UNIVERSITÀ DEGLI STUDI DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Centro di Ateneo di Studi e Attività Spaziali "Giuseppe Colombo" - CISAS

CORSO DI DOTTORATO DI RICERCA IN: Scienze Tecnologie e Misure Spaziali - STMS
CURRICOLO: Misure Meccaniche per l'Ingegneria e lo Spazio
CICLO XXXII

STEREO AND MONOCULAR VISION GUIDANCE FOR AUTONOMOUS AERIAL AND GROUND VEHICLES

Coordinatore: Ch.mo Prof. Giampiero Naletto
Supervisore: Ch.mo Prof. Stefano Debei

Dottorando : Riccardo Giubilato

Abstract

Robotic agents vastly increase the return of planetary exploration missions thanks to their ability of performing in-situ measurements. To this date, unmanned exploration has been performed by individual of robots such as the MER Spirit and Opportunity and later MSL Curiosity.

A fundamental asset to robotic autonomy is the ability to perceive the surroundings through vision systems such as stereo cameras. Since global localization using GPS-like approaches is unavailable in extra-terrestrial environments, rovers need to measure their motion in order to understand where they are heading. This allows to close high-level control loops to follow planned routes toward goals of scientific interest. Visual SLAM (Simultaneous Localization and Mapping) is an effective strategy to fulfill these needs. Stereo cameras are used to both reconstruct the environment structure through triangulation and use that information to localize the cameras while moving. While performing Visual SLAM on constrained resources is still challenging, many state of the art solution exist to solve this problem for single exploration sessions.

The future of planetary exploration however strongly involves cooperation amongst teams of heterogeneous robotic agents. While the SLAM problem is efficiently solved for single sessions and agents, robust solutions for collaborative map merging and re-localization are still topics of active research and constitute the first major objective of this thesis. Here is proposed and validated a robust re-localization pipeline targeted at planetary vehicles equipped with stereo vision systems allowing to localize them in previously built maps. Instead of common Visual SLAM approaches based exclusively on visual features, this algorithm exploits the invariant nature of 3D point clouds by using compact 3D binary descriptors in conjunction with texture cues. Maps are discretized in submaps which are represented in a lightweight form using the Bag of Binary Words paradigm. The algorithm is then tested and validated both in the laboratories of the DLR Robotics and Mechatronics Center and in Mount Etna, Sicily, an outdoor planetary analogous environment.

The second major research objective involves monocular vision for UAVs. Stereo depth perception is often infeasible for UAVs as small baseline systems degenerate to monocular as the vehicle takes off. 3D structure can be obtained using Structure-from-Motion approaches which are however unable to recover a global metric scale. Scale is traditionally recovered integrating accelerations from IMUs. However visual-inertial sensing is delicate being very sensitive on wrong extrinsic calibration. In addition, initialization of the visual-inertial pipeline is challenging and can diverge. These reasons challenge the implementation of unsupervised autonomous behaviors on UAVs. To address these issues, this thesis work proposes a sensor fusion approach between cameras and low resolution range sensors in order to exploit direct range measurements enforcing scale constraints in monocular Visual Odometry. This research objective is accomplished in two stages. Firstly a monocular Visual Odometry is developed without enforcing strict performance constraints and is used in conjunction with a low resolution Time of Flight camera, a lightweight sensor capable of measuring 64 ranges in a narrow Field-of-View. The algorithm is tested against both a

state of the art stereo visual SLAM system and a more accurate, while heavier, 2D LiDAR. Finally, a real-time monocular Visual Odometry is developed exploiting a multi-threaded architecture to enable concurrent tracking of the camera pose and scale optimization in the background. This algorithm is tested with a 1D LiDAR altimeter, a minimal range sensing configuration of just 1 point per measurement, demonstrating the ability of recovering and maintain a correct scale along the trajectory with very light and inexpensive off-the-shelf range sensors.

Abstract - Italian

L'utilizzo di agenti robotici porta ad un maggiore ritorno scientifico nel contesto di missioni per esplorazione planetaria grazie alla possibilità di effettuare misure in loco. Ad oggi, l'esplorazione autonoma compete a singoli veicoli non cooperanti come nel caso delle missioni MER Spirit e Opportunity e MSL Curiosity.

Una preziona risorsa per l'autonomia è la possibilità di percepire la struttura dell'ambiente circostante tramite sistemi di visione. Questo problema è tradizionalmente risolto tramite algoritmi di Visual SLAM, che permettono di ricostruire la struttura 3D osservata e di calcolare la posizione di una telecamera rispetto ad essa. Ad oggi, molti algoritmi esistono dedicati a singole sessioni operative o per singoli robot.

