase Algorithm.

(d) StdDev 4-Phase Algorithm.

Figure 5.11: Comparison of the 3-phase algorithm (a) and the 4-phase algorithm (b) with different modulation frequencies. (d) and (d) are the regarding deviations.

order to not fudge the results with improper data, only the image area with a maximal standard deviation of 4 cm is contemplated. The result for the 4-phase algorithm shows also a high deviation for the 40 MHz measurement in the inner blocks.

It can be summarized that the 3-phase algorithm delivers better results in terms of identical gradient for the considered modulation frequencies. The standard deviation is approximately the same for the 3-phase and the 4-phase algorithm. This behavior is subject for further research, since an increased standard deviation for the 3-phase algorithm has been expected. The standard deviation is also influenced by other effects: in this case by multi-path reflections at the highest frequency.

In a real scenario, the sensor will capture an image with diverging distances. Therefore, the next measurement explores the influence of different distances. Details are shown in the next section.

5.2.4 Measurement of a Scene with Different Distances

In order to receive further insight for different distances, an experiment with varying distances has been performed. The aim is to verify the assumption that the 3-phase algorithm will deliver more accurate data for varying distances.

Setting

The experiment has been performed with a shaded setting and a flat, equidistant plane. The illumination time was set to $t_{ill} = 2.5\,\text{ms}$ and the modulation frequency to $f_{mod} = 20\,\text{MHz}$.

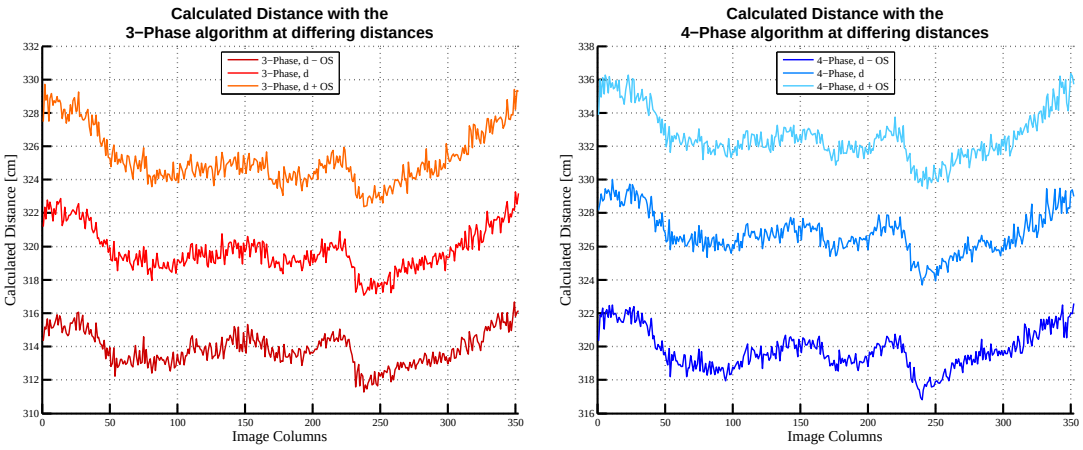
In this experiment three distances have been measured:

- d_1: a flat plane with distance d + offset,
- d_2: a flat plane with distance d, and
- d_3: a flat plane with distance d - offset.

As in the anterior experiments, the measurement has been repeated several times to receive a meaningful average. The distance variation is restricted by the illumination unit. In order to receive highly accurate data for distant scenes, a highly accurate and intense illumination is needed.

Results

The results of the mentioned setting can be seen in Figure 5.12.



(a) 3-Phase Algorithm.

