KEERTHI DATTA KONANUR RAMANNA, BEng

Visual Odometry for Off-Road Navigation

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Gerald Steinbauer

Institute of Softwaretechnology

Graz, January 2021

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Affidavit

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

Date   Signature

Acknowledgements

I would like to convey my thanks and gratitude to my supervisor Dr. Gerald Steinbauer for his support and time investment for guiding me through my thesis.

I would like to take this opportunity to thank all my colleagues and Tedusar team members at the Autonomous Intelligent Systems lab for their support to my work.

Furthermore, I am grateful for the effort put by Austrian Ministry of Defense who made the data collection experiment possible. Special thanks to Institute of Geodesy of Graz University of Technology for providing ground truth data.

I would like to thank Joanneum Research for providing the dataset collected at the military training grounds in Strass.

Finally, I am very thankful to all my friends and family for their encouragement, support and standing by my side during difficult times.

Abstract

For the autonomous navigation of robots, reliable and efficient state estimation is of utmost importance. In GPS-denied scenarios, cameras are often used as the primary sensors for wide range of applications, such as Visual Odometry, SLAM, 3D reconstruction, obstacle avoidance, to name a few. While these topics are well studied and perform well in structured environments like factory floors or automated driving, there is less work done in off-road scenarios.

In this thesis, three modern feature-based Visual Odometry approaches to estimate robot's ego-motion from stereo cameras are evaluated. The approaches are ORB-SLAM2, RTAB-MAP F2M and F2F, and SOFT VO. The evaluation is performed on three different off-road datasets: the Davon Island navigation dataset, the Seetaler Alps dataset, and the Strass datastet. The Seetaler Alps off-road dataset was collected in a challenging alpine environment using a robot platform equipped with a stereo navigation camera, two stereo hazard cameras and an inertial measurement unit. The navigation camera was designed and built up using two monocular cameras to meet the requirements of the data recording. The dataset will be published for the benefit of off-road navigation research across the robotics community.

The performance of Visual Odometry approaches are compared and assessed based on various error metrics. In the Davon Island dataset, the RTAB-Map F2M approach achieved the most accurate results, whereas, ORB-SLAM2 outperformed other approaches in the Strass dataset. Due to rain, snow, and adverse environmental conditions, all the approaches failed to achieve reasonable localization accuracy in the Seetaler Alps dataset. Quantitative results show that, in the Strass dataset, ORB-SLAM2 achieved significant improvement in trajectory estimation after loop closure and global optimization when compared to pure Visual Odometry.

Contents

List of Figures

1 Introduction

Robots have made their way into human lives in the last couple of decades. With the advancements in sensor technology, the development of new approaches and methods, autonomous mobile robots are used more and more in various applications like automation, logistics, rescue, space, military, and automotive, to name a few.

In the last few years, self-driving car technology has seen tremendous progress; autonomous cars are becoming a possibility in the transportation sector. Industrial robots are working efficiently side-by-side with humans to meet the high demands in volume and variability in the production sector. In field-robotics such as disaster response, mining, or agriculture, unmanned aerial and ground vehicles are essential tools. Autonomous robots impact society by assisting humans in dangerous environments such as radiation and nuclear disaster sites. Mars exploration rovers are continuously exploring and collecting the red planet's scientific data. A key aspect in most of these applications is that the robot will be moving autonomously in a mostly unknown environment.

In order to navigate in an environment autonomously, a robot needs to estimate its position in real-time. Accurate position estimation of the robot is an essential requirement for other vital tasks such as mapping, planning, and navigation. Robot pose estimation, more specifically called robot localization, involves estimating the robot's position and orientation with respect to a coordinate frame of reference. Robot localization for autonomous navigation is a difficult problem, and it is an active area of research. A wide range of sensors is used to tackle this problem. However, the choice of sensor suite mainly depends on the application and its parameters.

1.1 Motivation

This work's motivation originates from the common requirements and demands from various use cases in off-road scenarios. We focus on the two use cases in this thesis: autonomous navigation in a Mars-like environment and autonomous navigation for disaster relief in a military context such as hilly terrain and unpaved roads. Some of the examples of off-road conditions are shown in Figure 1.1.

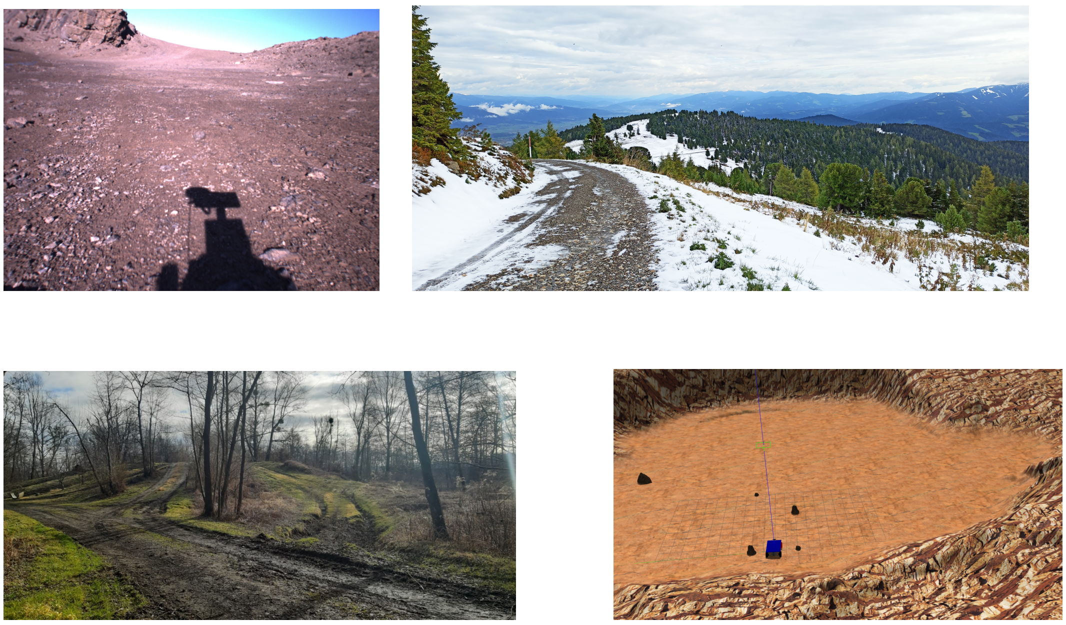


Figure 1.1: These pictures show different off-road use cases focussed in this thesis. Top left is a picture from the Davon Island Mars dataset, top right is taken at Seetaler Alps, bottom left is from the Strass off-road dataset and the right one is from Mars simulation world.

AMADEE-18 mars analog field simulation, which happened in 2018 in Oman (mission location shown in Figure 1.2), was a field research experiment for preparing for future human Mars missions in engineering, planetary surface operations, astrobiology, geophysics/geology, life sciences, and other (Groemer et al., 2020). The rover of Graz University of technology that participated in the mission used a 3D lidar sensor to perform mapping, autonomous navigation, and exploration (Stradner and G. Steinbauer, 2019).

However, lidar sensors are large, heavy, and not suitable for space applications. It is worth mentioning that the rovers sent to the red planet to date use cameras for navigation. This work, therefore, aims at the development of approaches suitable for space applications. The goal is to develop new approaches in space exploration and participate in AMADEE-20 (Gerald Steinbauer et al., 2020).

Figure 1.2: AMADEE-18 mars analog simulation experiments conducted at Oman. Picture credits: Austrian Space Forum (`https://oewf.org/en/portfolio/amadee-18/`).

In a military context, an autonomous vehicle demands the usage of only passive sensors as active sensors emit radiations and thus may interfere with radio services. A vehicle should be able to navigate in an off-road environment to perform various transport activities. PALONA stands for Passive infrastructural vehicle Localization for Navigation is a project funded by FFG Austrian army. The aim of this project is to use passive sensors for off-road autonomous navigation and transport tasks. Therefore, our goal is to develop new approaches for autonomous navigation in the military context.

1.2 Goals and Challenges

A common approach for planetary exploration and military use cases is to use cameras and especially stereo cameras because of their ability to perceive depth information. Moreover, cameras are helpful for mapping and navigation and other tasks like object recognition and scene understanding. The work presented in this thesis aims to develop a software framework that can form the core component of a robot localization for off-road environments using stereo vision, which is called Visual Odometry. While keeping track of its pose, the robot also needs to map the environment as Simultaneous localization and mapping(SLAM) are crucial for an autonomous robot. Although Visual Odometry is part of the most recent vision-based SLAM approaches, a thorough investigation of Visual Odometry is necessary before pairing up with a suitable SLAM approach.

Off-road environments comprise uneven terrain, unstructured tracks, gravel, mud, rocks, and other natural terrains. These are very challenging scenarios for autonomous navigation. Adverse weather conditions like rain and snow pose additional challenges for vision-based navigation. Most existing Visual Odometry approaches are benchmarked against standard KITTI self-driving car data sets (Geiger, Lenz, and Urtasun, 2012) and well-known datasets collected in a structured human-made environment. Although some of these approaches achieve extremely accurate estimation results, there is no guarantee that the results are the same in rather challenging off-road scenarios.

Furthermore, a custom stereo camera system needs to be developed that is configurable and appropriate to off-road use cases. The robot platform should be configured with the stereo setup, and an off-road dataset needs to be collected. Visual Odometry approaches are then evaluated on the collected dataset. A stereo camera's custom configuration poses many engineering challenges to obtain pictures useful for vision-based algorithms.

Some of these challenges are time-synchronization of left and right camera images, camera calibration and rectification, and varying environmental lighting conditions. These challenges need to be solved and well evaluated.

1.3 Contribution

In order to achieve the goals set in this thesis, the contributions have been done in the following areas.

1. Development of a Mars-like simulation environment and sensor plugins to test vision-based navigation algorithms.
2. Implementation and verification of different Stereo Visual Odometry approaches using the Mars-like simulation environment and the Davon Island rover navigation dataset.
3. Integration of Visual Odometry into SLAM systems to create 3D maps of the environment.
4. Development of the infrastructure for a stereo camera suite for the envisioned environments.
5. Collection of off-road datasets for the evaluation of visual-navigation approaches.
6. Evaluation of Visual Odometry approaches using multiple off-road datasets.

1.4 Outline

In Chapter 3, the background of Visual Odometry and SLAM is given. Chapter 4 covers the related research for Visual Odometry and SLAM methods. Chapter 5 introduces different Visual Odometry approaches used and evaluated in this thesis. We explain the concepts of ORB-SLAM2, RTAB-Map F2M, RTAB-Map F2F, and SOFT VO. In Chapter 6, the development of Stereo Vision system is discussed. In Chapter 7, we explain the various off-road datasets used in this thesis. In Chapter 8, the results of the various approaches described in Chapter 5 are presented. Finally, in Chapter 9 we discuss the outcome of this thesis and give details on future improvements.

2 Prerequisites

This chapter introduces all the necessary software frameworks that are used for the implementations reported in this thesis. At first, an introduction to the Robot Operating System (ROS) is given followed by Gazebo simulator and OpenCV.

2.1 Robot Operating System

Robot Operating System(ROS) is an open-source software framework for robotics applications (Quigley et al., n.d.). ROS framework comes with easy to use packages and tools which supports interprocess communication. The communication is based on TCP/IP sockets for transporting message over networks. It supports C++, python and several other programming languages and is thus suitable for a wide variety of uses. ROS is fully open source and comes with a diversity of pack- ages which have been released from a big community. Additional documentation can be found at `http://wiki.ros.org`.

The processes are called nodes, and communication between nodes is handled with predefined messages. The transaction of messages is done over ROS topics within which only one message type is allowed to be exchanged. Nodes can subscribe and publish on several topics at once and communicate asynchronously. In some cases, communication needs to be synchronized. This is achieved using ROS services. A core process called the ROS master manages all the nodes. ROS master tracks all the registered nodes, topics, and services. The ROS master is also responsible for setting up a peer-to-peer connection between appropriate nodes. ROS topic communication is

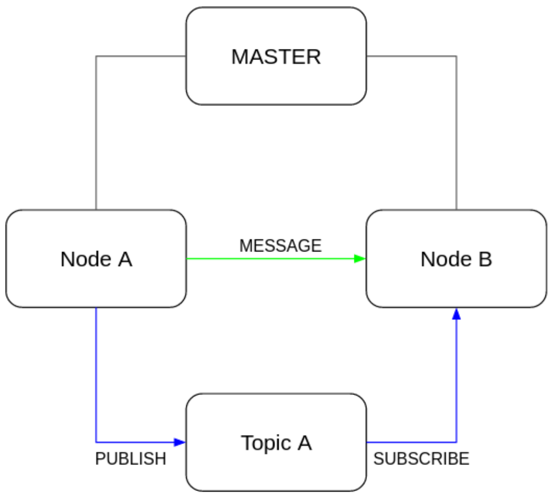


Figure 2.1: The figure illustrates the topic communication concept in ROS. Node A first registers the Topic A to ROS Master by advertising, and Node B subscribes to the Topic A. Messages are then directly transmitted from Node A to B over Topic A. (Adapted from `http://wiki.ros.org`).

illustrated in Figure 2.1. The ROS version used in this thesis is ROS Melodic on Ubuntu 18.04 Linux OS.

2.2 Gazebo

Robot simulation is an essential tool for the Robotics applications. A well-designed simulator helps test algorithms, design robots rapidly, and perform testing in realistic scenarios. Gazebo offers a robust physics engine, high-quality graphics, custom plugins, and a convenient programmatic and graphical interface (Koenig and Howard, 2004). A wide range of sensor data can be simulated with noise, from laser range finders, cameras, contact sensors, force, torque, and more.

The Mars environment was built using Gazebo's SDF model object. SDF model is essentially a collection of links, joints, collision objects, visuals, and plugins. The robot model is loaded to the Mars Gazebo environment, as shown in Figure 2.2. The Robot model is represented in the form of a Unified Robot Description Format (URDF), which is an XML format describing the robot model. The target robot platform used in this thesis is called MERCARATOR which is shown in Figure 2.3. A CAD model of the MERCARATOR robot platform (Halatschek et al., 2020) was constructed, and the model was further separated into links and joints that could be used in a URDF robot description file. URDF file format from ROS is converted to SDF format required by Gazebo to load the model in the gazebo environment. Gazebo plugins give the URDF models greater functionality and can tie in ROS messages and service calls for sensor outputs and control inputs. Existing gazebo Sensor plugins are adapted to the application needs and loaded to the robot platform. Wheels and joints of the robot model are controlled using gazebo ros control hardware interfaces. The hardware interface module allows actuation of joints and links based on the input control command.

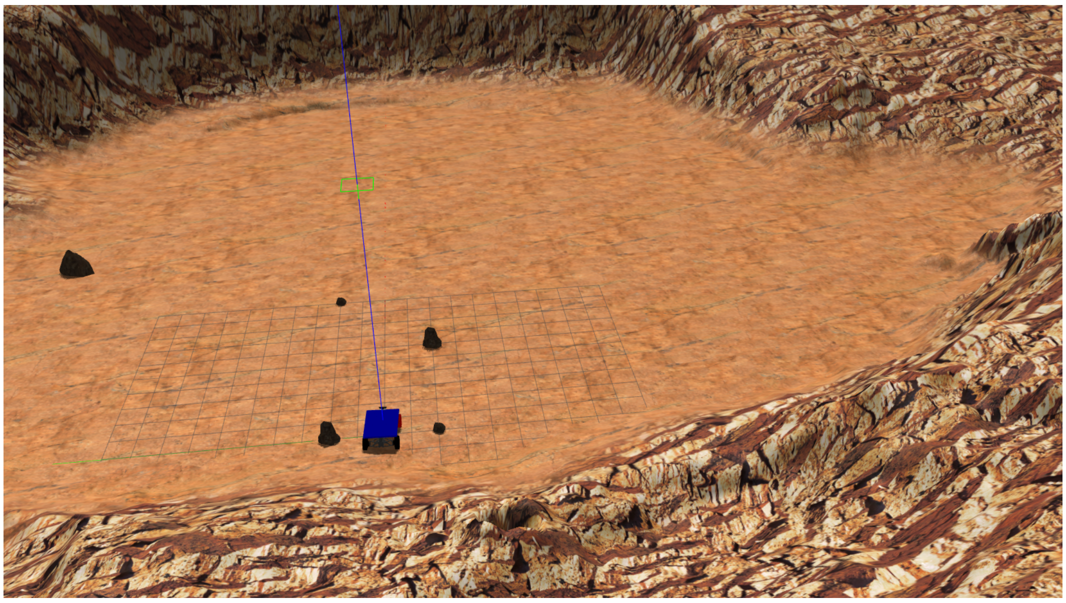


Figure 2.2: Illustration of the Mars simulation environment and the CAD model of Mercarator loaded in Gazebo.

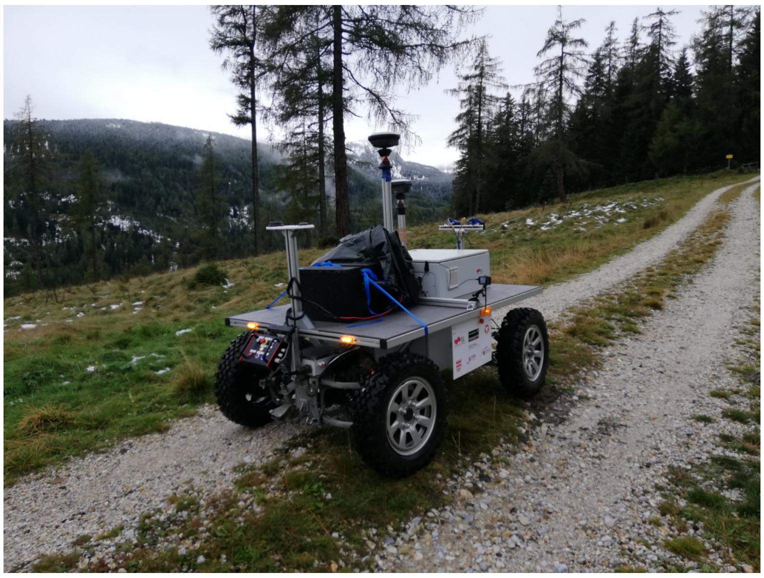


Figure 2.3: The robot platform used in this thesis called MERCARATOR.

2.3 OpenCV

Open Source Computer Vision Library is an open-source computer vision and machine learning software library written in C++. The library includes a comprehensive set of both classic and state of the art computer vision and machine learning algorithms. These algorithms could be used to detect faces, identify objects, track camera movements, produce a 3D point cloud from stereo cameras, to name a few. OpenCV has a modular structure. OpenCV is broadly structured into five major components. The CV component contains the image processing and computer vision algorithms. ML is the machine learning library containing statistical classifiers, data analysis functions, and clustering tools. HighGUI includes I/O routines and functions for storing and loading images and videos, and the CXCore contains the basic data structures and algorithms, XML support, matrix algebra, error handling, and drawing functions. CVCAM component contains camera interfaces for video access.

The CV component is further divided into modules. Some of the CV modules that are used in this thesis are core, imgproc, calib3d, and features2d.

For image filtering, geometrical image transformations, and histograms, the imgproc, an image processing module, is utilized. For stereo calibration, multiview geometry algorithms, 3D reconstruction, and camera pose estimation calib3d module is used. For the feature detection and descriptor matching module, calib3d is used. The basic data structures like multi-dimension array Mat are used from the core functionality of OpenCV. The OpenCV version used in this thesis is OpenCV 4.4.0 (Bradski, 2000)

3 Background

3.1 Coordinate Transformations

There are two main coordinate frames of reference: the World coordinate frame (W) and the camera coordinate frame (C), where the Camera coordinate frame is attached to the stereo camera left optical frame. A known point $P \in \mathbb{R}^3$ in the world seen from the camera can be described in the camera coordinate system by Euclidian transformation consisting of rotation R and translation t given by:

$$ {}^{c}P = {}^{c}R_w {}^{w}P + {}^{c}t_w $$

where, ${}^{c}R_w$ is rotation R from camera frame to the world coordinate frame, ${}^{c}t_w$ is translation t from the camera coordinate frame to the world coordinate frame, and ${}^{w}P$ is the point P in the world coordinate frame.

In the homogeneous coordinate system, rotation and translation together expressed as a transformation matrix $T \in \mathbb{R}^{4x4}$:

$$ {}^{c}T_w = \begin{pmatrix} {}^{c}R_w & {}^{c}t_w \\ 0 & 1 \end{pmatrix} $$

The transformation from the world coordinate to the camera coordinate is the inverse of ${}^{c}T_w$:

$$ {}^{w}T_c = \begin{pmatrix} {}^{c}R_w^T & -{}^{c}R_w^T {}^{c}t_w \\ 0 & 1 \end{pmatrix} $$

The transformation matrix belongs to Special Euclidean Group ***SE(3)***, the Lie group of rigid body transformations in three-dimensional space.

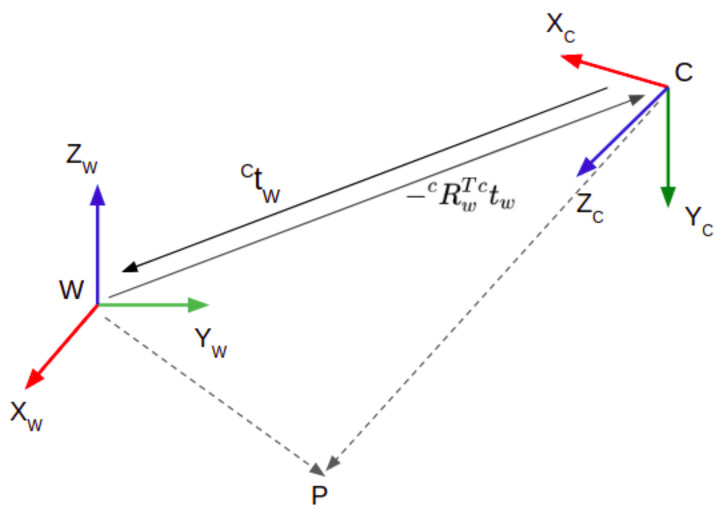


Figure 3.1: The figure illustrates the coordinate transformation between the world frame W and the camera coordinate frame C. In the camera coordinate frame, the point P is seen along the optical axis.