Il futuro dell'esplorazione planetaria coinvolge tuttavia la cooperazione tra unità robotiche eterogenee come veicoli terrestri ed aerei. Mentre la Visual SLAM è efficientemente risolta per singole sessioni, mappatura collaborativa e ri-localizzazione per exploratori multipli è ancora argomento di ricerca e costituisce il primo grande obbiettivo di questa tesi.

In questo lavoro si propone e valida sperimentalmente un innovativo algoritmo di ri-localizzazione per rover planetari equipaggiati con stereo camere permettedo di ricominciare sessioni di esplorazione interrotte. Contrariamente alla maggiorità di algoritmi SLAM, basati esclusivamente su informazioni visive, il nostro algoritmo sfrutta la natura invariante delle nuvole di punti usando descrittori binari compatti congiuntamente ad informazioni sulla texture degli oggetti osservati. Le nuvole di punti sono discretizzate in sotto-mappe e rappresentate in maniera compressa usando il paradigma delle Bag-of-Words. L'algoritmo è testato presso il centro di Robotica e Meccatronica dell'Agenzia Spaziale Tedesca (DLR) e validato in un dataset catturato sul Monte Etna, riconosciuto come terreno analogo planetario.

Il secondo obbiettivo di ricerca riguarda tecniche di Visual SLAM monoculare. Spesso non è infatti possibile equipaggiare stereo camere in veicoli aerei di ridotte dimensioni in quanto degenerano verso sistemi monoculari mentre la distanza dal suolo aumenta. Attraverso tecniche di Structure from Motion è possibile ricostruire una struttura 3D da osservazioni monoculari tuttavia mancando un fattore globale di scala. Tradizionalmente, la scala è stimata integrando l'accelerazione lineare da IMU, tuttavia rumore nelle misure ed errori nella calibrazione dei parametri estrinseci de-stabilizzano il sistema di visione inerziale. Questi motivi rendono delicata l'implementazione di comportamenti autonomi su UAV. In questo lavoro di tesi è presentata una possibile soluzione a questo problema. E' proposto un approccio alla sensor fusion tra sistemi di visione monoculari e sensori di distanza per il recupero ed un robusto mantenimento del corretto fattore di scala. Questo obbiettivo è raggiunto in due step. Per prima cosa è sviluppato un algoritmo di Visual Odometry senza alcun requisito sulle performance computazionali. L'algoritmo è abbinato ad un sensore di profondità operante con il principio del Time of Flight (tempo di volo). Le performances sono confrontate con un pesante ed accurato LiDAR a scansione planare ed infine un sistema di stereo Visual SLAM. Infine, un sistema di Visual Odometry monoculare real-time è sviluppato in architettura a thread multipli che permette di calcolare

contemporaneamente la posizione della camera nel tempo e di raffinarla tramite un processo di ottimizzazione. L'algoritmo è testato utilizzando un laser altimetro, che consiste in un sistema minimale di percezione di distanza ritornando un solo punto per misura. Si dimostra cosi di poter mantenere un corretto fattore di scala lungo la traiettoria utilizzando un semplice ed economico sensore di distanza comunemente impiegato per UAV.

Contents

Chapter 1

Introduction

Future prospects for the technological development of space exploration systems strongly involve cooperation within teams of *heterogeneous* robotic agents [38]. Depending on the scientific objectives, collaborating agents can be multiple orbiters, landers, rovers or other in-situ explorers such as aerial or legged robots. Coordinated observations between multiple observers in fact give insight to phenomena in a time-synchronized manner from the perspective of different instruments for a more complete characterization. Such tasks have already been performed between orbiters and rovers, an example is the simultaneous measurement of the Mars atmosphere at different altitudes where the MER Mini-TES (Thermal Emission Spectrometer) observations have been coordinated with the Mars Global Surveyor spectrometer. However this required an intensive manual work by the ground teams in order to synchronize and correlate data. Such way would prove to be infeasible for longer missions and more frequent observations, therefore the need for *autonomous behaviors* is critical in this case. Synchronous operations across ground agents benefit also from the perspective of mapping and navigation. Coordinated exploration allows to maximize coverage [19] [111] and build maps of a previously unknown environment in a fraction of the time required to a single robot. This poses significant challenges to the ability of vision system to merge 3D maps with high accuracy and stream them according to the given communication link [49]. Aerial agents can complement the perceptive reach of ground vehicles equipped with stereo cameras. The higher point of view and enhanced mobility allow in fact an UAV to assess the ground traversability for its ground counterpart [92], speeding up the path planning process substantially and identifying targets of interest from visual cues to be analyzed from the rover scientific equipment once reached.