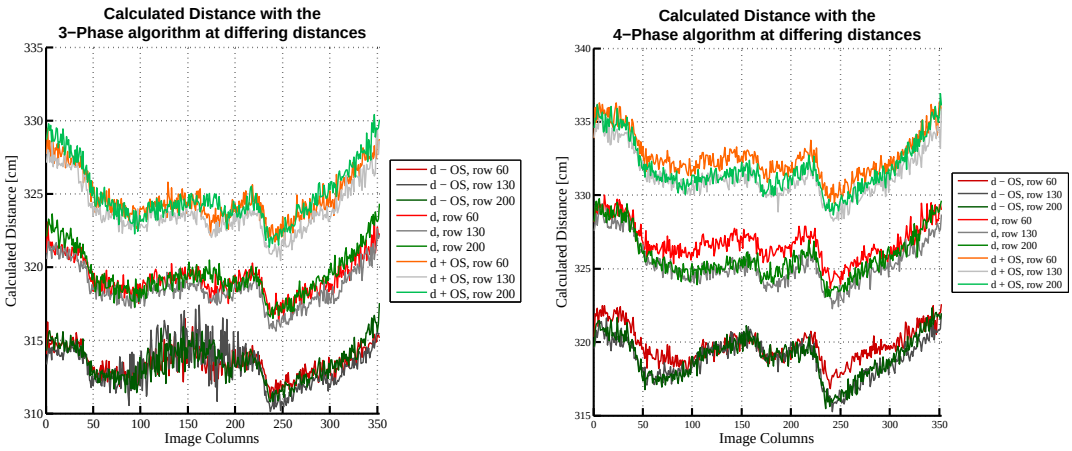
(b) 4-Phase Algorithm.

Figure 5.12: Comparison of 3-phase and 4-phase algorithm for different distances.

Figure 5.12(a) shows the resulting values for the 3-phase algorithm, and Figure 5.12(b) shows the distance for the 4-phase algorithm. As can be seen, the distance change is constant in all pixels of the shown row. The applied offset has been identical in absolute value, therefore the expection is to receive three rows with the same offset. The resulting distance offset for the 3-phase algorithm is exactly $\pm$ 6 cm. This result fulfills the expectations. When examining the result for the 4-phase algorithm, the figure shows a distance offset of +5 cm and -6 cm. The reason for this behavior can again be found in the wiggling error, which differs with the difference of the scene. The pixel dependent errors are not effected by the distance. The observed distance change within one row is identical for all examined distances.

The previous measurements have shown differing distances for different rows. Therefore, a comparison of multiple rows is given: Figure 5.13 shows these results. It can be observed that the distance change between each row is exactly 6 cm in the 3-phase algorithm. This leads to the conclusion that a distance change has the same impact on each pixel. A high distance variation within the two inner blocks can be observed for the closest distance. The reason for

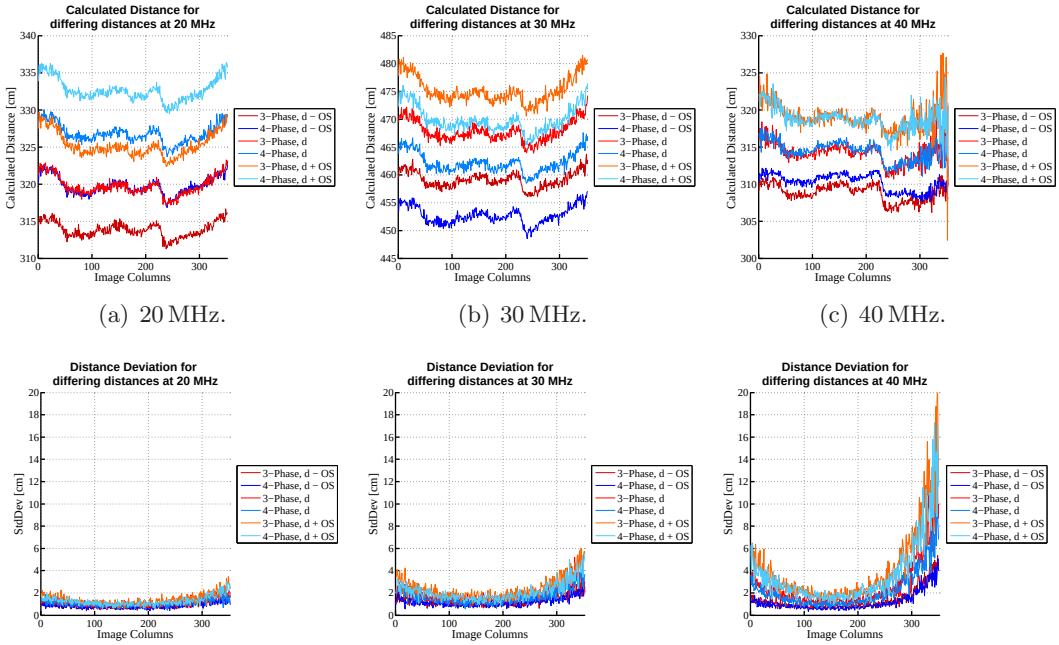
this behavior are saturation effects. The behavior for the 4-phase result is more inconsistent: the central row shows an identical behavior for all distances, whereas the distances for the outer rows differ especially for the closest distance.