The three-dimensnional rotations are represented using either 3 x 3 rotation matrix, Euler-angles or quaternions. The transformation between two coordinate frames of reference is illustrated in Figure 3.1

3.2 Pinhole Camera Model

The pinhole camera model or perspective projection model describes the relationship between a point in $3D$ coordinates and its projection onto a 2D image plane. This process is called perspective projection. In Figure 3.2, C is the origin of the camera coordinate system called the center of projection. The line Z from the center of projection perpendicular to the image plane is called optical axis or principal axis, and its intersection with the image plane is called the optical center or principal point denoted by c with $c = (c_x, c_y)$. The distance between the center of projection C and the optical center c is called focal length f. A point $P = (X, Y, Z)$ in camera coordinate system is

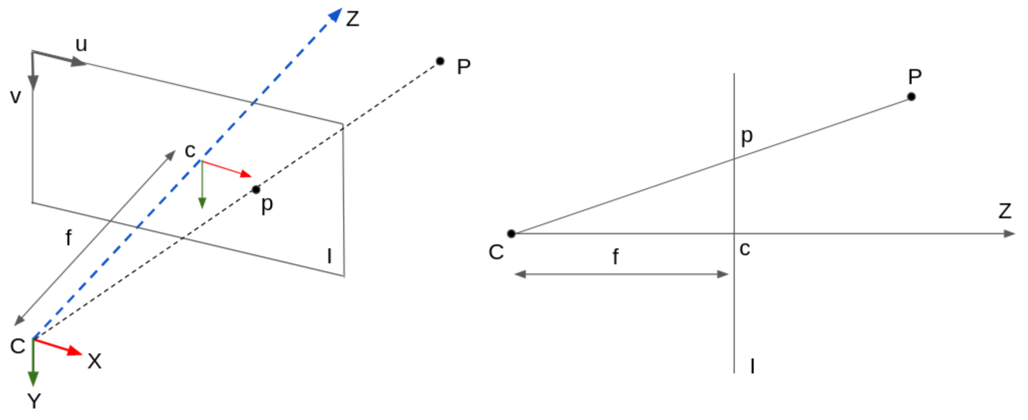


Figure 3.2: Pinhole camera model or Perspective projection model is shown in this diagram. The projection of a $3D$ point P in the camera coordinate to point p on the image plane I is illustrated.

projected $2D$ image plane. Using similar triangles, the projection (x, y) of a $3D$ point P is given by,

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z} \tag{3.1}$$

The projection (x, y) are expressed in image coorinates in *milimetres*. However, the optical measurements are expressed in pixels given by,

$$u = f_x\frac{X}{Z} + c_x, \quad v = f_y\frac{Y}{Z} + c_y \tag{3.2}$$

where, f_x and f_y are focal lengths along x and y respectively expressed in pixels. It could be expressed in matrix notation as,

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{3.3}$$

Using the equation (4.3), $3D$ points in the camera coordinate projected into $2D$ image coordinates. If the points are in the world coordinates, an additional transformation is needed to transform those points to camera coordinates, as discussed in the previous section. Therefore, the projection of points in the world coordinate to the image plane is given in homogeneous form by,

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.4}$$

where R is a $3D$ rotation matrix $R \in \mathbb{R}^{3x3}$ and t is a $3D$ translation vector $t \in \mathbb{R}^3$. f_x, f_y, c_x, c_y are so called intrinsics of the calibration matrix and R, t are extrinsics.

3.3 Epipolar Geometry

Epipolar geometry describes the relationship between the cameras, points in 3D, and the corresponding observations in each camera's image plane. In this thesis, we define relationship between left and right cameras of a stereo setup. As illustrated in the Figure 3.3, the stereo setup involves left and right cameras C and C', observing the same $3D$ point X, whose projection in each of the image planes is located at x and x' respectively. The camera centers C and C' are separated by a rigid body transformation, referred to as baseline. The plane formed by C, C', and the point X is called the epipolar plane. The image points, e and e', where the baseline intersects the two image planes, are known as epipoles or epipolar points. Finally, the lines at the intersection of the epipolar plane and the image planes are called epipolar lines. The epipolar planes intersect the baseline at the respective epipoles in the image plane. It is worth noting that in a scenario where the projected point x is known, and the epipolar line (e', x') is known, then the projection of point X on the right image plane x' must lie on the specific epipolar line. Using this concept, stereo correspondence problem is simplified as point in one view must lie on the epiplar line in the other view. Similarly, if two projection points x and x' are known and they correspond to the same 3D point X, then projection lines intersect at X. This process of computing the 3D point X from two image points is called triangulation.

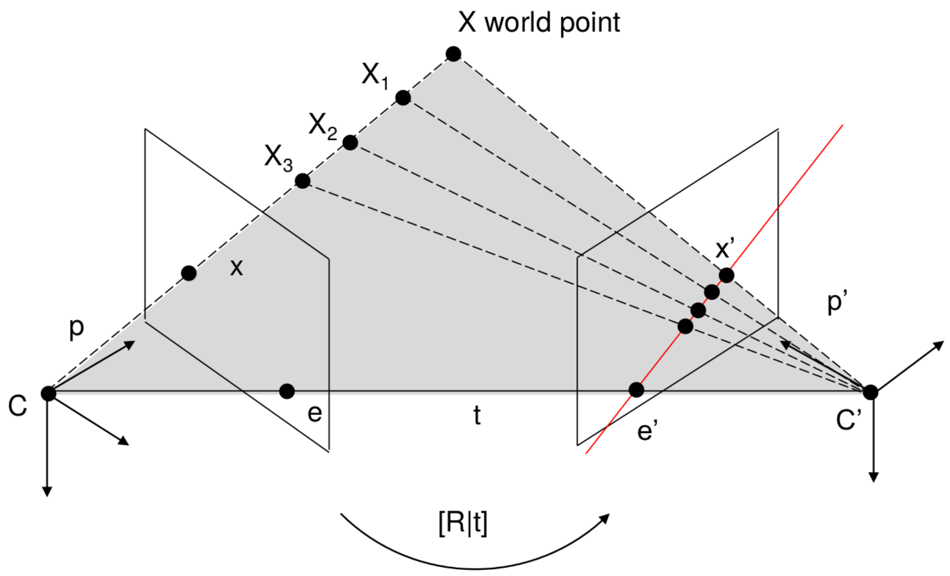


Figure 3.3: This figure is illustrates the concepts of epipolar geometry. Cameras C and C', and the world point X is forms the epipolar plane given in gray region, while the red line is epipolar line in the right image plane which intersects the epipole e' and image coordinate x'.

3.4 Visual Odometry

Given a stream of input images, Visual Odometry aims to estimate the relative camera poses. Formally, Visual Odometry is defined as "the process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras" (Scaramuzza and Fraundorfer, 2011). Robot pose estimation could be achieved using a single camera or a stereo camera. Visual Odometry can be described as a special instance of Structure from Motion, which only focuses on estimating the camera's 3D motion sequentially.

On the other hand, Structure from Motion estimates the 3D motion but does so using unordered image sets and tackles the problem of 3D reconstruction of the environment structure. The final structure and camera poses are typically refined with an offline pose-graph optimization (Kümmerle et al.,

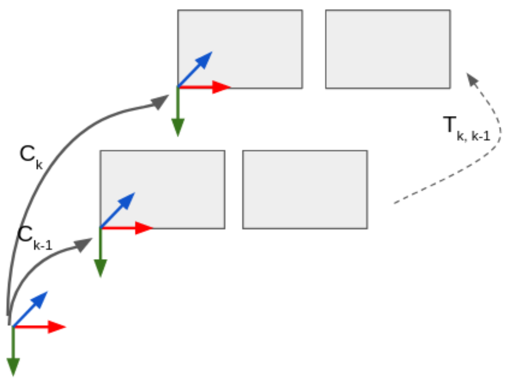


Figure 3.4: Rigid body transformation between subsequent frames $k-1$ and k encapsulates translation and rotation of frame k with respect to frame $k-1$. C_k is a concatenation of transformations $T_{k,k-1}$ and previous estimated transformation C_{k-1}. A schematic representaion of relative transformations is shown. Adapted from (Scaramuzza and Fraundorfer, 2011).

2011) (Madsen, Nielsen, and Tingleff, 2004).

A robot equipped with a monocular or a stereo camera moves in an environment captures images at discrete times $k = 1..K$ The two subsequent images at time $k-1$ and k are related by the rigid body transformation given by,

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

A schematic representation of rigid body transformation between images is shown in Figure 3.4. Given the transformation between two subsequent frames, the camera pose C_k at k is given by,

$$C_k = C_{k-1} T_{k,k-1}$$

The initial camera pose C_0 is typically set as the identity matrix or set based on the known pose of the mobile robot in the world coordinate frame. In most cases, pose estimates using GPS measurements are used as the initial position of the robot.

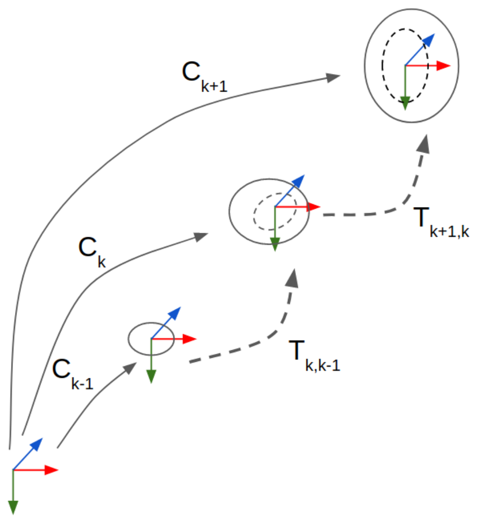


Figure 3.5: Visual Odometry accumulates error by combining uncertainties in the previous pose(solid ellipse) and the last transformation(dashed ellipse). Adapted from (Fraundorfer and Scaramuzza, 2012).

As the pose updates over time, frame to frame motion estimated using Visual Odometry introduces error which accumulates over time. The camera pose uncertainty is always increasing when concatenating transformations. The computed new pose has an uncertainty of the current pose estimate and also depends on the uncertainty of the past transformations. The error generates a drift of the estimated trajectory from the ground truth. This is illustrated in Figure 3.5. Thus, it is crucial to keep the uncertainities of the individual transformations low. The camera pose could be mathermatically defined as the function f of previous pose C_{k-1} and transformation $T_{k,k-1}$. The error coovariance of pose C_k is $\sum_k$ and given by,

$$\Sigma_k = J_{C_{k-1}} \Sigma_{k-1} J^T_{C_{k-1}} + J_{T_{k,k-1}} \Sigma_{k,k-1} J^T_{T_{k,k-1}}$$

where $J_{C_{k-1}}$ and $J_{T_{k,k-1}}$ are the Jacobians of function f with respect to the previous pose C_{k-1} and transformation T_{k-1}.

While the main goal of Visual Odometry is to estimate the relative motion of the camera, visual SLAM aims not only at tracking the camera motion,

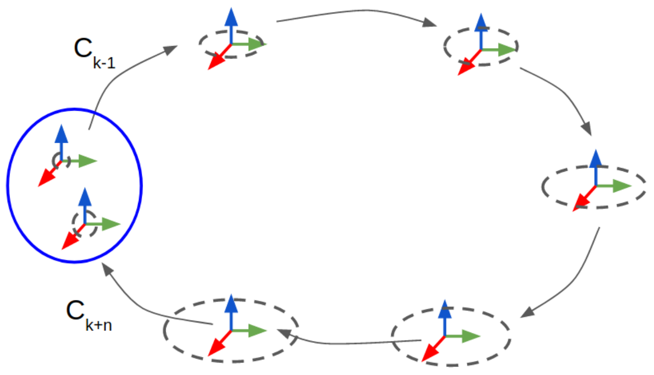


Figure 3.6: Loop closure (highlighted in blue) is performed when the robot re-visits an already visited environment, and the resulting edges are added as constraints to pose-graph optimization.

but also at building a globally consistent map. An advantage of SLAM is the ability to detect loop closures and build globally constistent map. To solve the drift problem, Visual Odometry methods can be extended to SLAM by adding loop closure detection and a global optimization scheme. Loop closure detection is used to reduce the drift in the camera trajectory and the 3D structure. Estimating the location of 3D points in the environment jointly with the corresponding camera pose is a crucial local refinement step to accurately predicting local camera trajectory. It is achieved by minimizing the reprojection error:

$$\operatorname*{argmin}_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$

where p_k^i is the point in image k corresponding to the 3D point X^i and $g(X^i, C_k)$ is the reprojection of the 3D point on the image. Since all 3D features of the environment are not observable from every camera pose, reprojection error minimization is performed only on a window of camera poses. This computation of minimizing the reprojection error and optimizing the 3D points and positions is called Bundle Adjustment. When using the pose-graph representation of the world, each camera pose represents a node, and edges interconnect nodes in a graph. These nodes and edges form spatial constraints. When the robot reobserves places that were already visited, a loop closure is performed as visualized in 3.6. Loop detection allows the robot to correct the accumulated drift and perform a more confident position estimation. Loop closure is generally performed based on visual place recognition using bag of words approach (Gálvez-López and Tardós, 2012).

Visual Odometry techniques are mainly categorized into feature-based and direct methods.

Feature-based methods, also called indirect methods, use only interesting feature points in an image to estimate the camera's motion. Features are the interesting keypoints in the form of corners or blobs. A good feature detector is characterized by robustness, distinctiveness, invariance to geometric and photometric changes, and computational efficiency. The detected feature points are either tracked or matched over subsequent frames. Tracking describes the local search for the same feature in the next images, while matching detects features in both the images and then tries to find the best matches using a similarity measure. The given correspondences are then used to compute relative camera motion.

In contrast to feature-based methods that compute the camera motion based on feature points, direct methods estimate the camera motion from the intensities of the pixels in the image. Direct methods work on the assumption that projections of a point in two frames have the same intensity values in both images, called photo-consistency. However, in reality, photo-consistency is often violated due to factors like sensor noise, different lighting conditions, dynamic objects, etc. This means that the Intensity difference will be non-zero. Therefore, the camera motion is estimated by minimizing the Intensity difference, called photometric error. The error function can then be solved using a numerical optimization technique such as the Gauss-Newton algorithm.

3.5 PnP RANSAC

The motion estimation step involves computing the transformation between two subsequent image frames. Based on the feature representations, there are different relative camera motion estimation approaches. However, in this thesis, all the Visual Odometry approaches use 3D to 2D motion estimation (Fraundorfer and Scaramuzza, 2012). In this technique, the transformation between two frames is computed from 3D to 2D correspondences. The features in the first frame $k-1$ is represented as 3D feature points which could be computed by triangulating the left and right image features. The features in the frame k are 2D reprojections of the 3D points. The problem of finding 3D to 2D motion is known as camera resection or perspective from n points, in short PnP. The solution is found by determining the transformation that minimizes the reprojection error, which is given by,

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \underset{T_{k,k-1}}{\operatorname{argmin}} \sum_i \| P_k^i - \hat{P}_{k-1}^i \|$$

where P_k^i is a feature in image I_k and $\hat{P}_{k-1}^i$ is the reprojection of the 3D point computed from frame $k-1$ into the image I_k according to the transformation $T_{k,k-1}$. If there are outliers in the point correspondences, PnP motion estimation leads to errors in output. To estimate the camera motion accurately, outliers need to be removed. Therefore PnP is used along with Random Sample Consensus (RANSAC) to make the estimated camera pose more robust to outliers (Lepetit, Moreno-Noguer, and Fua, 2009), (Bradski, 2000).

4 Related research

Vision-based robot state estimation has been popular in the last decades. There exists a wide variety of Visual Odometry techniques using monocular cameras, stereo cameras and multi-cameras. A complete tutorial and history of Visual Odometry in the last few decades is described in the Visual Odometry tutorials by Scaramuzza and Fraundorfer (Scaramuzza and Fraundorfer, 2011), (Fraundorfer and Scaramuzza, 2012). The Visual Odometry method was first introduced by (Moravec, 1980) in the 1980s. The implementation was the first to contain fundamentals such as a keypoint detector, a keypoint matcher, and a motion estimator, which forms the basis of the modern Visual Odometry pipeline. However, Visual Odometry was first coined by (Nister, Naroditsky, and Bergen, 2004) in their landmark paper. The authors of (L. Matthies and Shafer, 1987) used a binocular system for detecting and tracking corners. They used error covariance of triangulated features and incorporated them into the motion estimation step. Due to the limited availability of computational power, most of these early methods were executed offline. With the increase in processing power, Visual Odometry is performed online nowadays.

Visual Odometry is being used in the Mars rovers since three decades. Recently, authors of (Lambert et al., 2012) have used sun sensor and inclinometer along with Visual Odometry. The estimates of sensors together are significantly better than Visual Odometry alone. A detailed information of Visual Odometry in Mars is discussed in (Maimone, Cheng, and Larry Matthies, 2007). A very popular dataset called the Davon Island navigation dataset (Furgale et al., 2012) is published for the research on Mars exploration. The Davon Island dataset offers unique planetary terrain suitable for Mars exploration research. In (Grimes and LeCun, 2009), Visual Odometry is used along with wheel odometry in off-road conditions to achieve efficient off-road localization. Recently, (Gonzalez et al., 2012) have have applied Visual Odometry in off-road environment. In their work, estimates of the location of mobile robots operating in the off-road conditions are determined using Visual Odometry and visual compass.

Extensions to Visual Odometry are SLAM approaches that include global optimization and place recognition in some variants. An overview of Visual Odometry and SLAM for autonomous mobile robots is provided in (Yousif, Bab-Hadiashar, and Hoseinnezhad, 2015). It is shown in the recent work presented in (Strasdat, Montiel, and Davison, 2012) that keyframe-based methods which used global optimization techniques like bundle adjustment perform better than the probabilistic approaches like Extended Kalman filter (Paz et al., 2008) and particle filter based SLAM (Wu et al., 2008). One of the popular feature-based SLAM approaches is Parallel Tracking and Mapping (PTAM) (Klein and Murray, 2007). PTAM is a widely used robust monocular SLAM method which runs in real-time. PTAM was one of the first SLAM systems to use threads for tracking and mapping to enable parallel processing. The tracking thread extracts FAST features (Rosten, Porter, and Drummond, 2010) to estimate motion while the mapping thread optimizes the camera poses and feature points by minimizing the reprojection error and updating the map. The stereo variant of PTAM is Stereo-PTAM (SPTAM), which solves the monocular bootstrapping problem and allows to compute scale without any prior information (Pire et al., 2017).

More recently, ORB-SLAM2 (Mur-Artal and Tardós, 2017) has been proposed as the stereo variant of ORB-SLAM (Mur-Artal, Montiel, and Tardos, 2015) monocular SLAM system. ORB-SLAM2 tracks sparse ORB features and distributes work between multiple threads for tracking, mapping, and loop closing similar to PTAM. Another feature-based Graph SLAM approach is RTAB-Map (Labbé and Michaud, 2019) which supports custom configuration of feature detectors, matchers, and various processing modules in visual odometry pipeline. RTAB-Map creates a dense 3D map of the environment, whereas SPTAM and ORB-SLAM2 create sparse 3D features points. Recently, SOFT SLAM, a feature based pose-graph SLAM approach was proposed by (Cvišić et al., 2018). The stereo odometry algorithm relies on feature tracking(SOFT) for pose estimation. Similar to ORB-SLAM2 Soft SLAM runs on odometry and mapping threads.

In contrast to sparse feature-based methods, direct methods exploit all the pixels in the image for pose estimation under the assumption of photometric consistency. Direct methods tend to be more accurate, and recent research has shown that direct methods achieve better results in texture-less environments than feature-based methods (Engel, Stückler, and Cremers, 2015). However, direct methods are computationally demanding and changes in image brightness can affect results as the brightness constancy assumption is violated and tracking may fail. Also, there are numerous implementations of feature-based approaches for different environments and decades of research in feature-based methods makes it an attractive method for this thesis work.

In this decade, deep learning-based techniques have been applied for many computer vision tasks. In the last couple of years, deep learning-based Visual Odometry methods have gained popularity. Although deep learning methods do not achieve the state of the art accuracy, the results have improved a lot in the recent years. One of the early works that use deep learning is DeepVO (Wang et al., 2017). In DeepVO, a deep Recurrent Convolutional Neural Networks (RCNN) framework is built that takes a monocular image sequence, and returns pose. The network is mainly composed of a CNN-based feature extractor and RNN-based sequential modeling. Another deep learning-based approach called DPC-Net (Peretroukhin and Kelly, 2017) introduced deep pose corrections (DPC) uses a deep neural network to predict corrections through a supervised training method driven by ground truth pose information. In self-supervised DPC-Net (Wagstaff, Peretroukhin, and Kelly, 2020), the authors improve upon their previous work by replacing the supervised pose loss with a self-supervised photometric reconstruction loss that eliminates the necessity of ground truth pose labels, which is impractical to obtain. In this approach, they rely on the output of classical Visual Odometry estimate to produce a large prior and use a deep neural network to learn a smaller correction. This forms the correction step to the predicted output of a classical VO pipeline. Recently, Deep learning has been applied to SLAM in the off-road context (Yang et al., 2020). In off-road environments, SLAM encounters problems such as direct sunlight, leaf occlusion, rough roads, sensor failure, sparsity of stably trackable texture. Authors in (Yang et al., 2020), have proposed a panoramic vision SLAM method based on multi-camera collaboration, which utilizes panoramic vision and stereo perception to improve the localization precision in off-road environments.

In this thesis, only feature-based methods are evaluated. RTAB-Map is utilized as it supports the configuration of different combinations of feature detectors, matchers, and provides ROS integration. A possibility to create rich and dense 3D maps by RTAB-Map is also an essential feature for our work. On the other hand, ORB-SLAM2 is also evaluated as it is one of the most used and well-known SLAM implementations for monocular, RGBD, and stereo cameras. ORB-SLAM2 also achieves the best results compared to SPTAM in many open-source datasets. When compared with ORB-SLAM2, SOFT SLAM achieves better accuracy in the KITTI (Geiger, Lenz, and Urtasun, 2012) dataset. Therefore, SOFT-based VO approach is also evaluated along with ORB-SLAM2 and Rtabmap.

5 Evaluated Approaches

In the previous chapters, the Visual Odometry concept was introduced and related research was discussed. In this chapter, different Visual Odometry approaches will be discussed in more details. The approaches are: ORB-SLAM2, RTAB-Map Frame 2 Map (F2M), RTAB-Map Frame to Frame (F2F), and SOFT VO and these approaches are evaluated in Chapter 8.