Although many concepts for robotic cooperation on Earth have been presented, the technology required for performing robustly many of the involved tasks is still an active research topic. Furthermore, to this day no real autonomous behaviors have been implemented in space exploration missions. No aerial vehicle has yet flown on planetary surfaces and the role of the spacecraft in small bodies exploration missions (Rosetta until 2016 with the Philae lander [11] and Hayabusa2 with the MASCOT lander [110]) was mainly of communication relay with the control centers on Earth.

In this thesis are addressed multiple issues challenging the implementation of such collaborative systems respectively for the ground and aerial counterpart whose navigation

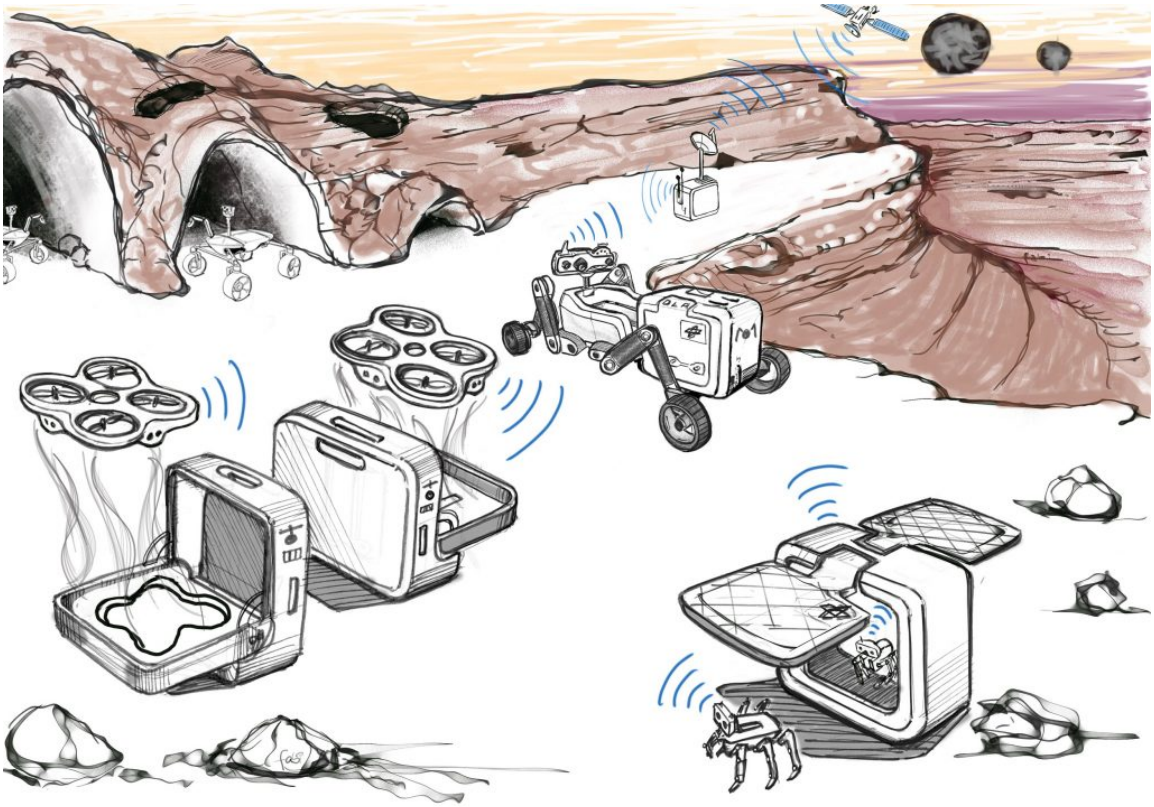


Figure 1.1: Artistic impression of a multi-agent exploration scenarios. Wheeled rovers explore environment in their reach while aerial agents are deployed to observe and map on longer distances. *Credit: German Aerospace Agency (CC-BY 3.0)* `https://event.dlr.de/en/ila2018/arches/`

system heavily relies on stereo and monocular vision systems. In the context of *long-term* operations, all the findings of each robotic explorer must be referenced to a common coordinate system. To this aim, the 3D maps in form of *point clouds*, built by the robots through their perception system, must be joined between each robot and across each exploration session. Accomplishing this task is challenging and the reason is twofold:

- The task of *place recognition*, although frequently addressed in the contexts such as autonomous driving, is particularly difficult while navigating in natural environments, where the textures and shapes that can be observed are usually very repetitive or lacking significant information. In addition to that, while rovers on Earth and even heavy UAVs can be equipped with precise LiDAR systems to measure shapes, to this date no planetary rover has been equipped with such sensors and rely instead on cameras to perceive depth and perform Visual SLAM (Simultaneous Localization and Mapping)

- Place recognition tasks as well as point cloud processing have a significant impact on *computational load* even for workstation grade computers. The performances of SLAM algorithms are in fact observed to degrade while imposing hardware constraints [50].