(a) 3-Phase Algorithm. (b) 4-Phase Algorithm.

Figure 5.13: Calculated distance for multiple rows at three distances.

Figure 5.14 shows a representation of the influence of a distance change when varying the modulation frequency.



(a) 20 MHz. (b) 30 MHz. (c) 40 MHz.

(d) StdDev 20 MHz. (e) StdDev 30 MHz. (f) StdDev 40 MHz.

Figure 5.14: Distance and standard deviation of distance variations for different frequencies.

In order to guarantee the accuracy of the calculated distance, the standard deviation has been plotted under each distance plot. The experiment with the modulation frequency of $f_{mod} = 20\,\text{Mhz}$ leads to an interesting result. The calculated distance of the 3-phase algorithm for distance d_2 and the calculated distance of the 4-phase algorithm for distance d_3 are almost

the same, see Figure 5.14(a). Similarities can be observed for distance d_1 and d_2. The conjectured reason for this phenomenon is the wiggling error, which is why the measurement has been redone with modulation frequencies of $f_{mod} = 40$ Mhz and $f_{mod} = 30$ Mhz.

In Figure 5.14(c) the results for $f_{mod} = 40$ MHz is shown. The difference between 3-phase and 4-phase algorithm is minimal. At the most-left part of the image, a high variation and a incredibly high standard deviation of 20 cm can be seen. As in the above measurement, the shading box is influencing the measurement. The effect of the multi-path error increases with the modulation frequency. The high equality in the 40 MHz result is due to the fact that the absolute distance error of the wiggling error decreases with increasing modulation frequency.

The measurement, which is shown in the next section, aims to study the influence of motion onto the resulting distance images.

5.2.5 Motion Measurements

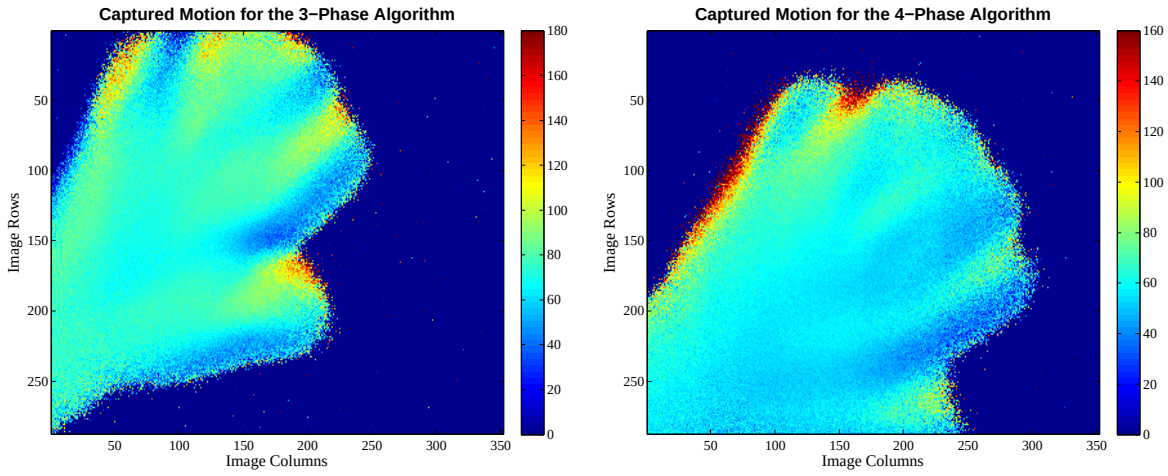
A ToF sensor is mostly used to detect distance changes, which are often motion. Due to the time-delayed capturing of the phase images, often an erroneous distance is calculated for the area of the motion. This effect is called motion artifacts. The influence of the two algorithms is studied.