5.1 ORB-SLAM2

The general processing pipeline for ORB-SLAM2 consists of three main components which run parallel: Tracking, Local Mapping, and Loop Closing (Mur-Artal and Tardós, 2017). The camera is localized every frame in the tracking thread by finding feature matches to the local map and minimizes the reprojection error by performing motion-only bundle adjustment. In the local mapping thread, key frames are inserted and the local map is optimized using local bundle adjustment. In loop closing thread, large loops are detected, and accumulated drift is corrected by running Levenberg–Marquardt pose-graph optimization method. A full bundle adjustment is also performed after loop detection, optimizing all key points and frames, resulting in optimal structure and motion solution. However, only tracking and local mapping are discussed in the further subsections. In Figure 5.1, functional blocks of ORB-SLAM2 is illustrated.

5.1.1 Tracking

The key steps in tracking algorithm are the following ones:

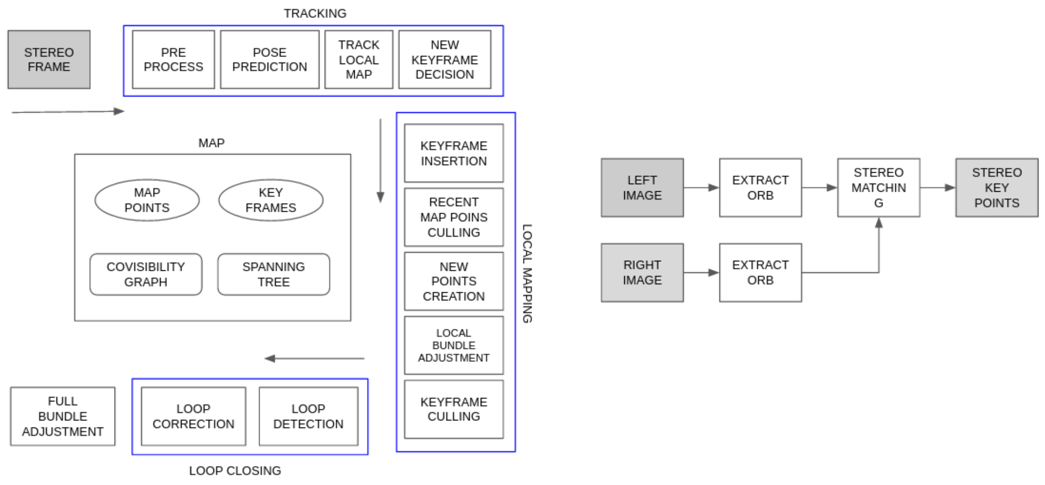


Figure 5.1: Left picture shows the functional blocks of ORB-SLAM2. The right picture illustrates the step by step process involved in keypoint extraction from rectified images. Adapted from (Mur-Artal and Tardós, 2017).

1. ORB descriptors are extracted from both rectified images. Stereo matching is performed by searching every left ORB descriptors in the right image on the same epipolar line.
2. It is important to mention that the stereo keypoints are defined by three coordinates $x_x = (u_l, v_l, u_r)$, where u_l, v_l are coordinates of left image and u_r is the right horizontal coordinate. A stereo keypoint triangulation is performed when the keypoint is close. Far keypoints are excluded because the scale and translation are not reliable even though the rotation information is useful.
3. The camera pose is computed using 3D to 2D PnP motion estimation algorithm and RANSAC is used to remove outliers.
4. Camera pose is optimized by performing motion-only bundle adjustment. In this optimization scheme, camera orientation R and position t are optimized by minimizing the reprojection error between matched 3D points X^i in the world coordinates and keypoints x^i in the stereo image, with $i \in A$, the set of all matches,

$$\{R, t\} = \underset{R,t}{\operatorname{argmin}} \sum_{i \in A} \rho \left(\left\| x^i - \pi \left(R X^i + t \right) \right\|_{\Sigma}^{2} \right)$$

where ρ is the Huber cost function used to down-weight the strong outlier, Σ is the covariance matrix associated to the scale of the keypoint, and π is the projection function. The projection function π is defined as follows:

$$\pi\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x \frac{X}{Y} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Y} + c_x \end{bmatrix}$$

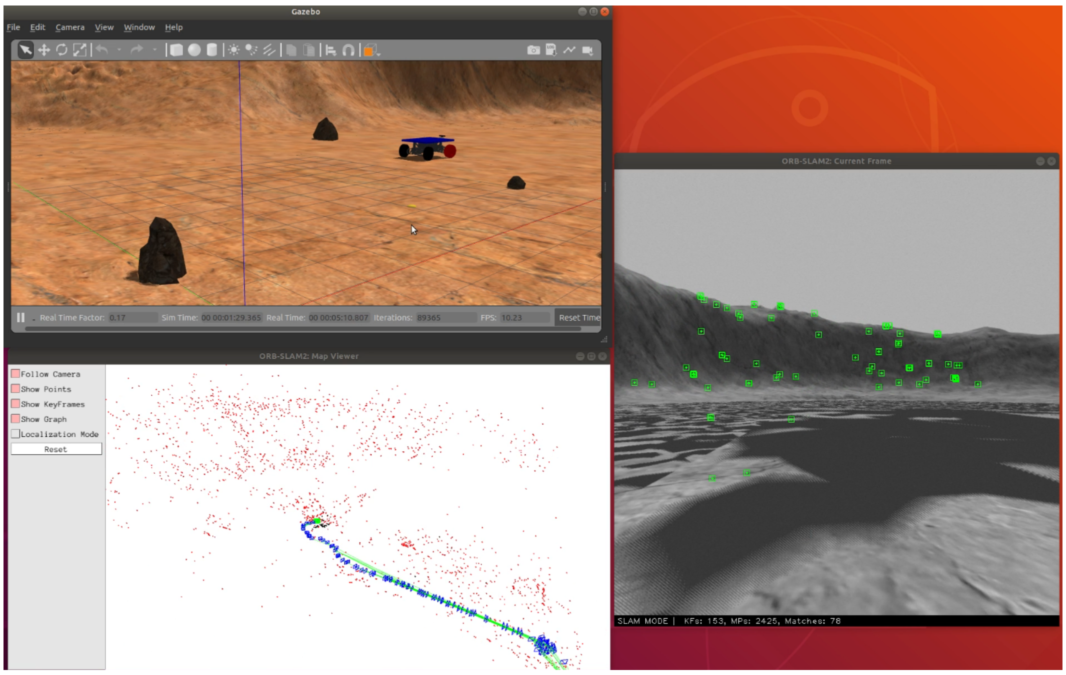
where (f_x, f_y) is focal length, (c_x, c_y) is the principal point, and b is the baseline of the stereo camera.

5.1.2 Local Mapping

A keyframe K_i is nothing but a camera pose, a rigid body transformation that transforms points from the world to the camera coordinate system. A covisibility graph is a weighted graph that contains keyframes and edges. An edge connects keyframes if they share observations of the same map points. When a new keyframe K_i is inserted, the covisibility graph is updated by adding a new node for K_i. It is linked to the keyframe, which shares most point observations, and when a keyframe is removed, the system updates the links affected by that keyframe.

A new keyframe is inserted when the associated feature points of the current frame falls less than 90% points than the latest keyframe. In addition, if the number of tracked points drops below a certain throshold a new keyframe is inserted. New map points are created by triangulating features from connected keyframes in the covisibility graph. In addition to creating map points, local bundle adjustment also performed in the local mapping thread. Local bundle adjustment optimizes the currently processed keyframe, the keyframes connected to it in the covisibility graph, and the map points seen by those keyframes.

In localization mode, the camera is continuously localized by making use of visual odometry matches and matches to map points. Visual odometry matches are the matches between features in the current frame and the 3D points created in the previous frame. These matches make localization



Real Time Factor: 0.17 Sim Time: 00 00:01:29.365 Real Time: 00 00:05:10.807 Iterations: 89365 FPS: 10.23

SLAM MODE | KFs: 153, MPs: 2425, Matches: 78

Figure 5.2: This figure illustrates the results of ORB-SLAM2 in the Mars simulation environment. The top left picture is the Gazebo Mars world where the robot is exploring. The bottom left picture shows the keyframes, local map and trajectory. The right image shows the computed ORB features based on the stereo images captured by the camera on the simulated rover.

robust in a well-mapped region, but drift accumulates in an unmapped area. The results of ORB-SLAM2 in the Mars simulation environment is shown in 5.2.

5.2 RTAB-Map

RTAB-Map is a graph-based SLAM approach integrated with the ROS framework (Labbé and Michaud, 2019). RTAB-Map's SLAM pipeline can be used with any external Visual Odometry. It is even possible to use RTAB-Map with other sensors like laser scanner or RGB-D camera. RTAB-Map's map is represented by graph of nodes and links. A link connects two nodes, by a rigid transformation. Links are categorized into Neighbor, Loop Closure, and Proximity link. The Short-Term Memory (STM) module creates a node memorizing the odometry pose, sensor's raw data, and additional data like features to retain certain information useful for loop closure and place recognition. These nodes and links are used as constraints for graph optimization. RTAB-Map also publishes OctoMap, Point Cloud, and 2D Occupancy Grid maps as ROS messages. The functional architecture of RTAB-Map is shown in Figure 5.3. A memory management approach is

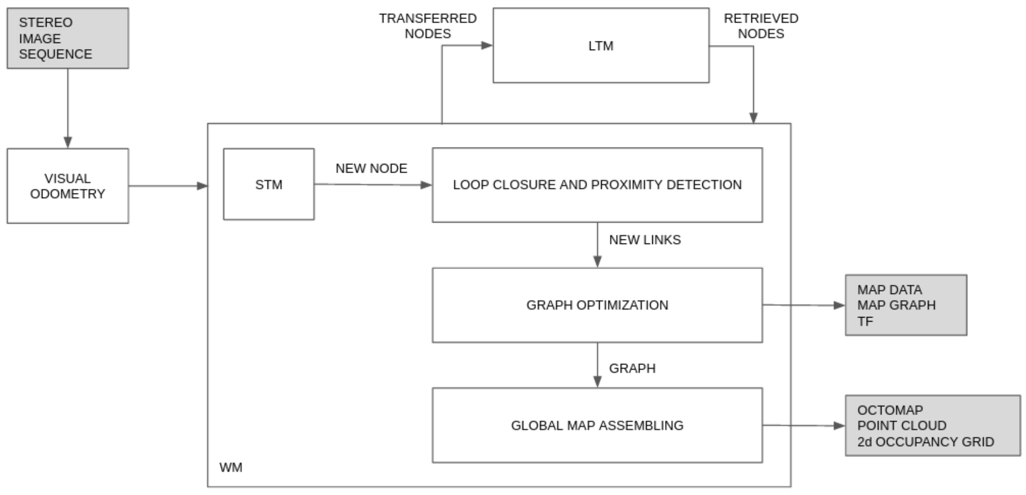


Figure 5.3: Architecture of RTAB-Map. Adapted from (Labbé and Michaud, 2019)

used to limit the size of the graph in order to achieve long-term online SLAM. Memory management is crucial for loop closure and proximity detection as the graph grows. If the graph is very large, graph optimization processing time becomes a problem without such memory management.

RTAB-Map's memory management is divided into Working Memory (WM) and Long Term Memory (LTM). After reaching a certain maximum number of nodes, nodes are transferred from WM to LTM. When a loop closure is detected with a node in the WM, neighbor nodes can be brought back from LTM to WM for more loop closure and proximity detections. A robot moving in a previously visited area can localize itself using the past locations and extend the current assembled map. Global consistency is achieved by realizing that a previously mapped area has been re-visited (loop closure), and this information is used to reduce the drift in the trajectory estimates.

5.2.1 Stereo Visual Odometry

RTAB-Map has two standard stereo visual odometry approaches; Frame to Map (F2M) and Frame to Frame (F2F). The algorithm is explained in the following steps:

1. Feature keypoints are extracted from incoming stereo rectified images. A variety of feature detectors like FAST, SIFT, GFTT, SURF are integrated in RTAB-Map.
2. Stereo correspondences are computed using optical flow (Lucas-Kanade method).
3. In the case of Frame to Map implementation, feature matching is performed by nearest neighbor search with nearest neighbor distance ratio test. The feature map, which consists of 3D features with descriptors, are matched against the extracted features. Whereas, in the Frame to Frame approach, the current 3D feature points are matched against the previous keyframe.
4. In Frame to Map, PnP RANSAC is used to compute the 3D-2D transformation between features in the current frame and the feature map. In Frame to Frame approach, transformation is computed between features in the current frame to the Keyframe's feature points.
5. In Frame to Map, the transformation is refined using local bundle adjustment on all the keyframes in the feature map. In Frame to Frame, the features in the last keyframe is optimized.

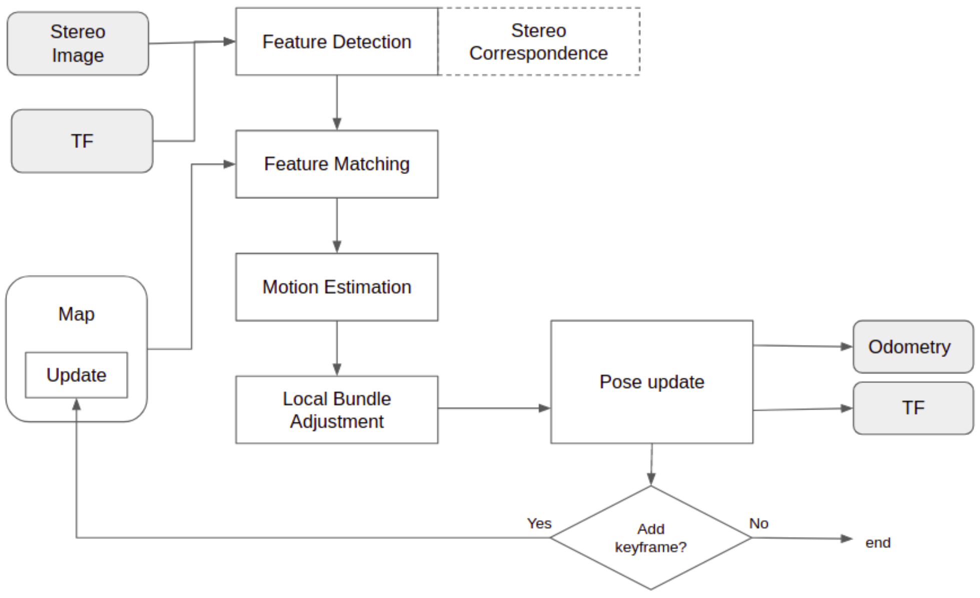


Figure 5.4: The functional blocks of RTAB-Map's stereo Visual Odometry approach

6. An error covariance is computed using mean absolute deviation between 3D feature correspondences. The Odometry message is published on the ROS topic.
7. A new keyframe is inserted when the number of inliers during motion estimation falls below a threshold value. The feature map is updated by eliminating the old features that are not matched with the current frame.

The standard RTAB-Map approach performs motion prediction before doing 3D-2D motion estimation. It uses a motion model to predict the Keyframe or feature map features in the current frame based on the transformation from the previous frame. However, this step predicts wrong estimates in Mars environment datasets. Hence motion prediction was removed from the Visual Odometry pipeline. The results of RTAB-Map in the Mars simulation environment is shown in Figure 5.5.

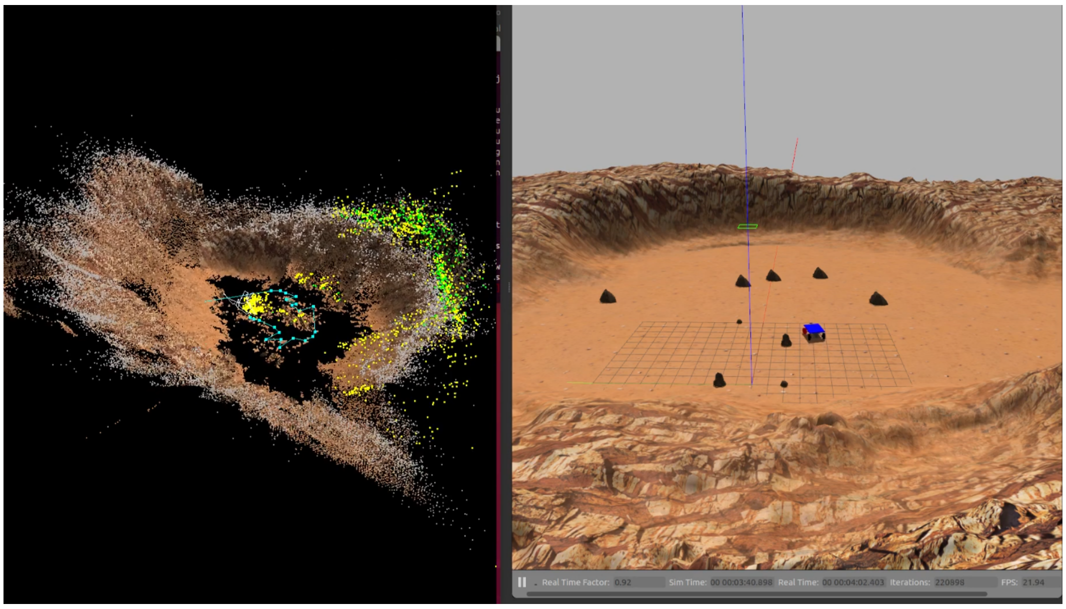


Figure 5.5: This figure illustrates the results of RTAB-Map in the Mars simulation environment. The right picture is the Gazebo mars world where the robot is exploring. The left picture shows the robot trajectory, and the 3D dense point cloud map.

5.3 SOFT VO

Soft Visual Odometry is a feature-based visual odometry approach implemented based on Soft SLAM (Cvišić et al., 2018). Similar to other feature-based methods, in SOFT VO, features are extracted and matched over subsequent frames. However, the feature matching step is performed by running feature search between consecutive stereo frames in a circular fashion, using optical flow. The motion of the vehicle is estimated by 3D-2D frame to frame estimation by minimizing reprojection error. The functional block diagram of Visual Odometry pipeline is shown in Figure 5.6.

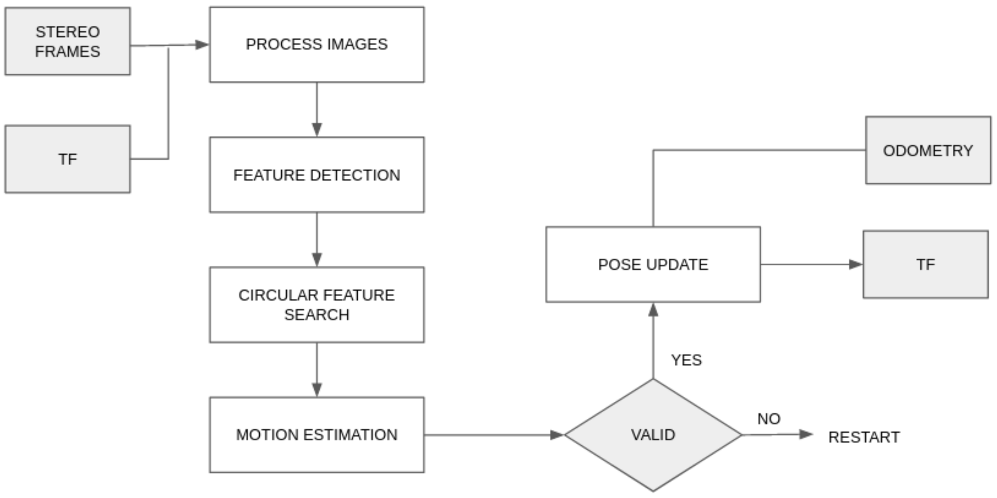


Figure 5.6: Functional block diagram of Soft Visual Odometry. Adapted from (Cvišić et al., 2018)

The Visual Odometry implementation of SOFT VO is illustrated in the following steps:

1. The incoming stream of subsequent stereo frames $k-1$ and k are processed first. In this step, the images are converted to gray scale and stereo-rectified.

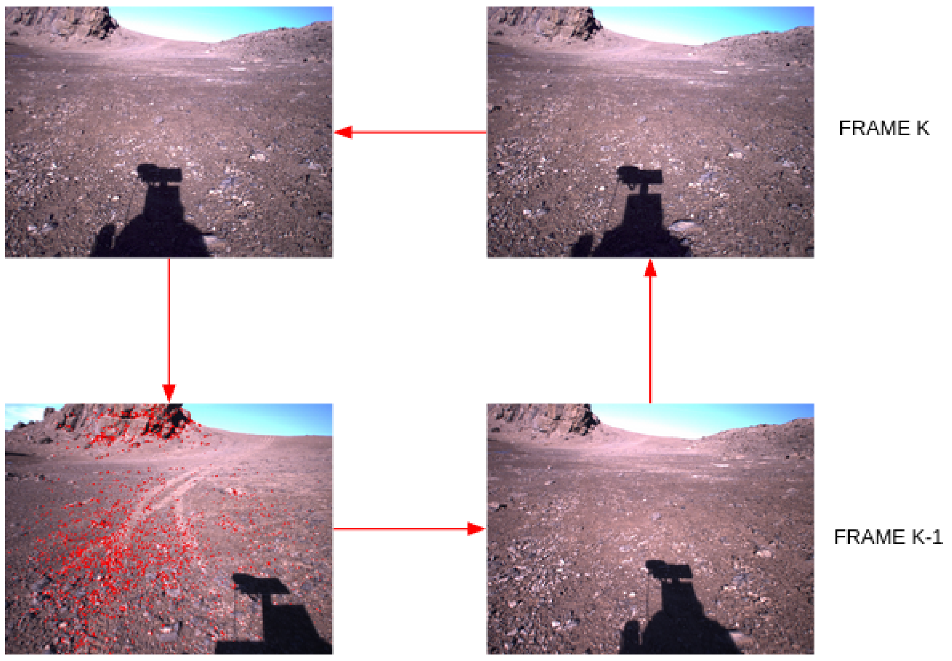


Figure 5.7: This picture illustrates circular feature search. The computed features (Harris corners) in the left image is searched over all the images in a circular fashion.

2. In feature detection step, Good Features To Track (GFTT) features, also called Shi-Thomasi corners (Jianbo Shi and Tomasi, 1994) are extracted from left image of frame $k-1$. The corner features are detected using the OpenCV implementation of Shi-Tomasi method. All corners below pre-configured quality level are rejected. The remaining corners based on quality threshold are sorted in descending order. The strongest corners are retained and the nearby corners to the strongest corners are thrown away.
3. Using Lukas-Kanade optical flow method, the features detected in the left image of frame $k-1$ is searched in the other three images in a circular way as shown in the Figure 5.7. The circular search algorithm takes a feature from one frame and finds its best match in an another frame following a sequential order from left image of frame $k-1 \rightarrow$

right image of frame $k-1 \rightarrow$ right image of frame $k \rightarrow$ left image of frame $k \rightarrow$ left image of frame $k-1$. The features that are successfully tracked through the entire sequence are considered as stable and thus saved as high-quality features for the next step and the rest of the features are ommitted from the feature space.