This leads to the first major contribution of this work, which is the development of a fast relocalization scheme for teams of heterogeneous robots equipped with stereo cameras as the main perceptive sensor. The work was developed in collaboration with the DLR Center for Robotics and Mechatronics and allows localization on a 3D map built in previous exploration sessions by the same robot or others. The pipeline was developed with the aim of providing a compressed representation of captured 3D data which is efficient to store on the on-board memory and stream to other robots and allows a very fast similarity evaluation with a database for relocalization. This first major contribution can be split in two minor ones, which are:

- The adaptation of a well-known paradigm for computing similarity across documents called the *Bag of Words* model to the 3D case. To the knowledge of the author of this work, this is the first time this paradigm is applied to the context of place recognition using *point clouds* and 3D *binary descriptors*

- The development of a fast and lightweight *incremental voting scheme* in alternative to commonly used RANSAC to select a rototranslation model between point clouds in presence of very high percentages of outliers.

In chapter 3 the aforementioned pipeline is explained in detail and tested on multiple mapping sessions and with multi-robot teams.

Ground robots have less mass and power constraints compared to UAVs therefore can easily carry equipment such as wide baseline stereo cameras. This allow them to obtain high precision point clouds in the close range for both precise mapping and ego-motion estimation. UAVs have more strict mass and power budgets and therefore it is less preferable to mount redundant systems such as multiple cameras. While in principle it is possible to

have more than one cameras to capture frames for later processing, trajectory estimation must be performed at high frequencies to close path following loops. In addition, the measurement accuracy of stereo vision systems degrade with the distance squared therefore for longer viewing distances all stereo setups degenerate to a monocular one. For this reason mainly monocular vision setups are used in the context of UAV ego-motion estimation, as it will be for the Mars Helicopter Scout, part of the NASA Mars 2020 mission [7]. However, monocular estimations provide 3D and trajectory information only up to an unknown *scale factor* which must be resolved in conjunction with additional information. Many monocular vision systems make use of IMU sensors for retrieving the absolute scale. However the initialization of such visual-inertial algorithms is delicate and many low power and weight sensors are characterized by excessive measurement noise, with an impact of the ego-motion estimation accuracy.

This leads to the second major contribution of this work, which is the development of a monocular Visual Odometry system where the absolute scale is retrieved using generic low resolution range sensors. This system allows a direct and robust estimation of the metric scale during initialization of the algorithm and adjusts the scale factor along the trajectory by means of an optimization back-end. The development of this algorithm is split in two phases. In chapter 4 it is evaluated the feasibility of such approach with a non real-time algorithm based on slow and accurate image feature matching. This algorithm is tested with a very low resolution Time of Flight camera of 8x8 pixels delivering range information. In chapter 5 a final version of the pipeline is presented where range measurements are provided by a 1D LiDAR range finder delivering only 1 range estimate along its measurement axis. Both versions of the algorithm are tested on ad-hoc experimental setups on Differential GPS ground truth. To summarize the following contributions can be highlighted:

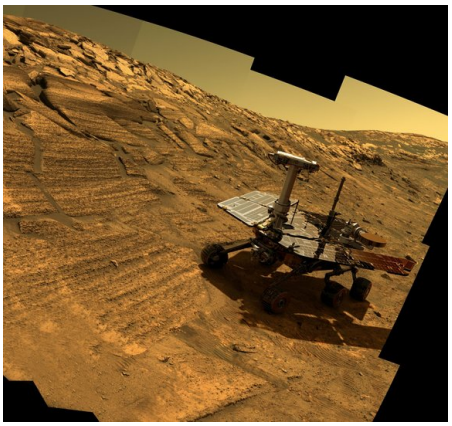
- It is here presented a sensor fusion approach for monocular Visual Odometry systems where *direct* scale information is given by a *minimal* sensing setup delivering a single range measurement. In such way, the benefit of accurate and immediate range measurements can be exploited with low mass, power and volume sensor setups suitable for small UAV navigation.

- It is proven that the performances that can be achieved by such setups are on-par with heavier and more requiring stereo vision setups.