Setting

The setting is a scene with an equidistant background and motion in the foreground. The illumination time is set to 1.5 ms and modulation frequency to 20 MHz. No shading of surrounding light has been performed. A hand has been moved with constant speed in front of the sensor to simulate motion.

Results

The results can be seen in Figure 5.15. The artifacts can be seen at the edges of the fingers.



(a) 3-Phase Algorithm.

(b) 4-Phase Algorithm.

Figure 5.15: Captured motion for the 3-phase and the 4-phase algorithm.

The occurrence of motion artifacts cannot be avoided with the implemented system. Due to the fact that three respectively four phase frames have to be captured successively, motion which occurs within the time needed to capture all phase frames is visible. The motion is visible in

both calculation methods. These artifacts may lead to a misinterpretation of gesture recognition algorithms. Motion artifacts appear especially in scenes with fast movements.

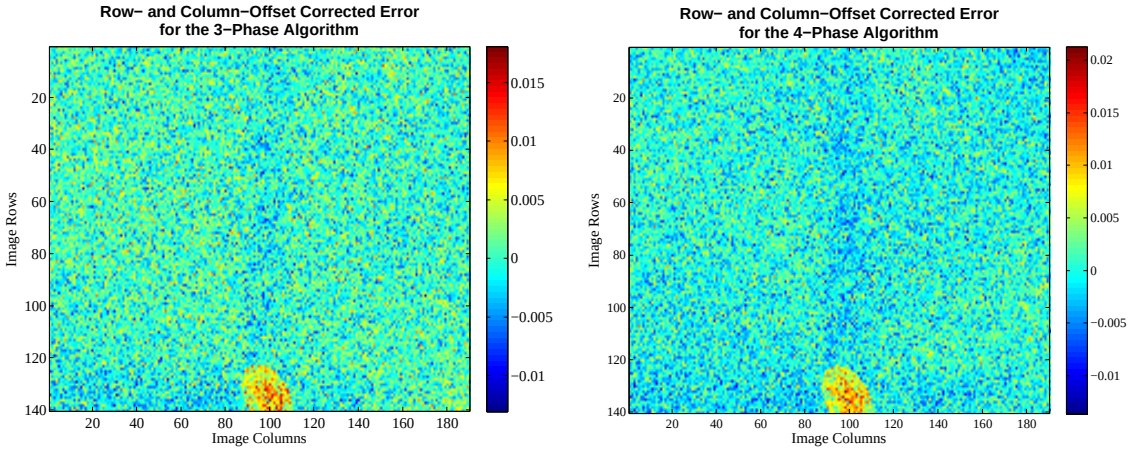
The effect of pixel-dependent errors could be seen in the previous measurments. The next section deals with a proposed calibration method to decrease the effect of FPPN.

5.2.6 FPPN Calibration

In order to calibrate the row- and column-wise offset of the captured image, an average correction for each row and column was performed. Therefore, a calibration image was taken and its column- and row-wise mean was calculated. This mean-values were used afterwards to correct each pixel value. The procedure was performed as follows:

1. computate the correction image,
2. take another distance image, and
3. subtract the correction image from the captured distance image in order to receive the true distance.

The result can be seen in Figure 5.16. The figure shows the resulting offset in centimeters. It can be seen that a row- and column-wise average error correction can minimize the pixel variations significantly. The resulting offsets are in the range of $\pm$ 0.2 mm. The stain in the lower center of the image is the above mentioned contamination.



(a) 3-Phase Algorithm.

(b) 4-Phase Algorithm.

Figure 5.16: Inner part of an average corrected image.

It can be concluded that the calibration of the FPPN is necessary to receive an accurate distance. The performing of a row- and column-wise calibration is effective und leads to proper results.

The following section gives a resumé of the performed experiments and their results.

5.3 Resumé

Various experiments have been performed to identify the characteristic of the 3-phase and the 4-phase algorithm. The gained insights are sumed up in the following paragraphs.

5.3.1 Illumination Time

For both algorithms can be concluded that the illumination time does not influence the calculated depth. The illumination time has been varied between 0.5 ms and 3 ms. Further increase of the illumination time leads to saturation effects and can therefore not be considered. The smaller the selected illumination time is, the higher is the standard deviation of the returned distance.