4. Additionally, a feature elimination technique is employed to remove invalid features. A threshold distance between features is pre-configured. Invalid features are identified based on the threshold distance, and eliminated from the feature space.
5. With the valid features in the feature space, features in frame $k-1$ are triangulated to get the 3D feature points.
6. PnP RANSAC motion estimation algorithm is used to compute the 3D-2D transformation between features in the frame k and the triangulated 3D feature points in frame $k-1$. RANSAC ensures that the outliers are rejected before computing the frame to frame transformation.
7. Pose update is performed by integrating the frame to frame transformation.

6 Stereo Camera hardware setup

A stereo vision setup was built up using two monocular cameras and an adjustable stereo rig. Two Basler acA1920-40uc monocular cameras are used for this setup. The acA1920-40uc is a USB 3.0 camera, uses a Sony IMX249 CMOS sensor, and has a frame rate of 41 frames per second at a 2.3 MP resolution. LM8HC 1″ 8mm 5MP C-Mount lenses are used, which have a focal length of 8mm. The setup is shown in Figure 6.1.

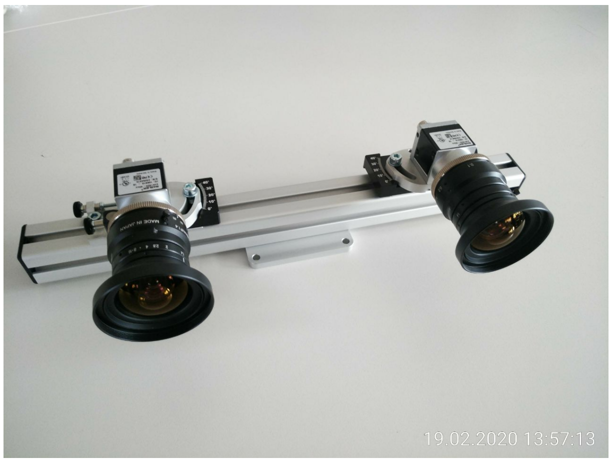


Figure 6.1: The stereo camera setup consists of two Basler acA1920-40uc cameras and an adjustable stereo rig.

The software architecture of the stereo camera system is in Figure 6.2. The monocular camera driver support is provided by the Pylon package in ROS. The driver was adjusted and configured to the stereo setup. Cameras are hardware synchronized by an external hardware timer to trigger images at 20 frames per second. Images are also time-synchronized to publish images at the same timestamp on ROS left and right image topics. Stereo camera calibration is performed, and exposure is controlled automatically

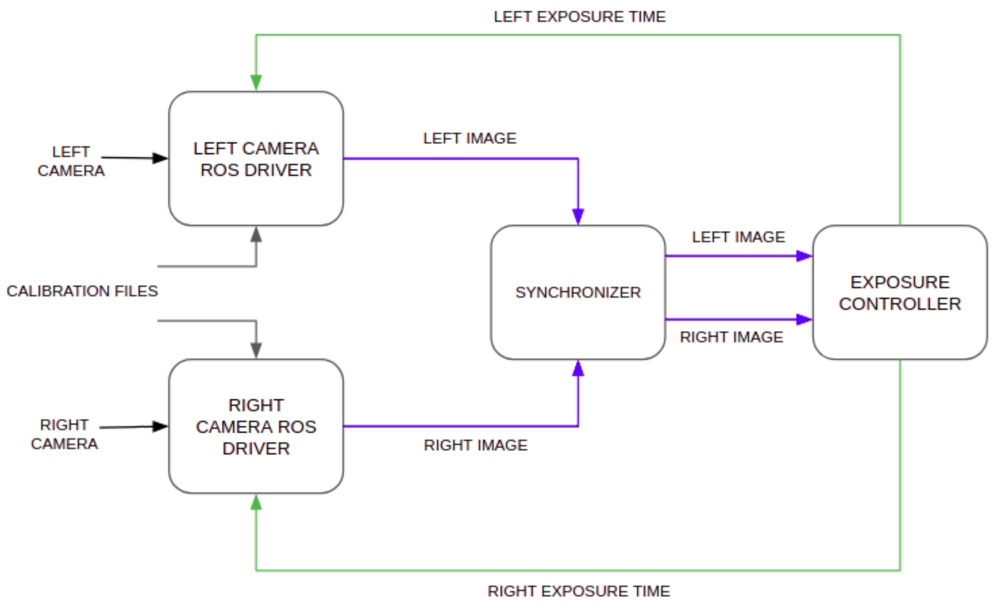


Figure 6.2: This overview diagram illustrates the software architecture of the stereo camera system. Blue line represents ROS topic and green line represents ROS service

based on the brightness information extracted from a stream of images. The stereo setup is mainly used as the navigation camera, and hence it is called NavCam. The different blocks are discussed in the next sections.

6.1 Synchronization

Hardware synchronization is configured using the Mbed LPC1768 32-bit ARM Cortex microcontroller. The communication between the controller and Rover computer is established using a serial bus, universal asynchronous receiver-transmitter. Synchronization to the cameras is done by pulse width modulation(PWM) with isolated fast switching opto-couplers. Even though hardware synchronization is performed, the images are not published at the same time to the ROS topics. It is crucial to have same timestamp for left and right images to perform various stereo operations. A timestamp synchronizer is implemented in ROS, which uses "approximate_time" synchronization policy of "message_filters" package.

6.2 Camera Calibration

Camera calibration is the process of estimating the intrinsic and extrinsic parameters of the camera system. The extrinsic parameters represent a rigid body transformation from the world coordinate system to the camera coordinate system. The intrinsic parameters represent a transformation from camera coordinates to image coordinates. In other words, intrinsic parameters describe the camera's internal characteristics, such as focal length, skew, distortion, and image center.

Camera matrix is given by,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where focal length f_x, f_y, principal point c_x, c_y form intrinsic parameters of the camera.

Camera lenses introduce radial and tangential distortions on images. These distortions need to be corrected before applying geometric algorithms on images.

Radial distortions are corrected using a polynomial funcion that relates with the distance to image center. Mathematical model for radial distortion is given by,

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Mathematical model for tangential distortion is given by,

$$x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)]$$

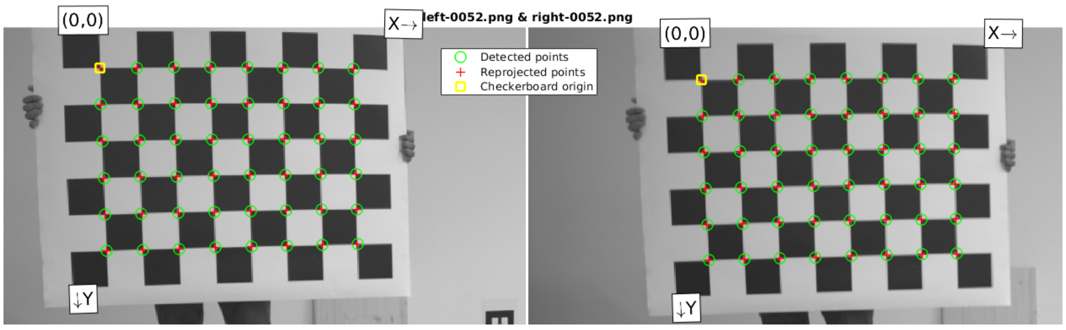
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2x]$$

where x and y are distorted image coordinates, $x_{corrected}$ and $y_{corrected}$ are undistored normalized image coordinates and k_1, k_2, p_1, p_2, k_3 are distortion coefficients.

In the case of NavCam, the camera matrix is determined for both left and right cameras. Stereo calibration procedure will essentially find out extrinsic parameters (rotation R and translation t) between left and right cameras. Stereo rectification process is used to rectify the images from left and right cameras onto a common image plane, so that the corresponding points lie on horizontal scan lines. In other words, the rectified images satisfy the following two properties:

- All epipolar lines are parallel to the horizontal axis (X-axis).
- The corresponding points have identical vertical coordinates.

A checkerboard of size 8x6 is used as a calibration target. Stereo calibration and rectification is performed in ROS using camera calibration package from ros-perception module. The camera calibration package is a ROS wrapper of OpenCV stereo camera calibration implementation. Additionally, the results are cross verified in Matlab stereo calibration tool. The correctness of the calibration is verified by reprojecting the 3D points on image corner points as shown in Figure 6.3. The images with high reprojection error are removed and recalibrated to improve accuracy of the calibration. The overall mean reprojection error is reduced to 0.24 after recalibration as seen in Figure 6.4. After stereo rectification, corresponding points are found along horizontal scan lines as shown in Figure 6.5



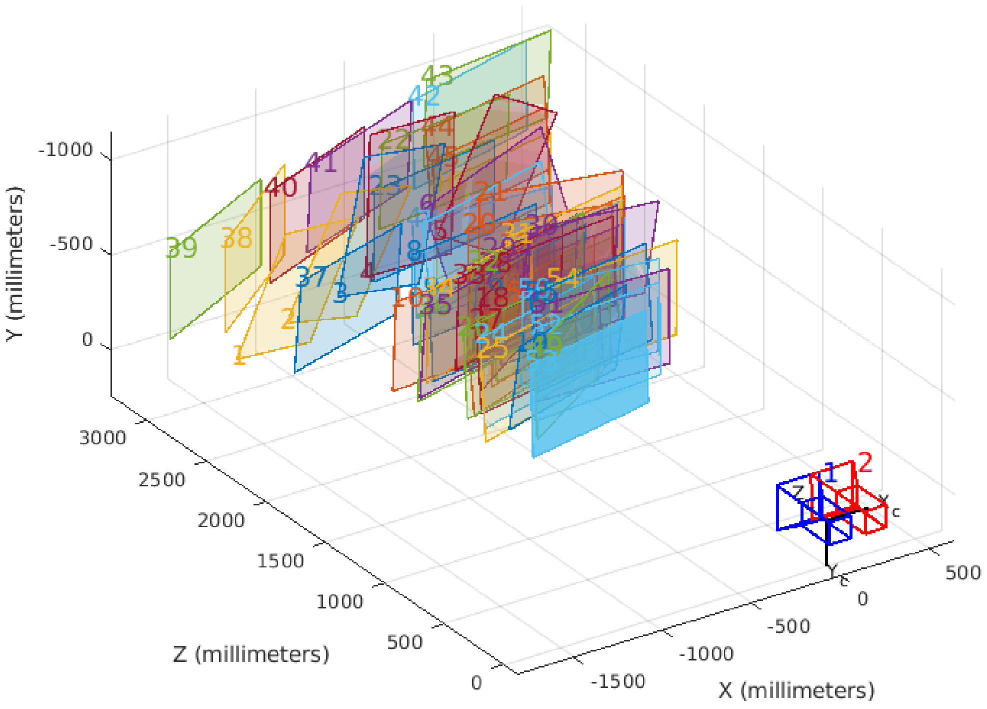


Figure 6.3: The top picture shows the detected corner points using FAST features and the corresponding reprojected points. The bottom plot shows the captured checkerboard images used for calibration.

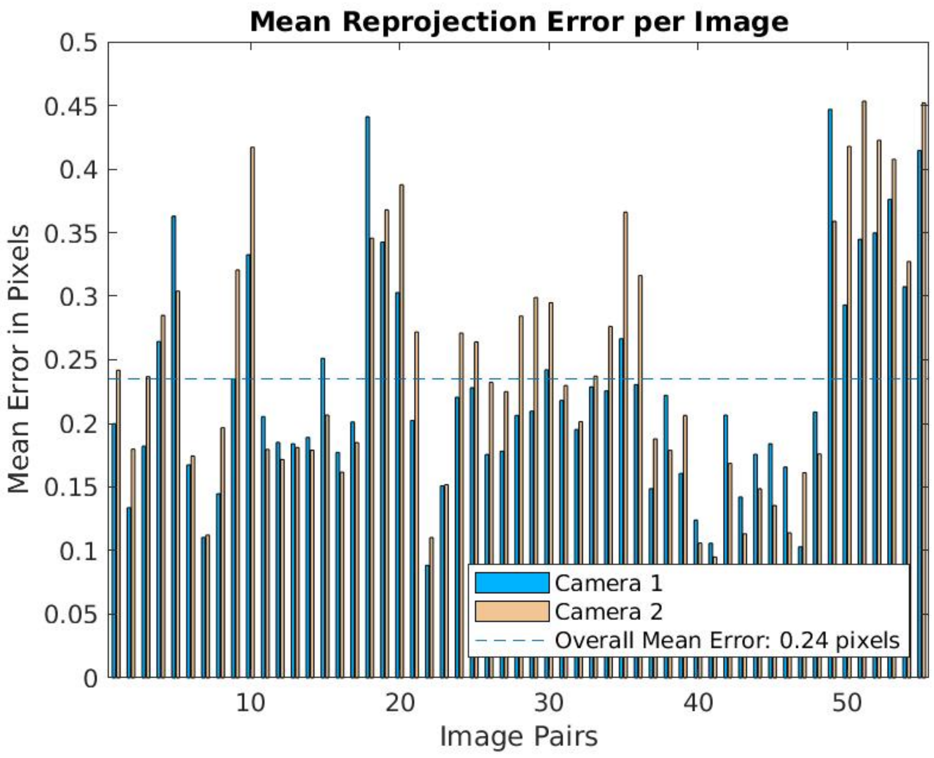


Figure 6.4: Mean reprojection error per image is shown in plot. This representation is helpful to remove the images that introduce large reprojection errors.

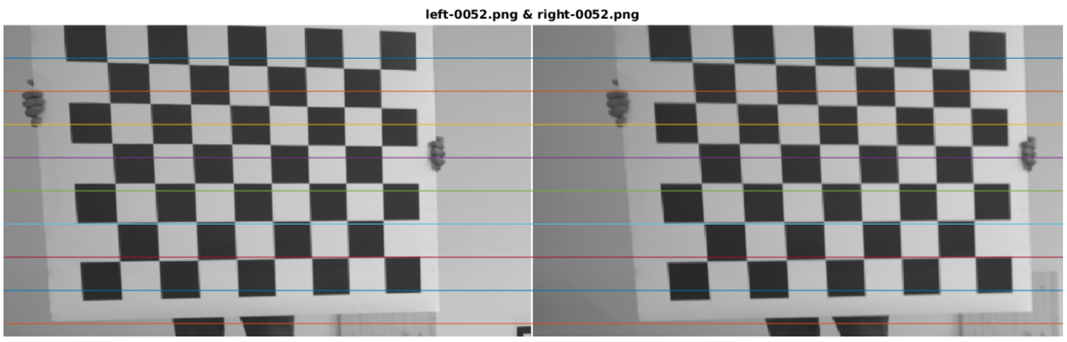


Figure 6.5: Left and right images are rectified after performing stereo rectification. It can be seen that, corresponding feature points lie on the same epipolar line.

6.3 Exposure Controller

The NavCam comprises of left and right cameras. Left and right cameras are hardware synchronized to trigger images at the same time. Our cameras do not offer auto-exposure feature when hardware trigger is used. In an uncontrolled environment, Vision algorithms often fail as the scene radiance is unpredictable due to large illumination changes over space and time (Shim, Lee, and Kweon, 2014). In such cases, a better control mechanism is necessary to monitor and control the exposure of the camera. Pylon ROS driver allows adjustment of exposure time by driver's ROS service "set_exposure_time".

Following the approach proposed in (Shim, Lee, and Kweon, 2014), the image histograms are used to compute the optimal exposure time. An image histogram is a graphical representation of the brightness of an image. Histogram bars correspond to luminance value in the image ranging from 0 to 255. Image histograms indicate the exposure of the image, the nature of lighting conditions, and whether the image is underexposed or overexposed.

In Figure 6.6 under and over exposed Images as well as the corresponding brightness histograms are shown. The histogram represents the response of the camera sensor. The left regions represent dark tone and right regions represent bright tone. An underexposed image will be leaning towards left while an overexposed image will be leaning to the right in the histogram. Image details reduce when the image is underexposed or overexposed. Hence, to get maximum details of an image, it needs to appear in the middle region of the histogram. To interpret histograms, the mean sample value (MSV) is computed, which determines the balance of the tonal distribution in the image. MSV is given by,

$$\mu = \frac{\sum_{i=0}^{4}(i+1) * x_i}{\sum_{i=0}^{4} x_i} \tag{6.1}$$

where i is the range of brightness values in the histogram (0 to 4), x_i is the sum of the values in region i. Using MSV, the image is correctly exposed when μ is close to 2.5 $(\mu_{optimal})$ (Nourani-Vatani and Roberts,

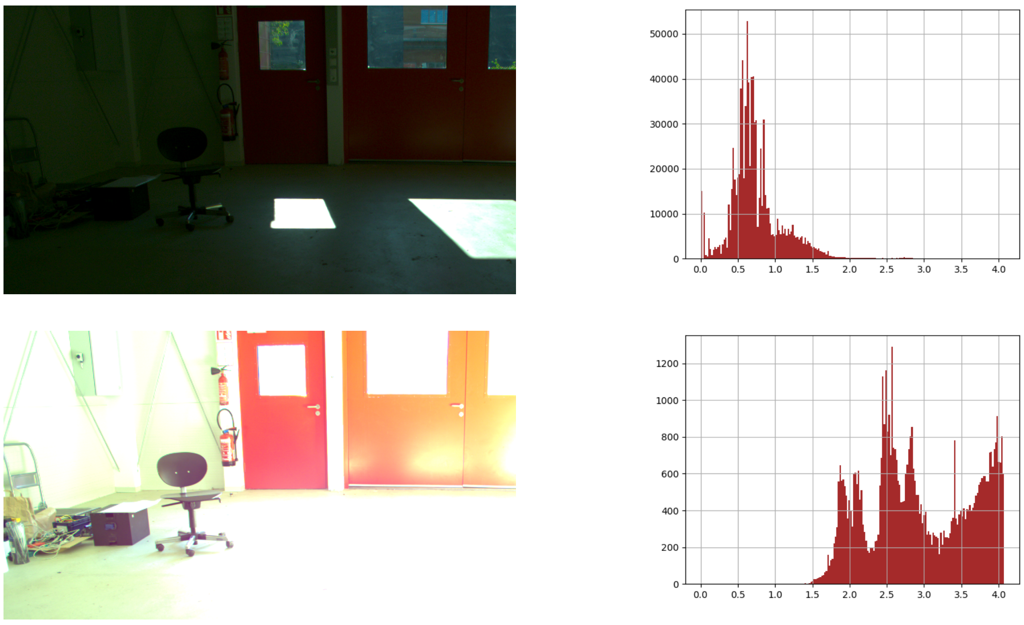


Figure 6.6: When a camera is under-exposed, an image captured from the camera appears dark, while it appears very bright if the camera is over-exposed. In either situation, the resulting image loses a lot of information. An image, if it is under-exposed or over-exposed, could be inferred from the brightness histogram

2007). To maintain optimal exposure, an optimal amount of light has to pass through the lens. There are two main factors that affect the amount of light entering the lens: aperture and exposure time or shutter speed. Aperture is a diaphragm that is made up of a set of blades that open or close based on a specific setting referred to as f-stop. Lower f-stops have a larger diaphragm opening that allows more light through the lens, while higher f-stops have a smaller diaphragm and allow less light through the lens. On the other hand, exposure time defines the time that a camera's shutter remains open. The longer the camera shutter is open, the more light is entered into the camera, whereas less light enters if the exposure time is shorter. Hence, exposure time is a critical parameter to maintain a high frame rate without compromising the quality of the image.

The exposure value specifies the relationship between aperture size and exposure time. It is given by,

$$EV = \log_2(\frac{N^2}{t}) \tag{6.2}$$

where N is the ratio of f-stop to aperture size and t is exposure time or shutter speed. Solving it for Exposure time t leads to,

$$t = \frac{N^2}{2^{EV}}$$

An optimal exposure value is computed which should give MSV close to $\mu_{optimal}$, optimal exposure value is given by:

$$EV_{optimal} = EV + \log_2(\mu) - \log_2(\mu_{optimal}) \tag{6.3}$$

$$t_{optimal} = \frac{N^2}{2^{EV_{optimal}}} \tag{6.4}$$

Finally, an optimal value of the exposure time t is computed by subsituting and solving $EV_{optimal}$. In essence, a relationship between brightness and exposure time is estiblished as following:

$$t_{optimal} = \frac{N^2}{2^{EV + \log_2(\mu) - \log_2(\mu_{optimal})}}$$

$$t_{optimal} = \frac{N^2}{2^{EV(\frac{\mu}{\mu_{optimal}})}}$$

$$t_{optimal} = t(\frac{\mu_{optimal}}{\mu}) \tag{6.5}$$

A proportional integrator (PI) controller is implemented to set the optimal exposure time on the ROS camera service. A PI controller is a control mechanism that continuously calculates an error value $e(t)$ as the difference between desired value and the measured actual value. The error is minimized using a control law with gain parameters. The overall control law is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

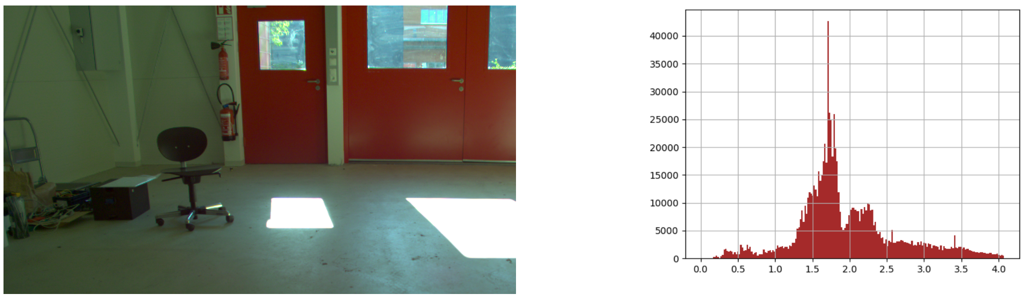


Figure 6.7: The Figure shows the example of an image and its brightness histogram when the controller has minimized the error.

where K_p and K_i are proportional and integral gains of the controller. $e(t)$ is the error in actual brightness and desired brightness. The MSV or μ for each image is computed and the proportional and integrator gains are tuned to minimize the error between μ and $\mu_{optimal}$. The exposure time is preconfigured with an initial value suitable for daylight conditions. As the camera requires less exposure time to capture an image in a daylight condition, daylight configuration ensures no drop in frame rate when the camera system is started. An example of image and its histogram controlled by a PI controller is shown in Figure 6.7.