- In order to test the algorithms on real setups, unique extrinsic calibration techniques are presented to express range measurements in camera coordinates. While in literature several example of extrinsic calibration algorithms are present, they usually refer to camera-LiDAR systems where the LiDAR is planar (2D) or 3D.

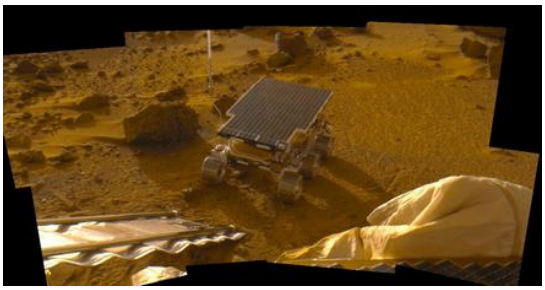
1.1 Brief History of Mars Rover Missions

The first autonomous robotic explorer on Mars was the Sojourner rover, deployed by the Pathfinder spacecraft which landed in the Ares Vallis region on July 4th 1997. Sojourner's primary objectives involved capturing data with the Alpha Proton X-ray Spectrometer (APXS) to analyze the chemical composition of rocks and minerals. In addition, the rover



(a) Opportunity

(b) Curiosity



(c) Sojourner

(d) Mockups of each rover compared

Figure 1.2: Self portraits of NASA's MER (Mars Exploration Rover), (Mars Science Laboratory) Curiosity and the Sojourner rovers. *Credit: NASA Jet Propulsion Laboratory* `https://jpl.nasa.gov`

carried three cameras with the main purposes of observing the condition of the lander, assess the soil mechanics by observing the deformation induced by the wheels and observe surfaces with higher resolution than the lander [89]. The rover was powered by six wheels, of which 4 steerable, and connected to the chassis by a *rocker-bogie* system which allowed to climb over obstacles the approximate size of the wheels. During motion, dead reckoning from wheel encoders and gyroscope readings provided an estimate of the trajectory. During stationary moments, a stripe laser projected light on the terrain in front of the rover such that through the camera system was possible to observe the presence of obstacles. After finding a viable path towards a location of interest, the rover was able to travel autonomously. Images captured before and after motion were transmitted back to Earth and used to correct trajectory estimations.

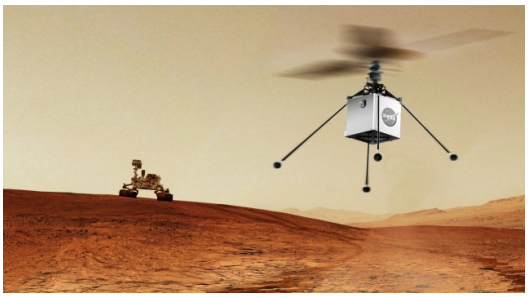
It is from the Rocky 7 research prototype [145] that stereo vision is used for autonomous navigation. Instead of manual localization from an operator on Earth as for Sojourner, the lander observed a colored cylinder mounted on top of the rover. This position measurement allowed for a global positioning with respect to the lander without any human intervention and avoiding so to accumulate errors from gyroscope biases and wheel slippage [144]. Absolute heading was instead measured using a top mounted sun sensor in addition to accelerometer readings.

1.1.1 Mars Exploration Rovers (MER)

In 2004 NASA's Mars Exploration Rovers Spirit and Opportunity (figure 1.2a) landed on the surface of Mars in order to observe and characterize the soil to find cues on past presence of water, necessary factor for life as it is known. Spirit landed on the *Gusev* crater, supposed to be originated from a water lake and later discovered to be the result of impacts and wind erosion [131]. Opportunity landed 21 days later on the *Eagle* crater within Meridiani Planum with the same purpose, demonstrating past water presence through visible geological cues and chemical analysis of rock composition [130]. With the Mars Exploration Rovers is firstly implemented full on-board stereo vision processing. Significant hardware and software upgrades allowed the rovers to conduct autonomous navigation tasks such as pose estimation through *Visual Odometry*, safe terrain navigation via hazard detection and global path planning. The two rovers carried an identical set of instruments comprising two stereo camera setups. The Pancam was a 30 cm baseline setup with 16 degrees field of view which was used for mapping of far objects while the Navcam, a 20 cm baseline stereo camera with a wider field of view of 45 degrees, was primarily used for close observations. The primary task for the stereo cameras was automatic terrain assessment. Stereo images were rectified to remove lens distortion and used to generate 256x256 pixels disparity images form which 48000 3D points in average could be reconstructed. The GESTALT (Grid-based Estimation of Surface Traversability Applied to Local Terrain) system for terrain assessment allowed Spirit to travel for 2 km in a 6 months period towards the Columbia Hills approximately 50% faster than with only blindly followed planned paths. Visual Odometry, originally conceived as an "extra feature", was used in case of rough and high slip terrains to correct for wheel slippage. However, wheel odometry was preferred while traversing over low slip terrains due to the high computational cost of image processing, which was performed on the onboard CPU. The Pancam stereo camera delivered was used also for absolute orientation sensing by detecting the position of the Sun in conjunction with tilt measurements from the accelerometers [86].