5.3.2 Modulation Frequency

The variation of the modulation frequency leads to differing results in the 3-phase algorithm and the 4-phase algorithm. Although the captured scene has stayed untouched, both algorithms show a different behavior: the distance difference between $f_{mod} = 20\,\text{MHz}$ and $f_{mod} = 40\,\text{MHz}$ for the 4-phase image is higher by the factor of 2 than the 3-phase algorithm. This behavior was expected by signal theory, since research revealed that the wiggling error is considerably lower for the 3-phase algorithm. In order to identify the exact influence of the wiggling error for the 3-phase algorithm a detailed measurement sweeping modulation frequency has to be performed.

5.3.3 Distance Variation

Distance changes are delivering different results for the 3-phase and the 4-phase algorithm. While a distance variation of 5 cm results in a $\pm$ 5 cm difference in the 3-phase algorithm, the same variation returns a +5 cmand -7 cm difference. This inconsistency is due to the wiggling error.

5.3.4 Motion

Measurements have shown that motion artifacts occur for both algorithms. This is due to the fact that the 3-phase and the 4-phase measurement require multiple raw frames which are captured consecutively in time. The emerging motion errors differ for both algorithms.

5.3.5 FPPN

The FPPN of both algorithms is identical for measurements with different illumination times. The 3-phase algorithm shows more consistent FPPN when changing the modulation frequency. Especially high modulation frequencies lead to highly differing FPPN in the 4-phase algorithm, whereas no such behavior is visible in the 3-phase algorithm. The FPPN changes from row to row, although influence of the six control boxes are still visible.

The following chapter provides a conclusion of this thesis and illustrates future work.

— CHAPTER SIX —

Conclusion and Future Work

6.1 Conclusion

3D imaging techniques in general have to deal with high amounts of data. This applies also to ToF systems. The amount of data can be decreased when performing an early calculation of the depth data. Other approaches pursue the efficient selection of regions of interest to avoid the transfer of unnecessary data. The designed and implemented systems include two signal processing algorithms: the first one is a state of the art 4-phase algorithm, which uses raw images captured with the phase shift of 0°, 90°, 180°, and 240° to compute the distance. The second implemented algorithm is a novel algorithm, which needs only three raw images with a phase shift of 0°, 120°, and 240°.

In order to minimize the data rate to the PC and diminish the latency, the implemented distance calculation was performed on an FPGA. Due to the high speed requirements, the system was implemented with finite state machines in VHDL. The designed system receives data from the image sensor, calculates the depth information and forwards these to the requesting device, in this case to the PC. The amount of data which needs to be handled exceeds the available internal memory. Thus the raw frames are temporarily stored in an external DDR3-memory. The distance calculation was performed with aid of an arc tangent calculation with 16 bit fixed-point data types. The loss of accuracy due to the limited available bit-width was estimated and analyzed before the start of the implementation.

Since the distance differences were in a tolerable range, the distance calculation was performed on the FPGA. The arc tangent calculation was implemented with a Cordic IP Core. The result analysis and verification was performed with Matlab.

The designed and implemented system is scaleable and can automatically handle z-frames with variable size, which offers a high degree of flexibility and distinguishes it from related work.

Subsequent to the practical implemention, multiple measurements were performed to compare the two mentioned algorithms. It could be shown that the 3-phase algorithm leads to an enhancement in terms of a decreased effect of the wiggling error. This could be shown by performing measurements with variable distance and variable modulation frequency. The algorithms react identical to changes in illumation time. The FPPN is almost identical for both algorithms. In the resulting distance images the six control blocks of the sensing unit can be detected. This offset can by eliminated either by dark image calibration or row- and column-wise corrections.

It could be generally shown that the implementation in hardware leads to load removal of the PC interfaces. Since the USB interface usually is not real-time capable, it is a limiting factor in the system's performance. The designed and implemented system could decrease the data rate towards the PC to a third respectively a fourth.

The next section gives an outlook on upcoming research and projects to be accomplished.