7 Off-Road Datasets

In this chapter, the different off-road datasets used for the evaluation are described. First, the details of the Davon Island navigation dataset is given, followed by the description of the data collection procedure performed at the military training ground in the Seetaler Alps, Austria, and the Strass off-road dataset collected by Joanneum Research at the military ground in Strass, Austria.

7.1 The Davon Island rover navigation dataset

The Devon Island Rover Navigation Datase[1] is a collection of data gathered at Mars analogue site on Davon Island, Canada. A pushcart outfitted with rover engineering sensors is used as the data collection platform. This dataset contains rover traverse data, including sensory information such as Stereo Images from Bumblebee XB3, data from a Sinclair Interplanetary SS-411 digital sun sensor, and readings from a HMR-3000 Inclinometer and ground truth positions (Furgale et al., 2012). For ground truth positions a pair of Magellan ProMark3 GPS units were used to get post-processed differential GPS data. The rover and sensor setup used to collect data is shown in Figure 7.1.

The rover traversal data of 10 kilometers is partitioned into 23 sequences. The traverse passes through areas exhibiting unique, vegetation-free, planetary-analogue terrain, boulder fields, and sandy flats. The resulting images, coupled with data from the other sensors and precise ground truth information, make a relevant dataset for planetary explortion research. The images are captured at 1 frame per second at 1280x960 resolution. An estimate

[1] http://asrl.utias.utoronto.ca/datasets/devon-island-rover-navigation/

of the platform's orientation in the topocentric frame is provided for the first image of each of the sequences, allowing easy comparison of motion estimate results to the ground truth. The location of data collection in the Davon Island is shown in Figure 7.2.

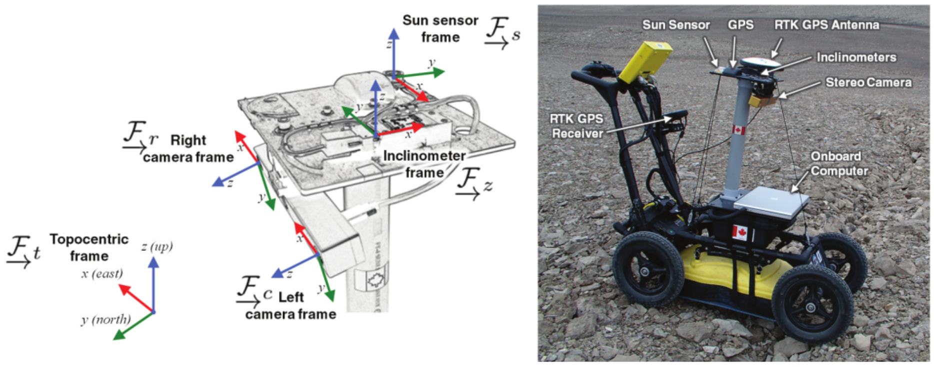

Sun sensor frame
$\underset{\rightarrow}{\mathcal{F}}_s$
Right camera frame
$\underset{\rightarrow}{\mathcal{F}}_r$
Inclinometer frame
$\underset{\rightarrow}{\mathcal{F}}_z$
Topocentric frame
$\underset{\rightarrow}{\mathcal{F}}_t$
z (up)
x (east)
y (north)
Left camera frame
$\underset{\rightarrow}{\mathcal{F}}_c$
Sun Sensor
GPS
RTK GPS Antenna
Inclinometers
Stereo Camera
RTK GPS Receiver
Onboard Computer

Figure 7.1: The platform and sensor setup used to collect the Davon Island navigation dataset (picture credits: (Furgale et al., 2012)).

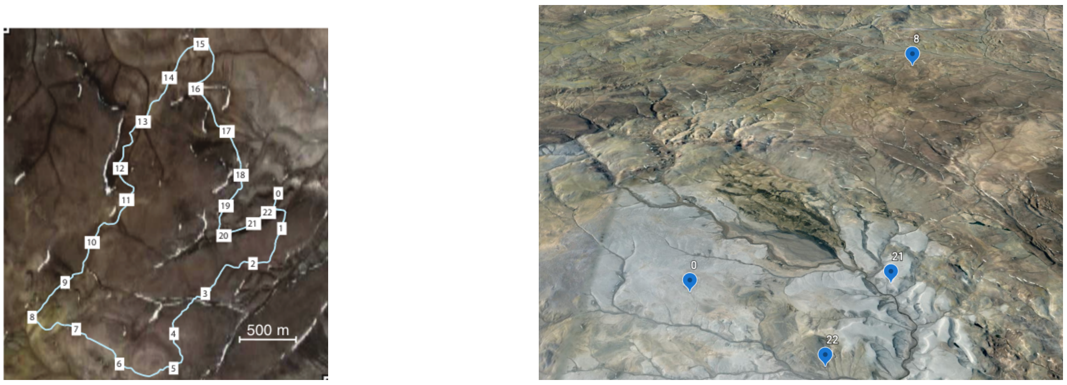


Figure 7.2: Pictures show the locations of the data collected at the Davon Island, Canada. The left picture shows the locations of all the sequences and the right one shows the sequences which are used in this thesis (credits: (Furgale et al., 2012), Google Earth).

7.2 Seetaler Alps Off-road dataset

An off-road dataset was collected at the military training grounds of Seetaler alps, Austria. Alpine terrain presents unique off-road conditions suitable for research in off-road navigation. We used MERCARATOR robot platform for the Seetaler alps off-road data collection. The MERCARATOR is a universal off-road rescue platform developed at the Autonomous Intelligent Systems lab from the Institute of Software Technology, TU Graz. The robot platform supports advanced 4-wheel steering system by wire and moderate off-road capabilities (e.g., slope, terrain). The platform is equipped with a sensor suite and a powerful computer for data processing that suits off-road autonomous navigation. Safety aspects are crucial for any autonomous robot. Therefore, several emergency stop mechanisms like remote emergency switch, multiple emergency switches on the platform, and battery brownout management are implemented (Halatschek et al., 2020). The robot platform and the sensor setup used to collect data is shown in Figure 7.3

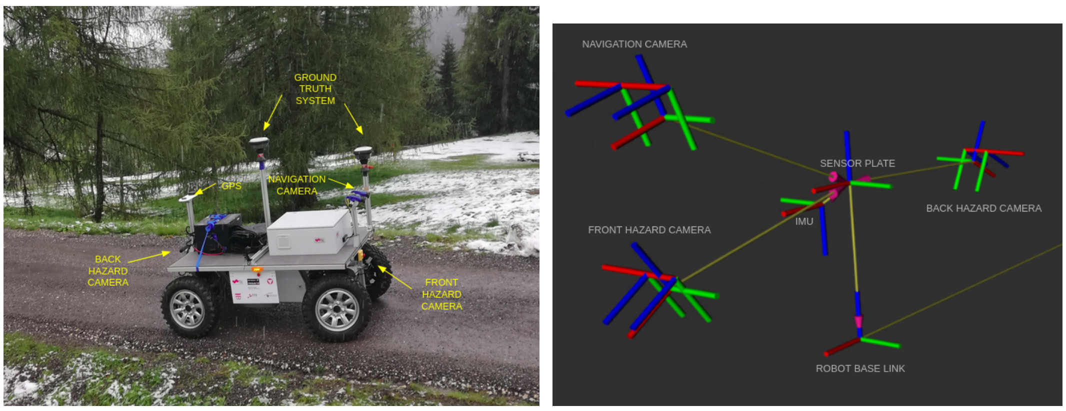


Figure 7.3: The picture on left is Mercarator, the off-road robot platform used for data collection. Different sensors are configured on Mercarator. The right picture shows the sensor frames from ROS tf tree.

The following sensors are mounted on the robot platform:

- Stereo Cameras:
  This comprises a navigation camera (NavCam) as the main stereo setup and two Hazard cameras for hazard detection and obstacle avoidance.

NavCam is the synchronized, calibrated, and exposure-controlled system discussed in the previous chapter. Zed2 stereo cameras are used as HazCams. The HazCams are calibrated as well as stereo rectified and use synchronization and exposure settings provided by the manufacturer. Color images from NavCam are captured at 20 frames per second at the resolution of 1280x720, whereas gray scale images from HazCams are captured at 10 frames per second at 640x480.

- Inertial Measurement Unit (IMU):
  xsens mti-g IMU sensors are used for integrated GPS-IMU measurements. Settings are adjusted to standard Automotive configuration. IMU measurements were also collected from ZED2 cameras. Additional data like magnetometer and barometer data were also collected from Zed2 cameras.

- Ground truth measurement system: Ground truth is obtained using a high-precision GNSS/IMU measurement system. A Trimble Zephyr 3 Geodetic Antenna in Combination with a Trimble NetRS Receiver has been used as a Base Station for the RTK processing. The coordinates in the national coordinate system (MGI) of the Trimble Base station have been computed with GNSS raw data provided by the Austrian Government Service APOS. The position of the platform is recorded in high precision of approximately 2cm accuracy.

Sensors are calibrated using hand-eye calibration and verified with the URDF and CAD models. Data is collected and stored in ROS bag files. Data is collected in multiple runs or sequences in different locations as shown in Figure 7.4. Four different areas were selected for the data collection and multiple sequences of data were collected. In this thesis, four sequences sequence 06, 07, 09, 11 are used. In the first run, rover was brought back to the starting point so that vision-based place recognition algorithms could be tested for loop closure and trajectory correction. The traverse passes through areas exhibiting unique off-road terrain that consists of grass, dirt road, and snow. In the second and third runs, traverse passes through hilly terrain consists of snow. The last run is a relatively easy traversal with a straight and flat terrain.

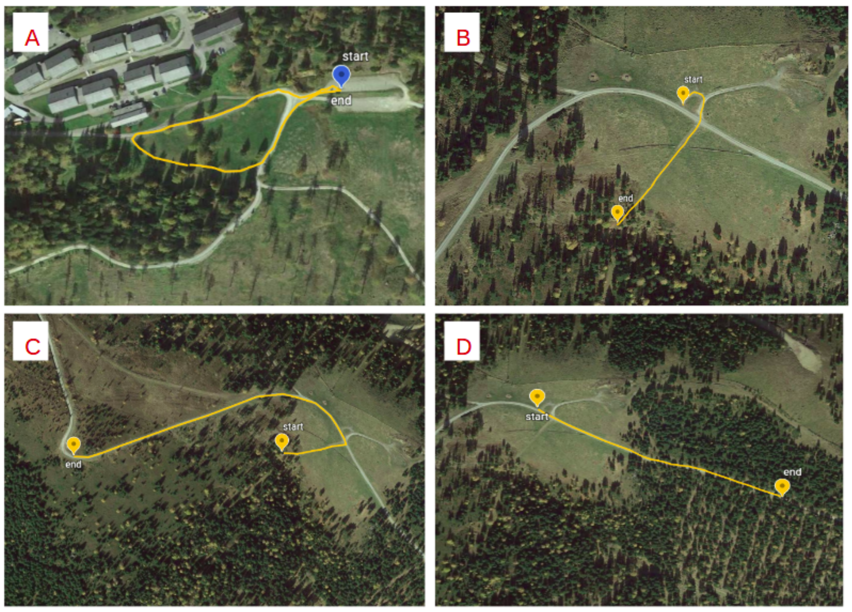


Figure 7.4: Pictures showing start location, end location and trajectories of four sequences of off-road data collected at the military training area in Seetaler Alps (credits: Google Earth).

7.3 Strass Off-road dataset

The Strass off-road dataset is a collection of data gathered at the military training ground in Strass, Austria. Strass off-road dataset is collected by Joanneum Research as a part of project PALONA funded by the Austrian Research Promotion Agency (FFG). A car equipped with sensors is used as the data collection platform. This dataset contains the traverse data of the car which includes sensory information such as images from 8 Sekonix 2MP cameras connected to an Nvidea Drive PX2 computer, measurements from a Litef ISA 100C IMU system, and GNSS data using the Novatel OEM7 GNSS receiver. The ground truth positions are computed using iMAR iNAT-FSLG02 ground truth system which uses measurements from the IMU and the GNSS receiver. The color images are captured at 30 frames per second at a resolution of 1920x1208. The front two cameras are used as a stereo setup with a baseline of 1.2 metres. The platform and the sensor setup used to collect the data is shown in Figure 7.5.

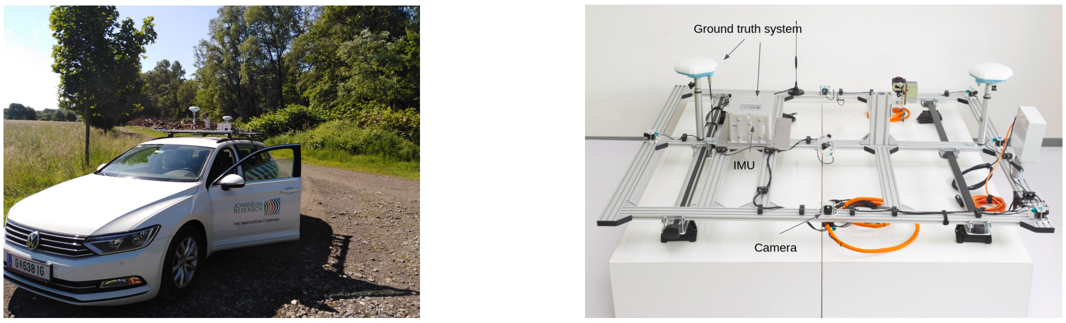


Figure 7.5: The picture on the left shows the platform used to collect the Strass dataset and on the top is the sensor plate. A general sensor setup on top plate is shown in the right picture (image credits: JOANNEUM RESEARCH).

The location and the trajectory of the data collected in Strass is shown in Figure 7.6. The traversal data consists of multiple sequences. In this thesis, a long sequence of 3400 meters consisting of 19387 stereo images is used. However, to assess Visual Odometry algorithms, a part of the long sequence of 1400 meters length is taken from start point to the loop closure point. The full-length sequence is used for comparing the estimates of Visual Odometry and SLAM. The traverse passes through flat terrain exhibiting unique dirt road surrounded by trees. The resulting images, coupled with data from the IMU and precise ground truth information, make a relevant dataset for vision-based off-road robotics research.

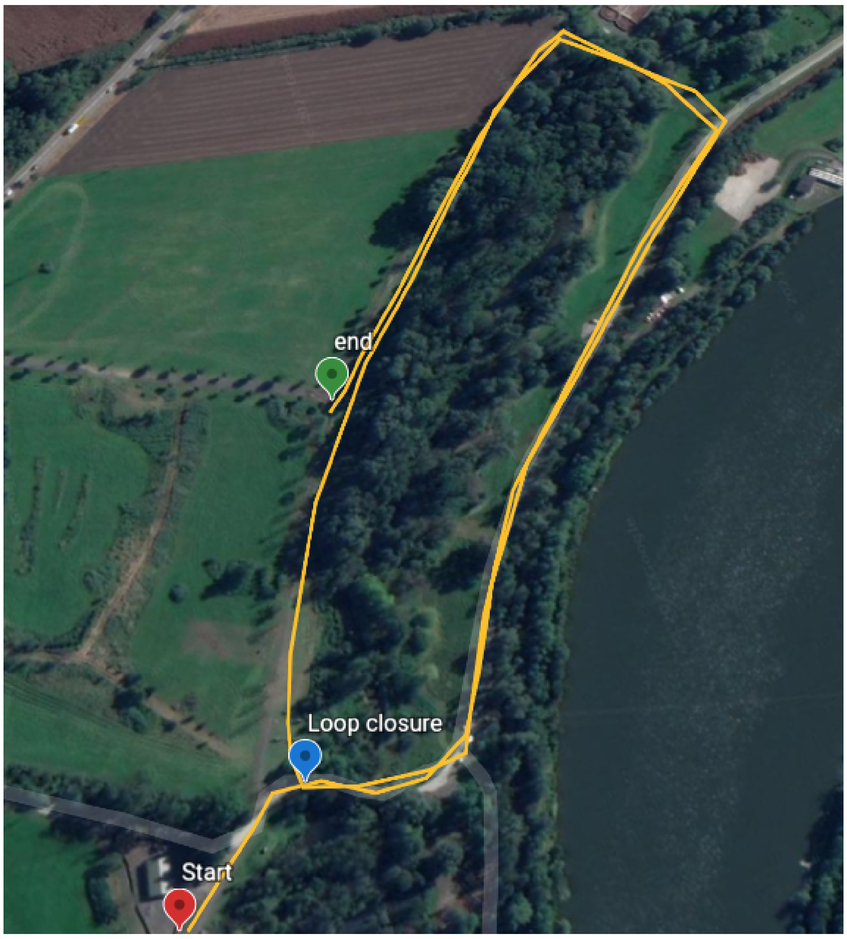


Figure 7.6: Pictures showing the start and end locations, loop closure point, and trajectory of the full-length sequence of off-road data collected at the military training area in Strass, Austria (credits: Google Earth).

8 Evaluation

In this chapter, the different Visual Odometry approaches discussed in the previous chapter are evaluated. The Visual Odometry approaches that are evaluated are ORB-SLAM2, RTAB-Map F2M, RTAB-Map F2F, and SOFT VO. Evaluation is performed using the different off-road datasets discussed in the previous chapter. The off-road datasets are the Davon Island dataset, the Seetaler Alps dataset, and the Strass dataset. All these datasets are different and pose a different set of challenges.

The error metrics used for the evaluation of Visual Odometry algorithms is proposed by (Sturm et al., 2012). The relative pose error (RPE) and absolute trajectory error (ATE) are two frequently used evaluation methods. RPE and ATE are used to compare estimated robot motion trajectory against the true trajectory given by the ground truth system. RPE is generally used to evaluate the drift of the Visual Odometry system or the accuracy at loop closures of SLAM systems by measuring pose difference. In ATE, the two trajectories are aligned first, and absolute pose difference is computed. RPE considers both translational and rotational errors, whereas ATE takes only translational errors into account. In this thesis, the ground truth measurements are available only for translational components, and hence ATE is used as an evaluation method. For an intuitively accessible visualization, trajectories are always shown in bird's eye persepecive.

Given the estimated trajectory $P_{1..K}$, ground truth trajectory $Q_{1..K}$ from K images, the estimated trajectory is aligned to the coordinate frame of ground truth trajectory by manually aligning the orientations of the estimated trajectory with the known orientation of ground truth using rigid body transformation T as explained in Section 3.1 of Chapter 3. Given this transformation, the absolute trajectory error for the image k is computed as

$$E_k := Q_k^{-1} T P_k$$

A root mean squared error (RMSE) is computed over all images as

$$RMSE(E_{1:K}) := \left(\frac{1}{n} \sum_{k=1}^{K} \|trans\,(E_k)\|^2 \right)^{\frac{1}{2}}$$

In addition, median errors are depictd using box plots for better understanding of general performance, as single outlier can significantly affect the final result. For simplicity, ground truth and estimated trajectory sequences are equally sampled. The missing data and incorrect samples are corrected by an interpolation step.

All experiments were conducted on a PC with an Intel CPU Core i7 running at 2.2GHz with 16 GB RAM on Ubuntu OS. In the following sections, the results of the Visual Odometry algorithms using three different off-road datasets are discussed.

8.1 Davon Island rover navigation dataset

The details of data sequences of the Davon Island dataset are discussed in the previous chapter. To evaluate Visual Odometry approaches, sequence 00, 08, 21, and 22 are selected from the entire dataset of 23 sequences. These sequences are selected based on parameters like traversal distance and variablility in the terrain. Sequence 00 is a long run containing more than 2000 images and a distance of around 500 meters. It is a rocky terrain with lots of cliffs. Sequence 08 is a rather straight path, and the terrain has fewer stones and appears sandy. Sequence 21 and 22 are relatively short sequences.

For a motion estimation algorithm to run successfully for the entire set of images, it needs to estimate the pose of the vehicle for every frame. In other words, the algorithm should be able to compute the new position of the robot based on the previous position. In a situation when the algoirthm fails to compute the new robot position from its previous position, the algorithm is said to have suffered from a tracking loss or tracking failure. In some cases, the algorithm is restarted when tracking fails. However, when using only Visual Odometry, the information of previous positions of the robot is completely lost if the tracking fails once. Restarting the algorithm would be helpful only if the previous estimated positions are saved and tracking is restarted from the known position.

Initially, all the Visual Odometry approaches were configured to their standard settings. In the standard configuration, ORB-SLAM2 suffered from tracking loss often within a 50 meters distance. This behavior was noticed in all the evaluated sequences. In some instances, tracking failed soon after initialization. To counter this problem, the first mitigation strategy used was to restart on failure. But, the results were not any better as it would fail very often again after the restart. On the other hand, RTAB-Map F2M, RTAB-Map F2F and SOFT VO suffered from tracking failures only on very few instances because of low feature count in some areas of the robot traversal.

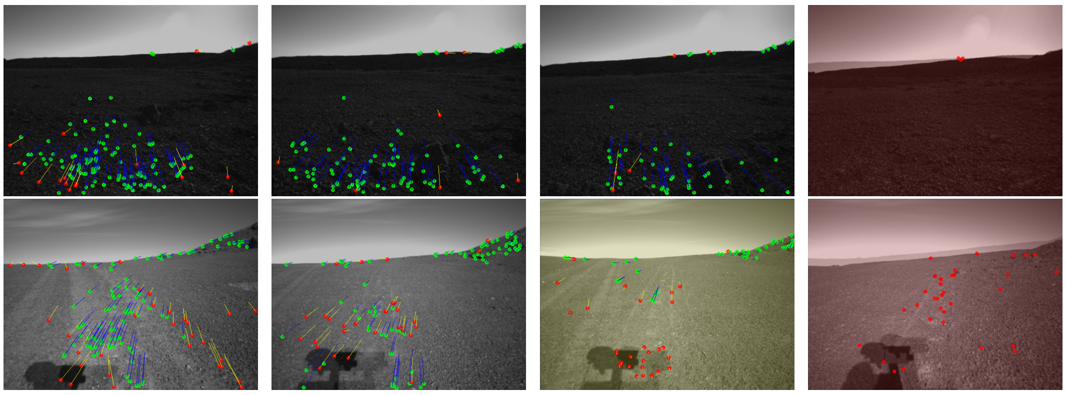
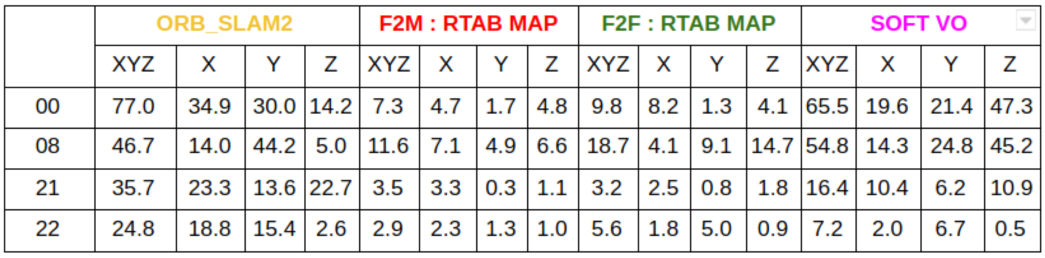


Figure 8.1: In this illustration, feature visualization is provided for two places where the tracking fails in all the approaches. Green points are inlier points; red ones are outliers. From the left images until the right, it can be seen that, as the robot moves, the inlier count reduces, and eventually tracking fails when there are no inliers. The top row images are from sequence 00, and the bottom row images are from sequence 22.