1.1.2 Mars Science Laboratory (MSL)

The MSL (Mars Science Laboratory) Curiosity (figure 1.2b) was launched in 2011 to explore the *Gale* crater with the objectives of investigating Mars geology and climate. The focus was on evaluating the habitability of the planet surface to assess the feasibility of future human exploration missions [14]. Curiosity is still operational to this day after more than 2500 martian sols of operations and features 8 Hazard avoidance cameras and 4 Navigation cameras, for a total of 12 engineering cameras with a design similar to the MER cameras [87]. The objective of this camera system is to assist all MSL operations on Mars, from assessing the terrain traversability to detect and avoid hazards, operate the robotic arm and localize the rover with respect to the environment. The Navcam setups are configured as a 42 cm baseline stereo system providing range measurements up to 100 meters while the Hazcams have a 10 cm and 16 cm baselines respectively for the front and back ones and a wide field of view of 120 degrees. As the MER rovers, Curiosity performs stereo Visual Odometry, uses both cameras and IMUs to measure its attitude and is able to plan



(a) Rover and Helicopter

(b) Helicopter

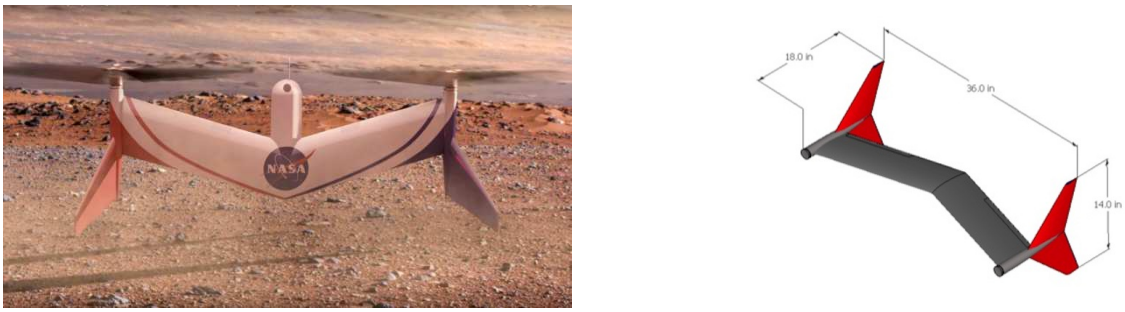
Figure 1.3: Artistic impression and real model of the rover-drone team part of NASA's Mars2020 mission. *Credit: NASA Jet Propulsion Laboratory*

paths avoiding immediate hazards and non traversable regions. Many improvements to the previous VO software [69] together with more powerful hardware allows Curiosity to use VO for longer sequences spanning multiple sols and activate it autonomously while detecting potential high slippage surfaces.

1.2 Future Prospects for Multi-Agent Exploration

1.2.1 Mars Helicopter Scout

In the context of NASA's Mars 2020 mission, a small helicopter will be deployed by the rover to demonstrate the feasibility of unmanned flight in the Mars atmosphere [7]. Figure 1.3b shows the 850 grams vehicle characterized by a co-axial rotor of 1.21 meters in diameter which will enable a sustained autonomous flight in the Mars atmosphere whose density amounts approximately at the 1% of Earth's atmosphere. Once deployed on the Mars surface, the rover will drive away from the helicopter stopping at a safe distance and relaying commands from Earth. Then, a series of *autonomous flight* tests will evaluate the vehicle capability of accomplishing navigation and exploration tasks without any intervention. The test campaign will involve a series of flights of increasing distance, altitude with durations up to 90 seconds [18]. As the vehicle is developed as a technological demonstrator, the navigation hardware makes use almost entirely of Commercial Off-the-Shelf components. In order to follow the predefined path in an autonomous way it is required to perform Visual Odometry in real-time which is a computationally expensive task. The selected CPU is in fact a smartphone-grade 2.26 GHz Snapdragon 801 quad-core processor accompanied by 2 GB RAM memory and 32 GB flash storage running a custom Linux kernel. Flight and attitude control is performed instead by a redundant system of Microcontroller Units which can be hot-swapped in case of faults. The navigation sensor setup also comprises off the shelf components. Two cameras are mounted on the helicopter, a navigation camera (NAV) used for visual tracking and a high resolution camera for capturing detailed pictures to send back to Earth. The Navigation Camera is a global shutter VGA (640x480 pixel) sensor with a field-of-view (FoV) of 133x100 degrees (horizontal and vertical) capturing frames at a rate of 10 Hz. The high resolution camera is referred to as Return-to-Earth (RTE) and is a color rolling shutter 4208x3120 pixels sensor with a 47x47 degrees field of view. The