6.2 Future Work

In this work, a first prototype for depth calculation of ToF sensors in hardware is shown. Further improvement would enhance the solution and ease the way into mass production. Prospective projects might include:

- **calibration** of known errors on the FPGA: the sensing unit delivers raw data, which needs to be calibrated in order to achieve highly accurate depth maps. The calibration parameters, which are obtained from measurement can by supplied into the FPGA. Afterwards, the FPGA can perform calibration steps to balance known offsets.
- **control** and start the sensor core from the FPGA: Currently the control setup and starting sequence is triggered by the PC. When controlling the sensor from the FPGA, a higher flexibility could be achieved since the FPGA could automatically configure the setting and deliver more accurate data.
- extract **regions of interest** (ROI): during most image acquisitions, only parts of the image will change whereas the bigger part stays unchanged. By identifying the ROI of the image and just transmitting new information, only relevant data will reach the PC. The advantage is that no irrelevant data is transmitted and hence no processing time of the PC is wasted.
- further **power reductions** by optimizing the clocking frequencies regarding the specified use case.
- **integration** into the ASIC: in the future, the calculation and depth calculation process could be done onto the IC. This would lead to holistic system where no further components besides the sensing unit and the illumination unit are needed. This process requires detailed research and advanced calibration techniques.

Further work and improvement in this field of research would be highly appreciated, to ease on the one hand the use of ToF cameras and widen on the other hand their field of application.

Bibliography

[3DV, 2013] 3DV (2013). 3DV Website.

[Albrecht, 2007] Albrecht, M. (2007). *Untersuchung von Photogate-PMD-Sensoren hinsichtlich qualifizierender Charakterisierungsparameter und -methoden*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Berlin.

[Andraka, 1998] Andraka, R. (1998). A Survey of CORDIC Algorithms for FPGA based Computers. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, pages 191–200.

[Bailey, 2002] Bailey, D. G. (2002). A new Approach to Lens Distortion Correction). In *Proceedings Image and Vision Computing New Zealand 2002*, pages 59–62.

[BustATech, 2013] BustATech (2013). Bust A Tech Webcam Viewer Website.

[Buttgen et al., 2006] Buttgen, B., Lustenberger, F., and Seitz, P. (2006). Demodulation Pixel Based on Static Drift Fields. *Electron Devices, IEEE Transactions on*, 53(11):2741–2747.

[Drayton, 2013] Drayton, B. M. (2013). *Algorithm and Design Improvements for indirect Time of Flight Range Imaging Cameras*. PhD thesis, Victoria University of Wellington.

[Einramhof et al., 2007] Einramhof, P., Olufs, S., and Vincze, M. (2007). Experimental Evaluation of State of the Art 3D-Sensors for Mobile Robot Navigation 1). In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1-3, pages 191–198.

[Ercegovac and Lang, 2004] Ercegovac, M. D. and Lang, T. (2004). *Digital Arithmetic*. Morgan Kaufmann Publishers.

[Foix et al., 2010] Foix, S., Alenya, G., and Torras, C. (2010). Exploitation of Time-of-Flight (ToF) Cameras. Technical report, IRI.

[Foix et al., 2011] Foix, S., Alenya, G., and Torras, C. (2011). Lock-in Time-of-Flight (ToF) Cameras: A Survey. *Sensors Journal, IEEE*, 11(9):1917 –1926.

[Frank et al., 2009] Frank, M., Plaue, M., and Hamprecht, F. A. (2009). Denoising of Continuous-Wave Time-Of-Flight Depth Images using Confidence Measures. *Optical Engineering*, 48(7).

[Fuchs, 2012] Fuchs, S. (2012). *Calibration and Multipath Mitigation for Increased Accuracy of Time-of-Flight Camera Measurements in Robotic Applications*. PhD thesis, Department of Electrical Engineering and Computer Science, Technical University of Berlin.

[Fuchs and Hirzinger, 2008] Fuchs, S. and Hirzinger, G. (2008). Extrinsic and Depth Calibration of ToF-Cameras. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:1–6.

[Ghobadi et al., 2006] Ghobadi, S., Hartmann, K., Weihs, W., Netramai, C., Loffeld, O., and Roth, H. (2006). Detection and Classification of Moving Objects-Stereo or Time-of-Flight Images. In *Computational Intelligence and Security, 2006 International Conference on*, volume 1, pages 11–16.