When the number of features configured to be extracted was 1500, in ORB-SLAM2, the number of features actually extracted was low (around 500), and the percentage of features that were matched between the subsequent frames (inliers) were always less than 10 percent (around 50). The inlier count would drop eventually over the next frames and tracking fails as the algorithm needs at least 5 inlier points to estimate motion. A similar issue was noticed in the other Visual Odometry approaches as well but only



	ORB_SLAM2				F2M : RTAB MAP				F2F : RTAB MAP				SOFT VO			
	XYZ	X	Y	Z	XYZ	X	Y	Z	XYZ	X	Y	Z	XYZ	X	Y	Z
00	77.0	34.9	30.0	14.2	7.3	4.7	1.7	4.8	9.8	8.2	1.3	4.1	65.5	19.6	21.4	47.3
08	46.7	14.0	44.2	5.0	11.6	7.1	4.9	6.6	18.7	4.1	9.1	14.7	54.8	14.3	24.8	45.2
21	35.7	23.3	13.6	22.7	3.5	3.3	0.3	1.1	3.2	2.5	0.8	1.8	16.4	10.4	6.2	10.9
22	24.8	18.8	15.4	2.6	2.9	2.3	1.3	1.0	5.6	1.8	5.0	0.9	7.2	2.0	6.7	0.5

Table 8.1: The Root Mean Squared Error (RMSE) for ORB-SLAM2, RTAB-Map F2M, RTAB-Map F2F, and Soft VO is computed against the ground truth. This table shows RMSE in [meters] for x, y, and z axis, and position xyz. xyz is the Euclidian distance between the ground truth position and estimations for translational components.

on one or two instances. Visualization is made in Figure 8.1 to illustrate tracking failure.

Some of the reasons for the lower inlier count were the rocky terrain, large stones, vehicle vibrations and a lower frame rate. A few mitigation strategies were employed to counter tracking failures. As it was clear that the number of features extracted was very low, "number of features" parameter was set to a high value (from 1500 to 5000), and the threshold values for FAST corners were changed from default values (20) to lower values (5). The threshold parameter is basically used to adapt the number of FAST corners per cell in an image. This step reduced the failures to almost 50 percent. However, a significant improvement in terms of number of failures was observed when the "far/close points" threshold was increased to a higher value. In ORB-SLAM2, the far keypoints are excluded from the motion estimation because the scale and the translation information are not reliable. When increasing the far/close points threshold, a large number of keypoints were included for motion estimation. In other Visual Odometry approaches, increasing the number of features parameter resolved the tracking failure problem. The results obtained after using the aforementioned mitigation strategies are given in Table 8.1. The trajectories for the data sequences are shown in Figure 8.2.

From the trajectory plots and the table it is seen that, the trajectory estimates of RTAB-Map F2M and RTAB-Map F2F approaches are close to ground truth

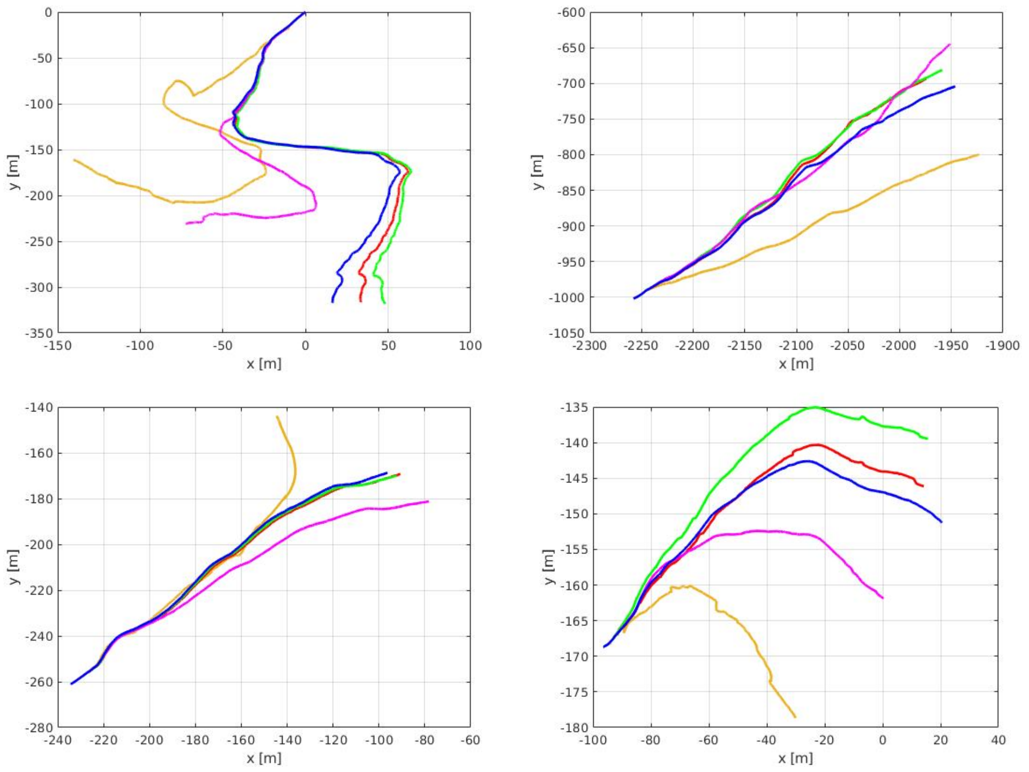


Figure 8.2: The results of ORB-SLAM2 (orange), RTAB-Map F2M (red), RTAB-Map F2F (green), and Soft VO (magenta) for sequence 00, 08, 21, and 22 are shown as 2D plots. The 2D plots are the projection of 3D trajectories on x-y plane. The ground truth is represented in blue.

in all the four sequences. The estimations of SOFT VO and ORB-SLAM2 have suffered large drifts when compared with ground truth trajectory. In the scenarios where robot movement involves large vibrations due to a bump, a large stone, or a small cliff, the robot's motion is highly un-predictable. ORB-SLAM2 often produced wrong estimates and subsequently suffered from more enormous drifts. By increasing the thresold of far/close points, motion estimation of ORB-SLAM2 became unreliable. The reason is, the triangulation of far keypoints resulted in unreliable depth information and therefore motion estimation using these keypoints produced inaccurate scale values. It is clearly visible that when there is a large inter-frame transformation, ORB-SLAM2 estimates are not accurate. In the Davon Island dataset, the frame rate is very low (approx. 1 frame per 20-centimeter movement). Thus, the large inter-frame transformation is also the result of the low frame rate. Figure 8.3 shows the error accumulated by the estimates

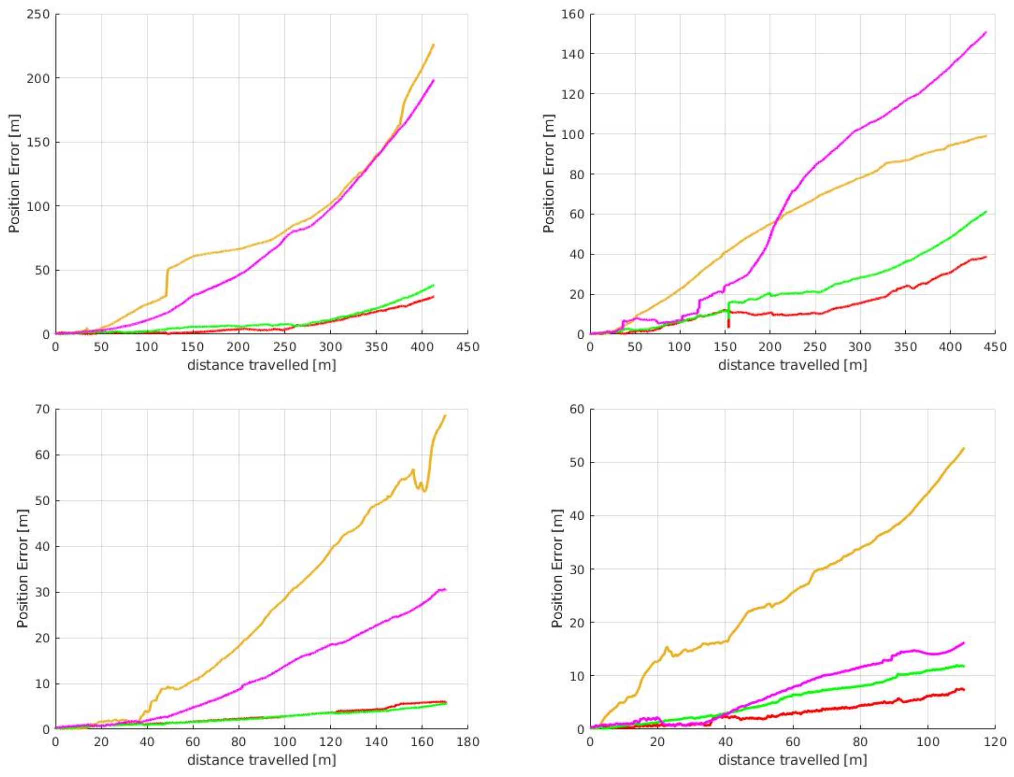


Figure 8.3: Comparison of error accumulation over the total distance travelled for sequence 08, 00, 21, and 22. The plots illustrate the error accumulated (drift) as a function of distance travelled by the rover. The position error is the Euclidean distance between ground truth and estimated position.

of different Visual Odometry approaches. The position error is computed as a function of ATE over distance traveled by the robot. It is shown from the plots that the drift is highest in ORB-SLAM2 followed by SOFT VO. The maximum distance traveled is around 500 meters in sequence 00. RTAB-Map F2M has a final position error of around 40 meters, RTAB-Map F2F goes up to 60 meters, ORB-SLAM2 and SOFT VO have a final position error of around 200 meters. The error in the final pose of the RTAB-Map F2M and F2F are significantly lower than ORB-SLAM2 and SOFT VO. The results in

Table 8.1 show that RTAB-Map F2M and F2F approaches are more suitable for the Mars-like Davon Island dataset. Therefore, a boxplot comparison is made for RTAB-Map F2M and F2F approaches, and shown in Figure 8.4. The plots include the comparison of position error (XYZ), translational error along X, Y, and Z, respectively. From the box plots, it can be seen that the position error, translational error in Y, and translational error in Z are smaller in F2M when compared with F2F appraoch. However, translational error in X is slightly higher in F2M than in F2F approach. As the error is accumulative, the final poses of the estimations exhibit large errors and these erors are visible as outliers in the box plot. Overall, RTAB-MAP F2M achieves the best results in terms of estimation accuracy.

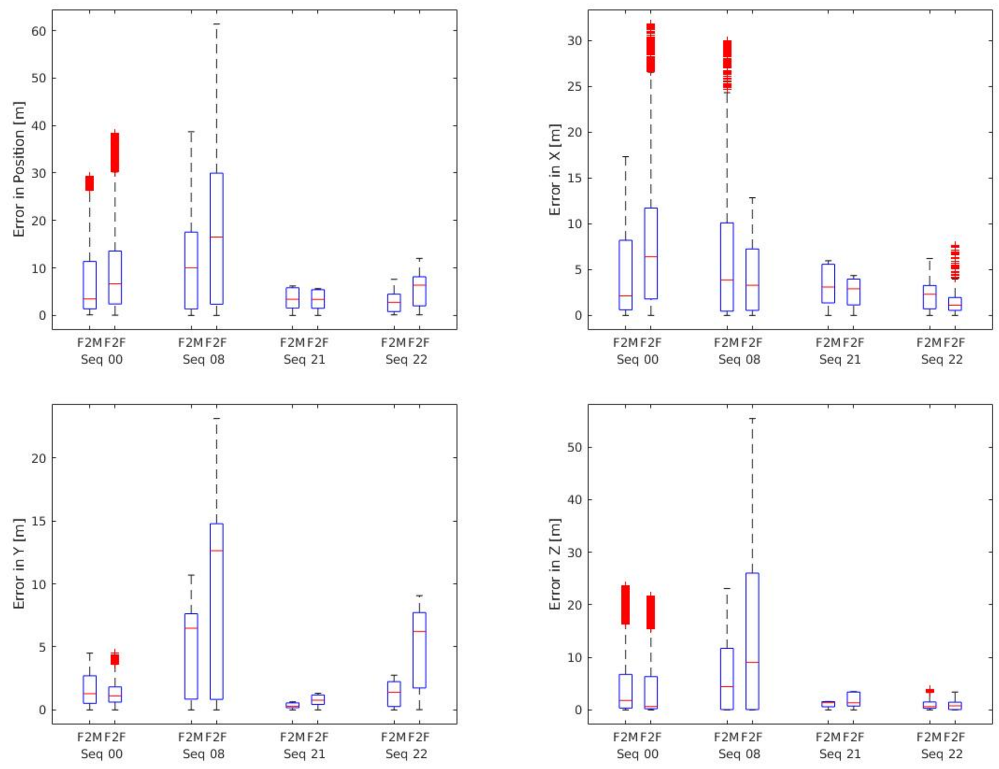


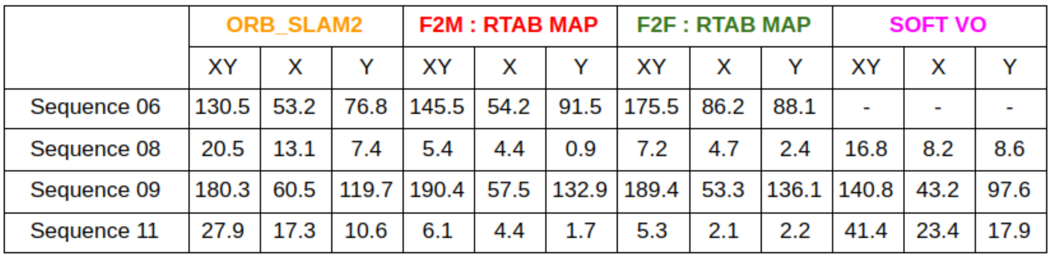
Figure 8.4: Error comparison between RTAB-Map F2M and RTAB-Map F2F. Error in xyz, x, y and z are depicted using the box plots. It is seen from these plots that the estimates of F2M approach is slightly better than F2F. The large errors in the estimations are shown in red.

8.2 Seetaler alps off-road dataset

The data collection details of Seetaler alps off-road dataset is discussed in the previous chapter. In this section we evaluate visual odometry algorithms on Seetaler alps dataset.

Figure 8.5 shows the results of estimated trajectories by Visual Odometry approaches. It can be noticed that the trajectories estimated for sequences 06 and 09 drift too much from the real path. With the standard configuration of the Visual Odometry algorithms, it was observed that all of the approaches ran successfully without any tracking failures. Whereas, Soft VO had issues in tracking, initialization and estimates often failed to compute. The number of features detected were always high but the frame to frame match ratio was too low. In sequence 06, Soft VO failed multiple times and restarting the estimation did not help either. It resulted in completely wrong estimates and hence not included in trajectory plots. It is seen in figure that, the estimates are better in sequences 08 and 11 whereas, there is too much of a drift observed in sequences 06 and 09.

Some of the reasons for the bad estimations are snow, rain, and vehicle vibrations. It could be seen that the estimates at turns are the most affected because of the above reasons. The wrong estimates in turn lead to drift as observed in sequence 06 and sequence 09. The vibrations in the vehicle also affected the performance of Visual Odometry algorithms. Although, in sequence 09 and 11, trajectories are estimated approximate to the ground truth,



	ORB_SLAM2			F2M : RTAB MAP			F2F : RTAB MAP			SOFT VO		
	XY	X	Y	XY	X	Y	XY	X	Y	XY	X	Y
Sequence 06	130.5	53.2	76.8	145.5	54.2	91.5	175.5	86.2	88.1	-	-	-
Sequence 08	20.5	13.1	7.4	5.4	4.4	0.9	7.2	4.7	2.4	16.8	8.2	8.6
Sequence 09	180.3	60.5	119.7	190.4	57.5	132.9	189.4	53.3	136.1	140.8	43.2	97.6
Sequence 11	27.9	17.3	10.6	6.1	4.4	1.7	5.3	2.1	2.2	41.4	23.4	17.9

Table 8.2: The Root Mean Squared Error (RMSE) for ORB-SLAM2, RTAB-Map F2M, RTAB-Map F2F, and Soft VO is computed against the ground truth. This table shows RMSE in [meters] for x, y and position xy

there is no approach that did significantly better than other. Nevertheless, ORB-SLAM2, RTAB-Map F2M and RTAB-Map F2F performed better than Soft VO. In most data sequences, optical flow based Soft VO was affected too much because of vehicle vibrations and resulted in tracking failures and wrong estimates. The RMSE for different approaches is computed as shown in Table 8.2.

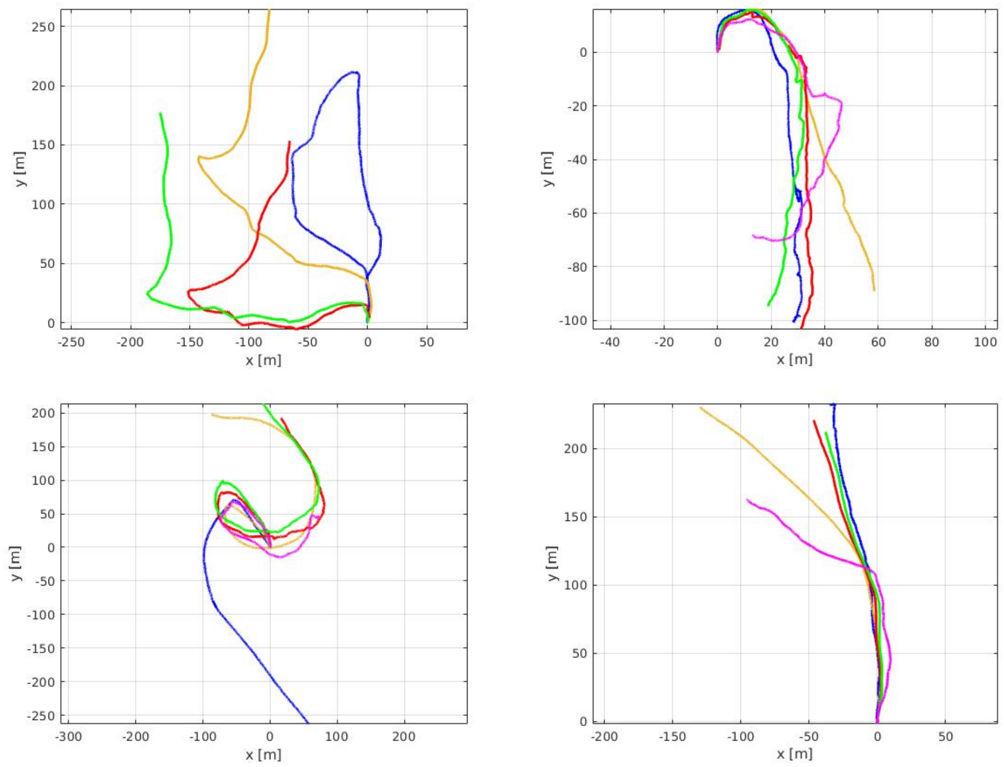


Figure 8.5: The results of ORB-SLAM2 (orange), RTAB-Map F2M (red), RTAB-Map F2F (green) and Soft VO (magenta) for the Seetaler Alps dataset sequence 06, 08, 09, and 11 are shown as 2D plots. The 2D plots are the projection of 3D trajectories on x-y plane. The ground truth is represented in blue.

It can be seen in Figure 8.6, the overall performance of RTAB-Map F2M and RTAB-Map F2F were slightly better than ORB-SLAM2 in sequences 08 and 11. In Sequence 06, it was observed that wrong estimates in the corners and turns lead to large accumulation of errors and drifted too much from the real trajectory. Even after including loop closure detection, places visited before were not recognized, and place detection strategy was heavily affected by environment conditions. There were some instances where loop closure detection was very inaccurate, and closing loops on wrong places.

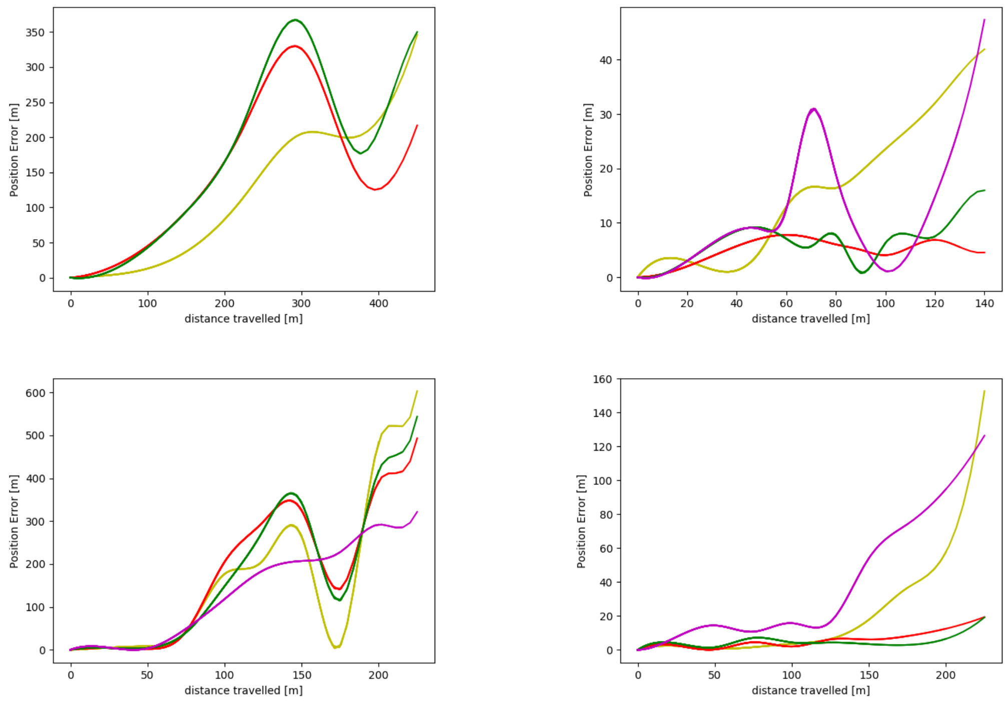


Figure 8.6: Comparison of Error accumulation over the total distance travelled for sequence 06, 08, 09, and 11. The plots illustrate the error accumulated (drift) as a function of distance travelled by the rover. The position error is the Euclidean distance between ground truth and estimated position.