(a) Artistic impression (b) ciccio

Figure 1.4: The Mars Electric Reusable Flyer prototype *Credit: NASA Langley*

two cameras are mounted with very little baseline given the size constraints and their field of view overlap slightly. This can be a useful for later processing at Earth since it allows to perform feature matching between the NAV and RTE cameras. Attitude and inertial measurements are provided by a 3-axis MEMS IMU produced by Bosch ((Sensortec BMI-160) and the altitude is measured by a 1D LiDAR rangefinder from Garmin, the Lidar-Lite V3 which has a maximum range of 10 meters, a 1 cm resolution and a mean accuracy of 1% of the maximum range.

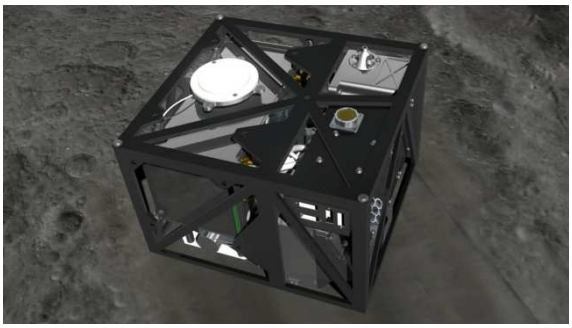
1.2.2 Mars Electric Reusable Flyer

Following the idea of a deployable UAV to assist and enhance rover operations, a concept for a Vertical Take Off and Landing (VTOL) aircraft is in testing phase at NASA Langley[1]. The vehicle, visible in figure 1.4, would allow safe take off and landing for recharging as well as fast forward flight for wide terrain coverage. An engineering model has been tested in the Spring of 2017 on a stratospheric balloon drop to gain insight on the aerodynamics. The hovering capabilities have been tested furthermore in a low pressure chamber as for the Mars Helicopter Scout. Research on this concept is on-going especially regarding vision based autonomous navigation and hovering in complex scenarios such as canyons and lava tubes.

1.2.3 Small Bodies Exploration

In October 3 2018 the MASCOT (Mobile Asteroid Surface Scout) lander touched down on the surface of 162173 Ryugu, a near-Earth asteroid orbiting between the Earth and Mars. MASCOT (visible in figure 1.5a) is part of JASA's Hayabusa2 sample-return mission launched in 2014 whose scientific objective is to study the asteroid composition in search of interactions between ice, minerals and organic compounds. Study of asteroids and comets provides insight on the origin of planets, water and organic life. MASCOT, developed by the German Aerospace Center (DLR) carries a camera (MasCam), and an hyperspectral microscope (MMARA) [76] enhancing so the remote sensing capabilities of the Hayabusa2 spacecraft on multiple scales. MASCOT's mobility results from rotating a

[1]https://catalog.data.gov/dataset/mars-electric-reusable-flyer



(a) MASCOT

(b) MMX Lander

Figure 1.5: Renders of the MASCOT and MMX landers *Credit: German Aerospace Agency (CC-BY 3.0), CNES*

swing arm contained inside the lander body to generate tumbling motions. Precise commands sequences allows the body to stand on its sides exposing the instruments to the location of interest, further details on the motion mechanism can be found in [124]. The MasCam is mounted slightly inclined towards the ground with an angle of 22 degrees such that the field of view of 54.8 degrees covers both the close ground as well as the horizon [119]. The MasCam operated both during day and night with the help of an array of monochromatic LEDs illuminating in 4 different spectral bands allowing to identify variations in the surface spectrum. Amongst other findings, high resolution measurements of the asteroid surface temperature by the MARA radiometer allowed to explain the lack of rocks belonging to Ryugu-like asteroids on Earth, being the surface of the asteroid more fragile and brittle than expected [55].