[Giannakopoulou and Masselos, 2012] Giannakopoulou, V. and Masselos, K. (2012). Hardware Performance Analysis of a Parametric CORDIC IP. In *Signals and Electronic Systems (ICSES), 2012 International Conference on*, pages 1–6.

[Grünwald, 2013] Grünwald, J. (2013). Investigation of Systematic Errors in Time-of-Flight Imaging. Master's thesis, Graz University of Technology.

[Haker, 2010] Haker, M. (2010). *Gesture-Based Interaction with Time-of-Flight Cameras*. PhD thesis, Institute for Neuro- und Bioinformatics, University of Luebeck.

[Hansard et al., 2012] Hansard, M., Lee, S., Choi, O., and Horaud, R. P. (2012). *Time of Flight Cameras: Principles, Methods, and Applications*. SpringerBriefs in Computer Science. Springer.

[Heinol, 2001] Heinol, H. G. (2001). *Untersuchung und Entwicklung von modulationslaufzeitbasierten 3D-Sichtsystemen*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen.

[Hernandez et al., 2008] Hernandez, A., Gardel, A., Perez, L., Bravo, I., Mateos, R., and Sanchez, E. (2008). Real-Time Image Distortion Correction using FPGA-based System. In *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pages nil7–nil11.

[Hsu et al., 2006] Hsu, S., Acharya, S., Rafii, A., and New, R. (2006). Performance of a Time-of-Flight Range Camera for Intelligent Vehicle Safety Applications. In *Advanced Microsystems for Automotive Applications 2006*, pages 205–219. Springer Berlin Heidelberg.

[IEEE, 2001] IEEE (2001). 1149.1-2001 - IEEE Standard Test Access Port and Boundary Scan Architecture.

[Infineon, 2013] Infineon (2013). 3D Image Sensor - Infineon.

[Jongenelen et al., 2008] Jongenelen, A. P., Carnegie, D. A., Dorrington, A. A., and Payne, A. D. (2008). Heterodyne range imaging in real-time. *Proceedings of International Conference on Sensing Technology*, pages 57–62.

[Karel and Pfeifer, 2009] Karel, W. and Pfeifer, N. (2009). Range camera calibration based on image sequences and dense comprehensive error statistics. *Proceedings of Three-Dimensional Imaging Metrology*, 7239:72390D–72390D–12.

[Kolb et al., 2008] Kolb, A., Barth, E., and Koch, R. (2008). ToF-Sensors: New Dimensions for Realism and Interactivity. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–6.

[Kollorz et al., 2008] Kollorz, E., Penne, J., Hornegger, J., and Barke, A. (2008). Gesture Recognition with a Time of Flight Camera. *Int. J. Intell. Syst. Technol. Appl*, 5(3/4):334–343.

[Lange, 2000] Lange, R. (2000). *3D Time-of-Flight Distance Measurement with custom Solid-State Image Sensors in CMOS/CCD Technology*. PhD thesis, Department of Electrical Engineering and Computer Science at University of Siegen.

[Lange et al., 1999] Lange, R., Seitz, P., and Schwarte, R. (1999). Time-of-Flight Entfernungskamera in CMOS/CCD-Technik mit pixelintegrierten Lock-in Verstärkern. *Messen in der Fertigung: 3D-Meßtechnik in Produktion und Entwicklung*, pages 271–280.

[Luan, 2001] Luan, X. (2001). *Experimental Investigation of Photonic Mixer Device and Development of TOF 3D Ranging Systems Based on PMD Technology.* PhD thesis, Department of Electrical Engineering and Computer Science.

[Marr and Poggio, 1979] Marr, D. and Poggio, T. (1979). A Computational Theory of Human Stereo Vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 204(1156):301–328.

[MathWorks, 2013] MathWorks (2013). MathWorks Website.

[May et al., 2009] May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nuechter, A., and Hertzberg, J. (2009). Three-Dimensional Mapping with Time-of-Flight Cameras. *J. Field Robot.*, 26(11/12):934–965.

[MentorGraphics, 2013] MentorGraphics (2013). MentorGraphics ModelSim Website.

[Microsoft, 2013] Microsoft (2013). Microsoft xbox kinect.