8.3 Strass off-road dataset

The details of the Strass dataset were discussed in the previous chapter. In this section, different visual odometry approaches are evaluated and compared against ground truth positions. The data sequence contains an initial sychronization pattern of circles followed by the full length trajectory. To assess Visual Odometry better, the synchronization pattern is not included in evaluation. The full trajectory and sychronization pattern are evaluated using SLAM variant of the Visual Odometry approaches.

Using the default configuration settings, it was observed that ORB-SLAM2 ran successfully without any tracking failures. Tracking failed once in SOFT VO while taking a large turn, but there were a few such failures in RTAB-Map F2M and F2F approaches. Even with the increase in the number of detected features, the percentage of inliers and matches did not increase and resulted in loss of tracking in a couple of occassions. An increase in the threshold value of feature matcher solved the problem of tracking failures. Figure 8.7 shows the results of estimated trajectories provided by the Visual Odometry approaches. It can be noticed that the trajectories of ORB-SLAM2 and SOFT VO are close to ground truth whereas, RTAB-Map F2M and F2F approaches drift too much.

One of the significant issues observed was, estimations of RTAB-Map were less accurate in sharp turns. Thus, estimates of visual odometry drifted and large error accumulated as noted in the RMSE Table 8.3. On the other hand, ORB-SLAM2 and Soft VO drifted less, although the latter was better than the former in the final pose's accuracy. ORB-SLAM2 and Soft VO give the best results for this dataset. Comparative plots for ORB-SLAM2 and Soft VO are depicted in 8.8. It is worth mentioning that ORB-SLAM2's estimations were accurate for most of the trajectory, but there was a large drift due to rotation, and hence Soft VO had a better final pose accuracy. The box plot in Figure 8.8 shows that ORB-SLAM2 exhibits a larger error in the Y direction and results in a larger X-Y pose error.

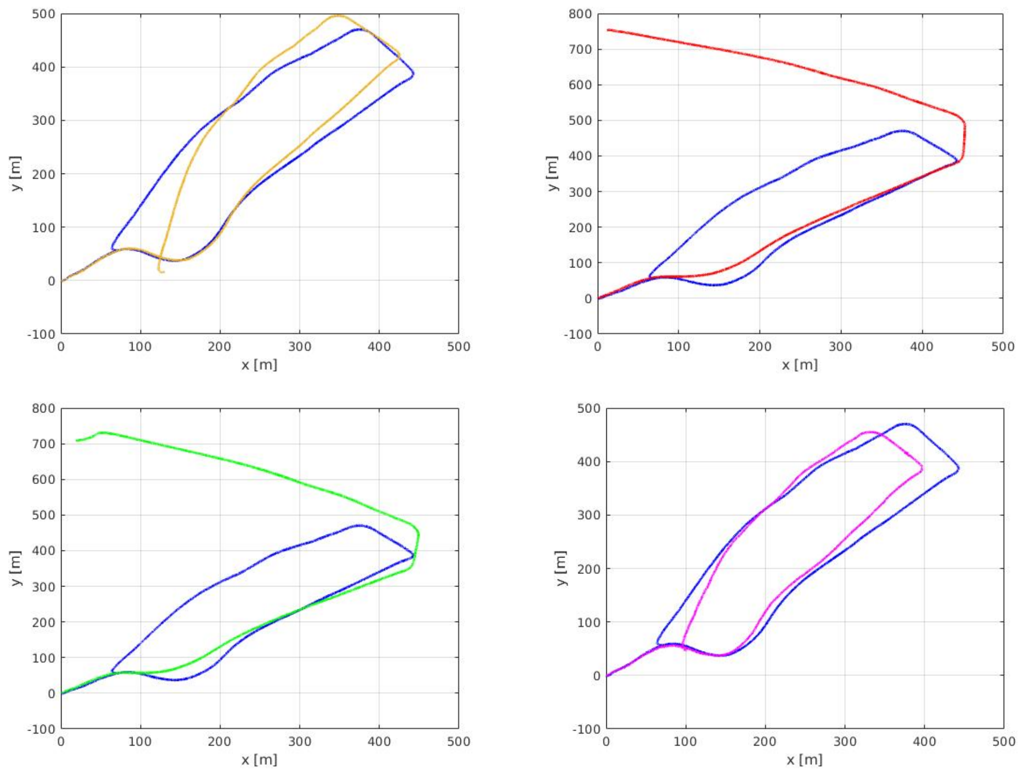


Figure 8.7: The results of ORB-SLAM2 (orange), RTAB-Map F2M (red), RTAB-Map F2F (green) and Soft VO (magenta) for the Strass dataset are shown as 2D plots. The 2D plots are the projection of 3D trajectories on x-y plane. The ground truth is represented in blue.

	ORB_SLAM2			F2M : RTAB MAP			F2F : RTAB MAP			SOFT VO		
	XY	X	Y	XY	X	Y	XY	X	Y	XY	X	Y
SEQ 01	21.8	14.1	15.6	205.9	43.9	192.3	196.8	39.7	182.3	20.4	18.6	5.9

Table 8.3: The Root Mean Squared Error (RMSE) for ORB-SLAM2, RTAB-Map F2M, RTAB-Map F2F, and Soft VO is computed against the ground truth. This table shows RMSE in [meters] for x and y, and position xy.

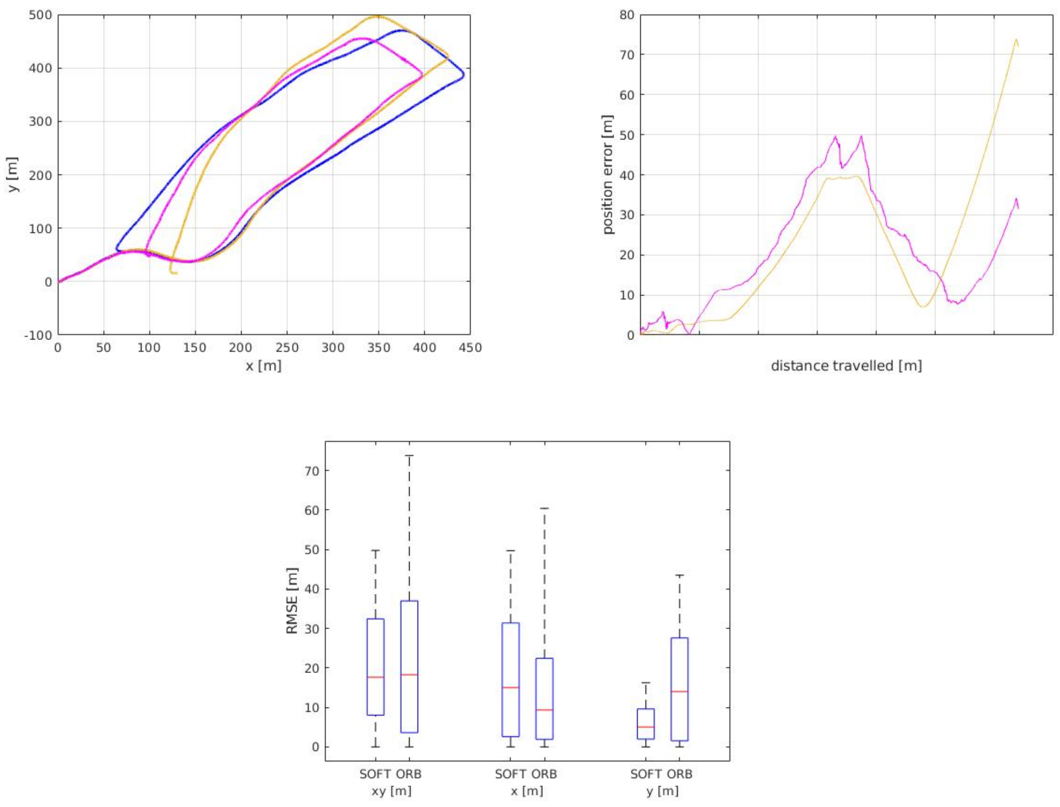


Figure 8.8: A comparison of ORB-SLAM2 (without loop closure and global optimization) (orange) vs Soft VO (magenta) is depicted in these plots for the Strass dataset. The top left plot is the estimated trajectories of SOFT VO, ORB-SLAM2, and the ground truth. The top right picture is the position error function over the distance travelled. The bottom picture is the box plot of RMSE comparison of ORB-SLAM2 with SOFT VO for translational error in X, Y, and position error X-Y.

In this section so far, only Visual Odometry approaches are evaluated where loop closure and global optimization techniques have been removed from the standard SLAM implementation. One of the vital features of ORB-SLAM2 is its loop closing and full bundle adjustment threads. We use this ability to evaluate the performance of loop closure and global optimization. A significant improvement was noticed in ORB-SLAM2 compared to its results without loop closure and full global bundle adjustment. A comparative analysis and the estimated trajectory of ORB-SLAM2 VO and its SLAM variant are shown in Figure 8.9.

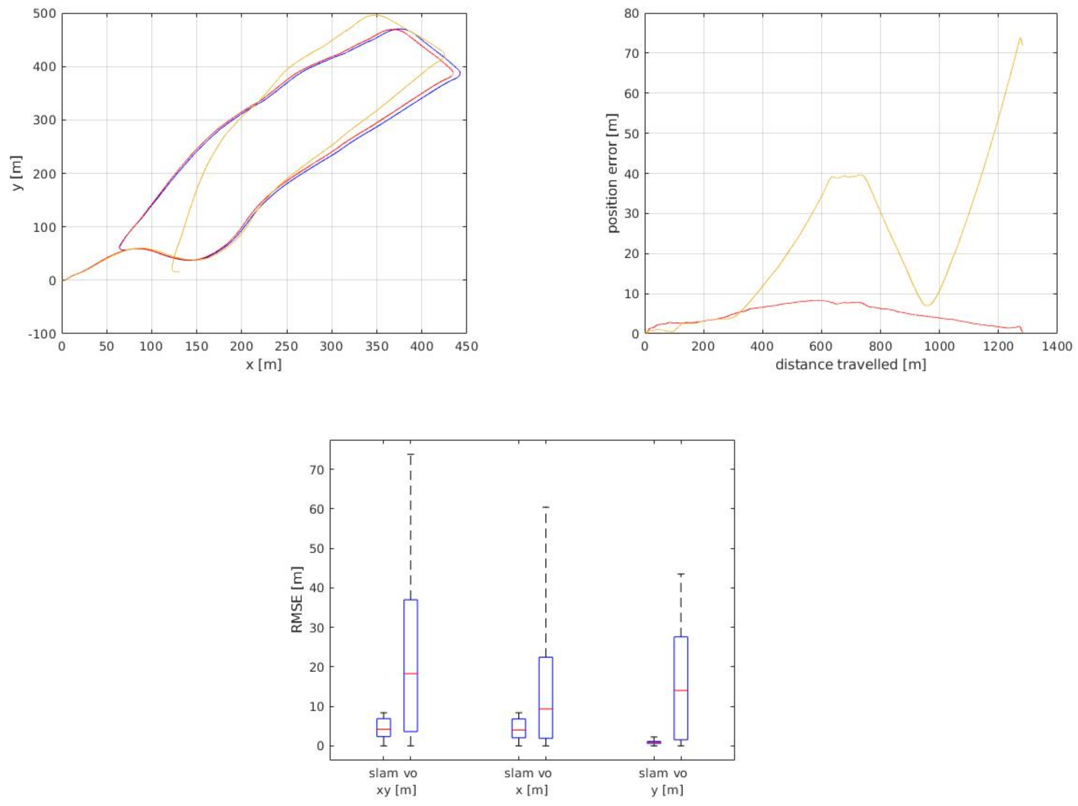


Figure 8.9: A comparison of ORB-SLAM2 SLAM (Red) vs Visual Odometry (orange) is depicted in the plots above. A significant performance is noticed as SLAM corrects the drift error introduced by VO. The final position error of 72 meters is reduced to 0.1 meters as seen in the top right plot. RMSE is compared in the box plot for SLAM vs Visual Odometry.

With loop closure and bundle adjustment, the average accumulated trajectory error is optimized to 4 meters from an average error of 19 meters. The final pose error in ORB-SLAM2 (only VO) was 72 meters, but after loop closure and global optimization, the error reduced to less than 0.1 meters. In addition to ATE as an error measure, we evaluate the performance improvement when loop closure and bundle adjustment were added to pure Visual Odometry. An improvement of 58.7 percent was noticed when loop closure and global bundle adjustment were added to Visual Odometry.

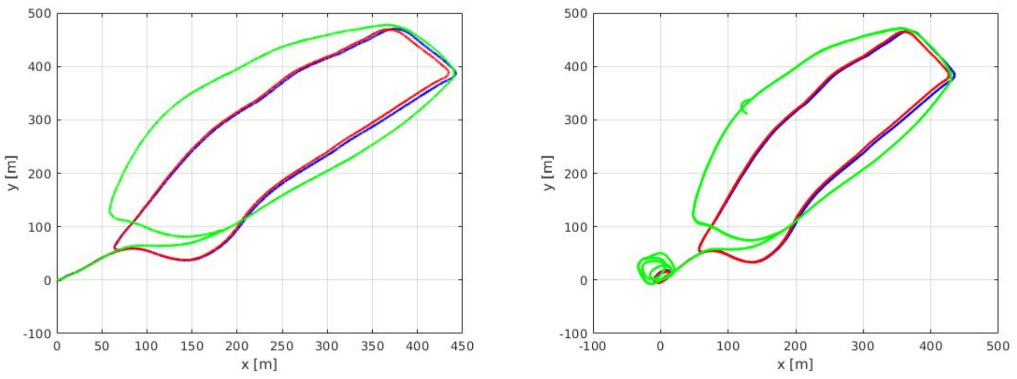


Figure 8.10: Trajectory estimation of ORB-SLAM2(red) and RTAB-Map(green) against Ground truth(blue) with loop closure and global optimization. Left plot is the sequence without synchronization pattern and the right plot is the trajectory estimates for the entire sequence of images.

On the other hand, due to a large drift error, the loop closure and the global optimization scheme employed by RTAB-Map were sub optimal. The plots of RTAB-Map and Orbslam are shown in 8.10.

8.4 3D maps

A Dense 3D map can be created using RTAB-Map 3D reconstruction module. Figures 8.11 and 8.12 show the 3D maps for sequence 00 and 22 of the Davon Island dataset. Figures 8.13 shows the 3D map for a part of the Strass dataset and Figure 8.14 shows the 3D map for a part of sequence 11 of the Seetaler Alps dataset. The map is represented by optimized robot trajectory (blue) and the optimized dense point cloud. The dense point cloud map created in the Davon Island dataset show the complete terrain as there is no vegetation in the rover traversal area. On the other hand, only parts of the map are shown in Figures 8.13 and 8.14.

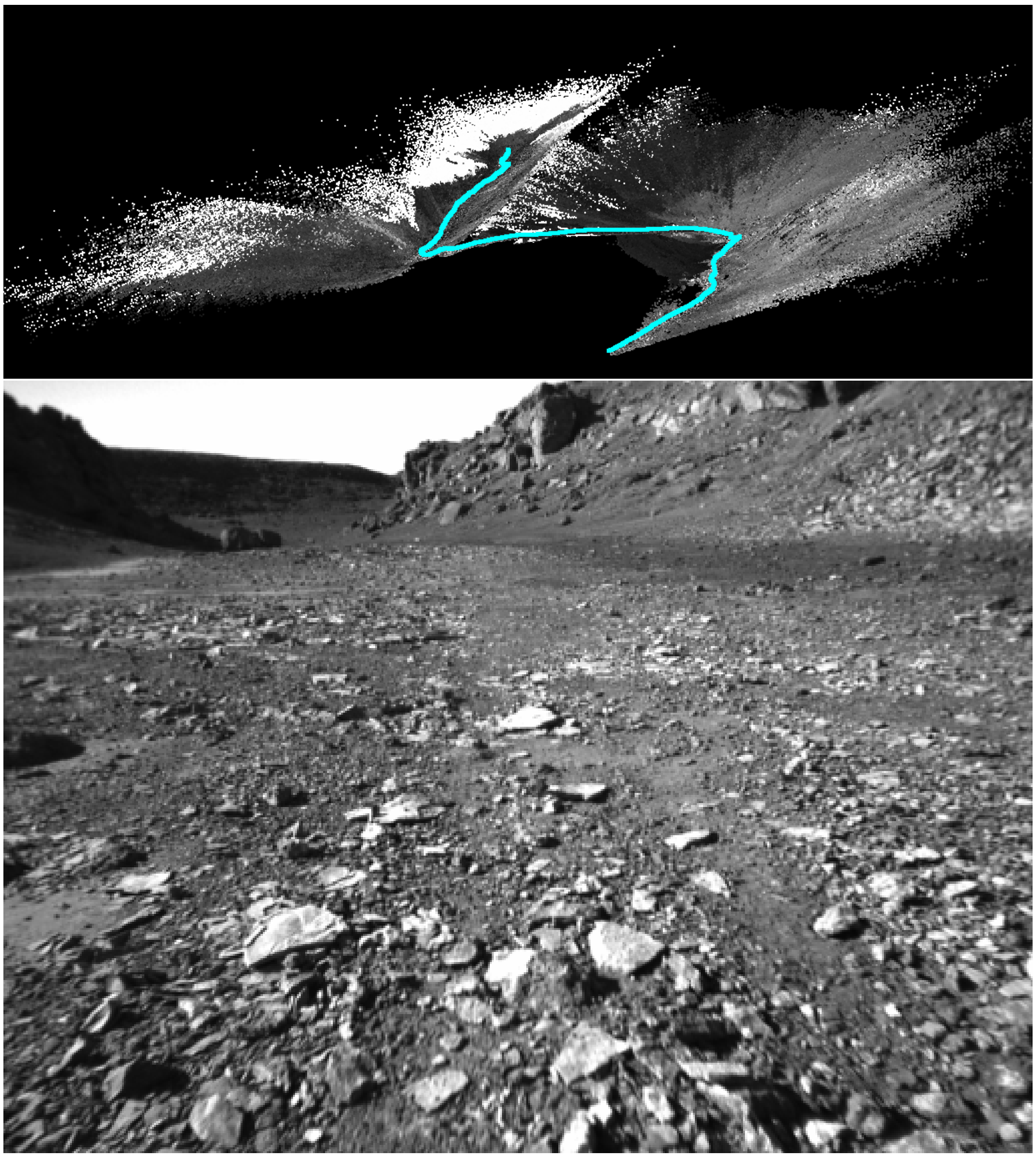


Figure 8.11: The picures show the 3D pointcloud map (top picture) created using RTAB-Map for the Davon Island navigation dataset sequence 00, and a sample image (bottom picture) from the sequence 00.

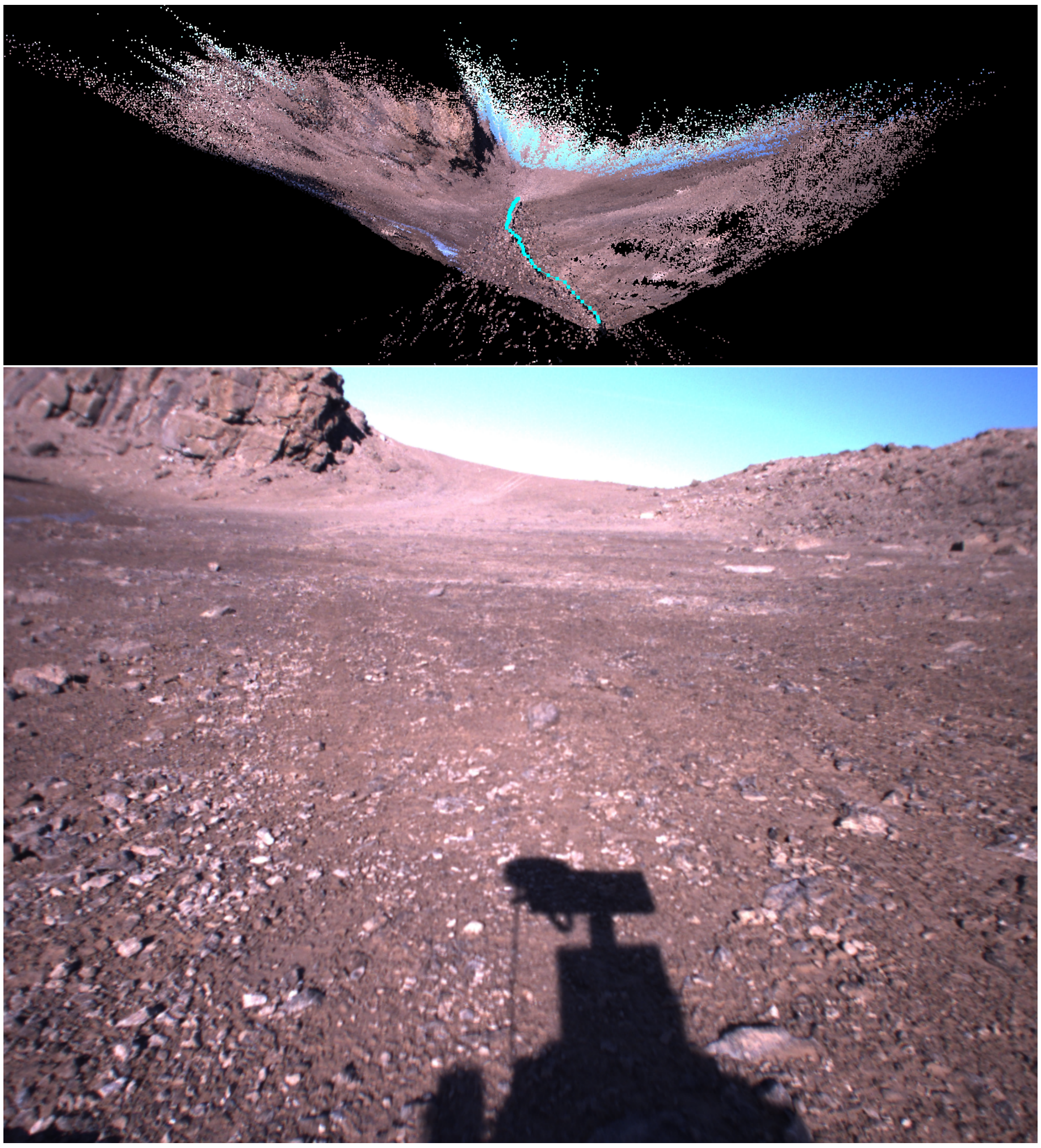


Figure 8.12: The picures show the 3D pointcloud map (top picture) created using RTAB-Map for the Davon Island navigation dataset sequence 22, and a sample image (bottom picture) from the sequence 22.

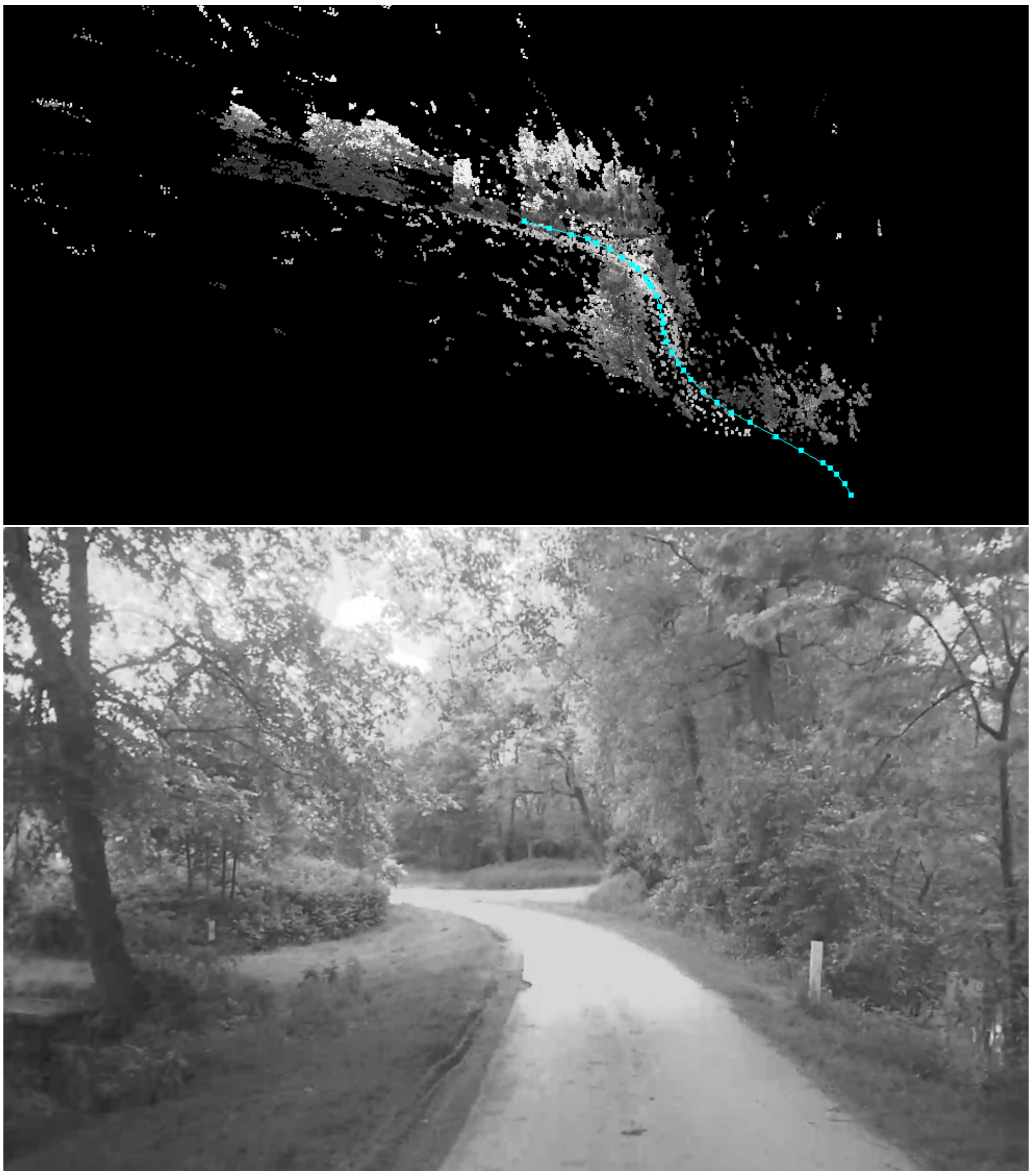


Figure 8.13: The picures show the 3D pointcloud map (top picture) created using RTAB-Map for the Strass dataset, and a sample image (bottom picture) from the Strass dataset.

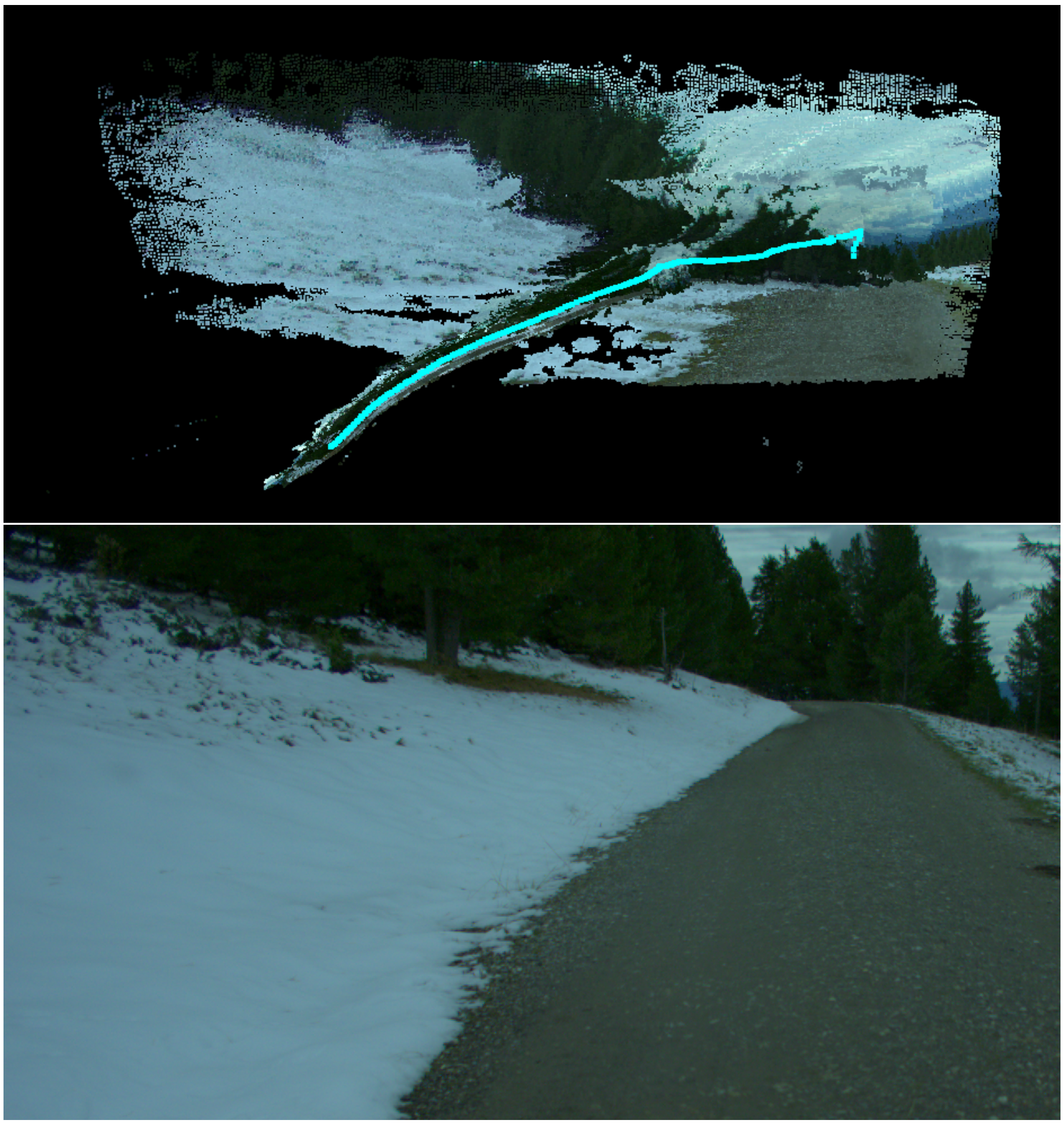


Figure 8.14: The picures show the 3D pointcloud map (top picture) created using RTAB-Map for the Seetaler Alps dataset sequence 11, and a sample image (bottom picture) from the sequence 11.

9 Conclusion

This chapter summarizes this thesis and discuss potential future improvements to this work.

9.1 Discussion

In this thesis, different feature-based Visual Odometry approaches are evaluated using off-road datasets. The use of the Vision sensor as a primary sensor for robot navigation in various off-road conditions is assessed. The performance of these techniques on challenging datasets has been evaluated quantitatively in terms of accuracy of position estimations. Moreover, a Mars-like simulation environment is created to allow for a easy quick and initial evaluation of Visual Odometry techniques in vision-based sensor configuration in GPS-denied environments. Performance evaluation in a simulated environment does not guarantee to behave similarly in a real environment. On one hand an existing Mars-like rover navigation dataset was selected. On the other hand, we collected our own challenging off-road dataset at the military training ground that features a challenging alpine environment in the Seetaler Alps, Austria. In order to be able to perform data recording, an appropriate stereo camera system was designed, built and configured. We also evaluated a challenging off-road dataset collected at the military training ground at Strass, Austria.

The Visual Odometry algorithms that were evaluated are ORB-SLAM2, RTAB-MAP F2M, RTAB-MAP F2F, SOFT VO. Evaluating different feature-based Visual odometry methods shows that Visual Odometry techniques largely depend on the type of the terrain. It was also shown that these techniques require much fine-tuning for a specific environment. In the Davon

Island Mars dataset, RTAB-Map F2M and RTAB-Map F2F outperformed ORB-SLAM2 and Soft VO. The results showed that ORB-SLAM2 drifts too much from the real robot path, and tracking often failed. A higher level of accuracy is reached by using the RTAB-Map F2M approach. On the other hand, in the Seetaler Alps off-road dataset, in the presence of snow, rain, and adverse conditions, all the approaches failed to achieve good results as the estimated trajectories tend to drift too much from the real one. Along with challenging environmental conditions, it was noticed that vehicle vibrations caused unreliable estimations and sometimes a failure in tracking. In the Strass dataset, ORB-SLAM2 and Soft VO performed well, but wrong estimates in large turns caused results of RTAB-Map F2M and RTAB-Map F2F to drift too much from the ground truth trajectory. Additionally, pure Visual Odometry performance was compared against full SLAM (loop closure and global bundle adjustment). The accuracy of absolute trajectory estimation improved by 58.7 percent by visual SLAM compared to Visual Odometry.

In direct comparison to the sparse-map of ORB-SLAM2, RTAB-Map's dense reconstruction contained far more details and is well suited for various navigation applications like detecting obstacles.

Overall, the evaluation results showed that RTAB-Map F2M achieves the best accuracy in the Davon Island Mars dataset. ORB-SLAM2 outperformed ORB-SLAM2 in the Strass dataset, and none of these approaches provided useful results in the Seetaler Alps dataset. A pure vision-based navigation in off-road terrain is hence proved to be very difficult. Especially in challenging scenarios, it is seen that tracking loss and trajectory drift are inevitable. Potential improvements to achieve better localization accuracy are discussed in the next section.

9.2 Future Work

The results of RTAB-Map for the Mars dataset showed that it achieves high accuracy in challenging off-road terrain. However, unfortunately, the estimations drifted too much from the ground truth in the Strass dataset. RTAB-Map is a software library that allows users to configure and tune the software based on the application. As seen in the Strass dataset results, RTAB-Map's estimations were wrong in corners and sharp turns. As a potential solution to this problem, various parameters of RTAB-Map software could be tuned and tested until satisfactory results are attained. One could try out different feature extractor and matcher combinations. One could also change the motion estimation approach from 3D-2D to 3D-3D. It is also worth changing bundle adjustment parameters by varying the distance threshold of visible feature points in 3D space.

Additional sensors like inertial measurement units (IMU) could be used to correct the drift by fusing their with the data from the stereo camera sensor. A new version of ORB-SLAM called ORB-SLAM3 uses IMU and stereo camera together to estimate the robot trajectory (Campos et al., 2020). As ORB-SLAM2 already achieves the best results in the Strass dataset, an extensive evaluation of ORB-SLAM3 on the Mars dataset could solve the drift problem. On the other hand, Soft VO performed well in the Strass dataset and achieved relatively good results in the Mars dataset. A full SLAM implementation of Soft VO could be evaluated and tested against different datasets.

A more promising way to correct drifts is to use the deep learning-based method "Self-supervised deep pose corrections for Robust Visual Odometry" (Wagstaff, Peretroukhin, and Kelly, 2020). It uses a classical visual odometry estimated output as a prior and corrects the estimates using a deep neural network. A possible way to integrate this is to use RTAB-Map's F2M approach as a visual odometry pipeline and correct the drifts using a deep neural network. As RTAB-Map's estimates drifted too much in sharp turns in the Strass dataset, integrating deep learning-based pose correction step could enhance the localization accuracy.

Bibliography

Bradski, G. (2000). "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (cit. on pp. 11, 22).

Campos, Carlos et al. (2020). "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM." In: *arXiv preprint arXiv:2007.11898* (cit. on p. 79).

Cvišić, Igor et al. (2018). "SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles." In: *Journal of field robotics* 35.4, pp. 578–595 (cit. on pp. 24, 35).

Engel, J., J. Stückler, and D. Cremers (2015). "Large-scale direct SLAM with stereo cameras." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1935–1942. DOI: 10.1109/IROS.2015.7353631 (cit. on p. 25).

Fraundorfer, F. and D. Scaramuzza (2012). "Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications." In: *IEEE Robotics Automation Magazine* 19.2, pp. 78–90. DOI: 10.1109/MRA.2012.2182810 (cit. on pp. 19, 22, 23).

Furgale, Paul et al. (2012). "The Devon Island rover navigation dataset." In: *The International Journal of Robotics Research* 31.6, pp. 707–713 (cit. on pp. 23, 49, 50).

Gálvez-López, Dorian and J. D. Tardós (Oct. 2012). "Bags of Binary Words for Fast Place Recognition in Image Sequences." In: *IEEE Transactions on Robotics* 28.5, pp. 1188–1197. ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2197158 (cit. on p. 21).

Geiger, Andreas, Philip Lenz, and Raquel Urtasun (2012). "Are we ready for autonomous driving? the kitti vision benchmark suite." In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3354–3361 (cit. on pp. 4, 26).

Gonzalez, Ramón et al. (Oct. 2012). “Combined visual odometry and visual compass for off-road mobile robots localization.” In: *Robotica* 30, pp. 1–14. DOI: 10.1017/S026357471100110X (cit. on p. 23).

Grimes, M. and Y. LeCun (2009). “Efficient off-road localization using visually corrected odometry.” In: *2009 IEEE International Conference on Robotics and Automation*, pp. 2649–2654. DOI: 10.1109/ROBOT.2009.5152880 (cit. on p. 23).

Groemer, G. et al. (2020). “The AMADEE-18 Mars Analog Expedition in the Dhofar Region of Oman.” In: *Astrobiology* 20 11, pp. 1276–1286 (cit. on p. 2).

Halatschek, Richard et al. (Nov. 2020). “Universal Offroad Robot Platform for Disaster Response.” English. In: 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics , SSRR 2020 ; Conference date: 04-11-2020 Through 06-11-2020 (cit. on pp. 9, 51).

Jianbo Shi and Tomasi (1994). “Good features to track.” In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600. DOI: 10.1109/CVPR.1994.323794 (cit. on p. 36).

Klein, G. and D. Murray (2007). “Parallel Tracking and Mapping for Small AR Workspaces.” In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852 (cit. on p. 24).

Koenig, N. and A. Howard (2004). “Design and use paradigms for Gazebo, an open-source multi-robot simulator.” In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727 (cit. on p. 8).

Kümmerle, R. et al. (2011). “G2o: A general framework for graph optimization.” In: *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949 (cit. on p. 17).

Labbé, Mathieu and François Michaud (2019). “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation.” In: *Journal of Field Robotics* 36.2, pp. 416–446 (cit. on pp. 24, 31).

Lambert, Andrew et al. (May 2012). “Field Testing of Visual Odometry Aided by a Sun Sensor and Inclinometer.” In: *J. Field Robot.* 29.3, pp. 426–444. ISSN: 1556-4959. DOI: 10.1002/rob.21412. URL: https://doi.org/10.1002/rob.21412 (cit. on p. 23).

Lepetit, Vincent, Francesc Moreno-Noguer, and Pascal Fua (Feb. 2009). "EPnP: An accurate O(n) solution to the PnP problem." In: *International Journal of Computer Vision* 81. DOI: 10.1007/s11263-008-0152-6 (cit. on p. 22).

Madsen, Kaj, Hans Nielsen, and O Tingleff (Jan. 2004). "Methods for Non-Linear Least Squares Problems (2nd ed.)" In: p. 60 (cit. on p. 18).

Maimone, Mark, Yang Cheng, and Larry Matthies (Mar. 2007). "Two Years of Visual Odometry on the Mars Exploration Rovers." In: *J. Field Robotics* 24, pp. 169–186. DOI: 10.1002/rob.20184 (cit. on p. 23).

Matthies, L. and S. Shafer (1987). "Error modeling in stereo navigation." In: *IEEE Journal on Robotics and Automation* 3.3, pp. 239–248. DOI: 10.1109/JRA.1987.1087097 (cit. on p. 23).

Moravec, H. (1980). "Obstacle avoidance and navigation in the real world by a seeing robot rover." In: (cit. on p. 23).

Mur-Artal, Raul, J. M. M. Montiel, and Juan D. Tardos (Oct. 2015). "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." In: *IEEE Transactions on Robotics* 31.5, pp. 1147–1163. ISSN: 1941-0468. DOI: 10.1109/tro.2015.2463671. URL: http://dx.doi.org/10.1109/TRO.2015.2463671 (cit. on p. 24).

Mur-Artal, Raul and Juan D Tardós (2017). "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." In: *IEEE Transactions on Robotics* 33.5, pp. 1255–1262 (cit. on pp. 24, 27, 28).

Nister, D., O. Naroditsky, and J. Bergen (2004). "Visual odometry." In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1, pp. I–I. DOI: 10.1109/CVPR.2004.1315094 (cit. on p. 23).

Nourani-Vatani, Navid and Jonathan M Roberts (2007). "Automatic camera exposure control." In: *Proceedings of the Australasian Conference on Robotics and Automation 2007*. Australian Robotics & Automation Association ARAA, pp. 1–6 (cit. on p. 45).

Paz, L.M. et al. (Nov. 2008). "Large scale 6DOF slam with stereo-in-hand." In: *Robotics, IEEE Transactions on* 24, pp. 946–957. DOI: 10.1109/TRO.2008.2004637 (cit. on p. 24).

Peretroukhin, Valentin and Jonathan Kelly (2017). "Dpc-net: Deep pose correction for visual localization." In: *IEEE Robotics and Automation Letters* 3.3, pp. 2424–2431 (cit. on p. 25).

Pire, Taihú et al. (Apr. 2017). "S-PTAM: Stereo Parallel Tracking and Mapping." In: *Robotics and Autonomous Systems* 93. DOI: 10.1016/j.robot.2017.03.019 (cit. on p. 24).

Quigley, Morgan et al. (n.d.). "ROS: an open-source Robot Operating System." In: () (cit. on p. 7).

Rosten, E., R. Porter, and Tom Drummond (2010). "Faster and Better: A Machine Learning Approach to Corner Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, pp. 105–119 (cit. on p. 24).

Scaramuzza, D. and F. Fraundorfer (2011). "Visual Odometry [Tutorial]." In: *IEEE Robotics Automation Magazine* 18.4, pp. 80–92. DOI: 10.1109/MRA.2011.943233 (cit. on pp. 17, 18, 23).

Shim, I., J. Lee, and I. S. Kweon (2014). "Auto-adjusting camera exposure for outdoor robotics using gradient information." In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1011–1017. DOI: 10.1109/IROS.2014.6942682 (cit. on p. 45).

Steinbauer, Gerald et al. (2020). "AMADEE-20 Exploration Cascade using Robotic Vehicles." In: *IROS 2020-Workshop on Planetary Exploration Robots* (cit. on p. 3).

Stradner, M. and G. Steinbauer (2019). "Lifting Robot Exploration to 3D Environments." In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–8. DOI: 10.1109/SSRR.2019.8848942 (cit. on p. 2).

Strasdat, Hauke, J. M. M. Montiel, and Andrew J. Davison (Feb. 2012). "Editors Choice Article: Visual SLAM: Why Filter?" In: *Image Vision Comput.* 30.2, pp. 65–77. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2012.02.009. URL: https://doi.org/10.1016/j.imavis.2012.02.009 (cit. on p. 24).

Sturm, Jürgen et al. (2012). "A benchmark for the evaluation of RGB-D SLAM systems." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 573–580 (cit. on p. 57).

Wagstaff, Brandon, Valentin Peretroukhin, and Jonathan Kelly (2020). "Self-Supervised Deep Pose Corrections for Robust Visual Odometry." In: *arXiv preprint arXiv:2002.12339* (cit. on pp. 25, 79).

Wang, Sen et al. (2017). "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2043–2050 (cit. on p. 25).

Wu, Er-yong et al. (Apr. 2008). “Stereo vision based SLAM using Rao-Blackwellised particle filter.” In: *Journal of Zhejiang University - Science A: Applied Physics & Engineering* 9, pp. 500–509. DOI: 10.1631/jzus.A071361 (cit. on p. 24).

Yang, Yi et al. (2020). “Multi-camera visual SLAM for off-road navigation.” In: *Robotics and Autonomous Systems* 128, p. 103505. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2020.103505. URL: http://www.sciencedirect.com/science/article/pii/S0921889019308711 (cit. on p. 25).

Yousif, Khalid, Alireza Bab-Hadiashar, and Reza Hoseinnezhad (Nov. 2015). “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics.” In: *Intelligent Industrial Systems* 1. DOI: 10.1007/s40903-015-0032-7 (cit. on p. 24).