[MIPI, 2013] MIPI (2013). MIPI CSI-2 Specification.

[Oh et al., 2013] Oh, K., Hwang, S.-H., You, S., Cho, J., Jeon, M., and Kim, M.-K. (2013). Gesture Sensor for Mobile Devices. Technical report, Samsung Electronics.

[Pantech, 2013] Pantech (2013). Pantech Website.

[Penne et al., 2008] Penne, J., Schaller, C., Joachim, H., and Kuwert, T. (2008). Robust real-time 3D respiratory motion detection using time-of-flight cameras. *International Journal of Computer Assisted Radiology and Surgery*, 3.

[PMDTechnologies, 2012] PMDTechnologies (2012). PMD[vision] CamCube 3.0. http://www.pmdtec.com/products-services/pmdvisionr-cameras/pmdvisionr-camcube-30/.

[Poggio and Poggio, 1984] Poggio, G. F. and Poggio, T. (1984). The Analysis of Stereopsis. *Annual Review of Neuroscience*, 7(1):379–412.

[PrimeSense, 2013] PrimeSense (2013). PrimeSense Website.

[Rapp, 2007] Rapp, H. (2007). Experimental and Theoretical Investigation of Correlating TOF-Camera Systems. Master's thesis, University of Heidelberg, Germany.

[Samsung, 2013] Samsung (2013). Samsung Website.

[Schaller, 2011] Schaller, C. (2011). *Time-of-Flight - A New Modality for Radiotherapy.* PhD thesis, University of Erlangen-Nürnberg, Germany.

[Schmidt, 2011] Schmidt, M. (2011). *Analysis, Modeling and Dynamic Optimization of 3D Time-of-Flight Imaging Systems.* PhD thesis, Ruperto-Carola University of Heidelberg, Germany.

[Schmidt, 2008] Schmidt, M. O. (2008). *Spatiotemporal Analysis of Range Imagery.* PhD thesis, Ruperto-Carola University of Heidelberg, Germany.

[Schneider, 2003] Schneider, B. (2003). *Der Photomischdetektor zur schnellen 3D-Vermessung fuer Sicherheitssysteme und zur Informationsuebertragung im Automobil.* PhD thesis, Department of Electrical Engineering and Computer Science at University of Siegen.

[Schwarte et al., 1999] Schwarte, R., Heino, H., Buxbaum, B., Ringbeck, T., Xu, Z., and Hartmann, K. (1999). Principles of Three-Dimensional Imaging Techniques. *Handbook of computer vision and applications*, pages 463–484.

[Schwarz et al., 2011] Schwarz, L., Mkhitaryan, A., D., M., and Navab, N. (2011). Estimating human 3D pose from time-of-Flight images based on geodesic distances and optical flow. In *Automatic Face Gesture Recognition and Workshops (FG 2011, 2011 IEEE International Conference on*, pages 700–706.

[Semiconductor, 2012] Semiconductor, C. (2012). *GPIF II Designer 1.0*.

[Spickermann, 2010] Spickermann, A. (2010). *Photodetektoren und Auslesekonzepte für 3D-Time-of-Flight-Bildsensoren in 0,35-um-Standard-CMOS-Technologie*. PhD thesis, Fraunhofer Institute for Microelectronic Circuits and Systems, University of Duisburg-Essen.

[Van den Bergh and Van Gool, 2011] Van den Bergh, M. and Van Gool, L. (2011). Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 66–72.

[Vladimirova and Tiggeler, 1998] Vladimirova, T. and Tiggeler, H. (1998). FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm. In *Proceedings of Military and Aerospace Applications of Programmable Devices and Technologies International Conference (MAPLD'99)*.

[Volder, 1959] Volder, J. (1959). The CORDIC Computing Technique. In *Papers presented at the the March 3-5, 1959, western joint computer conference*, IRE-AIEE-ACM '59 (Western), pages 257–261, New York, NY, USA. ACM.

[Xilinx, 2010a] Xilinx (2010a). *Memory Interface Solutions*.

[Xilinx, 2010b] Xilinx (2010b). *Spartan-6 FPGA Memory Controller*.

[Xilinx, 2013] Xilinx (2013). Xilinx ISE Design Suite.