Following the success of Hayabusa2, a new JAXA/DLR/CNES mission called MMX (Martian Moon eXplorer) will launch in early 2020 to explore the Martian moon Phobos [20]. The primary scientific objective of the mission is to observe the composition of Martian moons, which would give insight on their origin and to answer questions related to planetary formation and transport of matter between the inner and outer parts of the solar system. The mission is scheduled to launch in 2024 and return with samples from Phobos in 2027 and comprises three main components: a propulsion stage, an exploration module and a return module. The rover module will be equipped with a vision system for navigation and localization and will carry a scientific payload comprising a spectrometer and a radiometer for surface analysis.

Chapter 2

Preliminaries

This chapter introduces some preliminary notions about how cameras are treated as sensors capable of perceiving depth, constructing 3D maps of an environment and compute their position in it. More specifically, both monocular and stereo perception systems are addressed with reference to the chapters that follow.

2.1 Pinhole Camera Model

Mathematically, a camera is a *mapping* from $\mathbb{R}^3$, the 3D world, to $\mathbb{R}^2$ or the image plane, where the sensor resides. The *pinhole* model is the simplest for a camera and describes the perception of 3D points through their central projection onto a plane. Let be a 3D point be denoted by $\mathbf{X} = \{x, y, z\}$. Its central projection is the intersection between the line that connects it to the *camera center* and the image plane. With reference to figure 2.1b, the coordinates of the projection of $\mathbf{X}$ are:

$$\mathbf{x} = \left[f\frac{X}{Z}, f\frac{Y}{Z} \right] \tag{2.1}$$

where the parameter f is the camera *focal length*. However, if the origin of the image coordinate system is not located at the principal point, eq. 2.1 becomes:

$$\mathbf{x} = \left[f\frac{X}{Z} + u, f\frac{Y}{Z} + v \right] \tag{2.2}$$

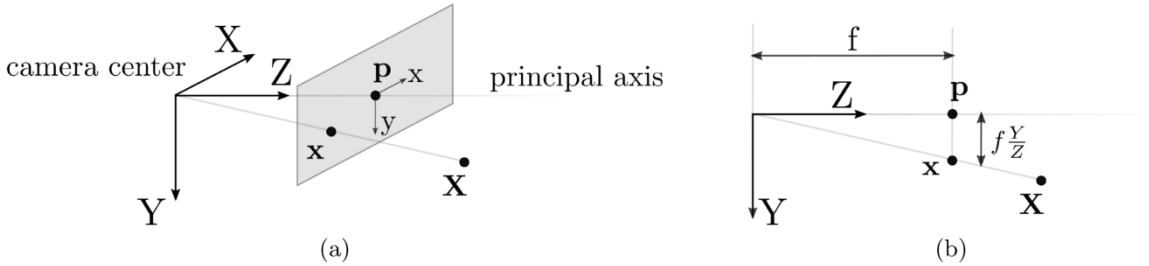


Figure 2.1: (a) Graphical representation of the pinhole model. **p** is the principal point, at the intersection between the image plane and the principal axis. (b) side view of (a), f is the focal length.

which can be expressed in matrix form as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fS_x & 0 & c_x \\ 0 & fS_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X/Z \\ Y/Z \\ 1 \end{pmatrix} \tag{2.3}$$

where u, v are expressed in pixel coordinates as well as c_x, c_y and the coefficients S_x, S_y transform the focal length in pixels. The 3x3 matrix in eq. 2.3 is called *camera matrix*, its parameters are fixed for each camera and can be determined through calibration [134][157]. Generally, the position of points and cameras are expressed in a common coordinate system which is denoted as *world*. In this case, the mapping from points in world coordinates to the image plane is function of the rototranslation from world to camera. Eq. 2.3, express in *homogeneous coordinates* becomes:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} fS_x & 0 & c_x \\ 0 & fS_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.4}$$

Lens Distortion

Real cameras do not behave accordingly to the linear pinhole model. Their lens introduce optical distortions, which can be modeled to convert the distorted image to its linear counterpart allowing to use the pinhole model. A distortion model applicable to many commercial cameras combines radial and tangential distortions. The radial contribution is induced by lens distortion while the tangential part is due to the image sensor being not parallel to the image plane. The relation between distorted and non-distorted pixel coordinates is:

$$\begin{aligned} u_d &= u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 uv + p_2(r^2 + 2u^2) \\ v_d &= v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 uv + p_1(r^2 + 2v^2) \end{aligned} \tag{2.5}$$

where u_d, v_d are *distorted* pixel coordinates, k_1, k_2, k_3 are radial distortion parameters, p_1, p_2 are tangential distortion parameters and $r^2 = u^2 + v^2$.

2.2 Image Features

The previous section describes the relationship between 3D points and their projection in the image plane. Many machine vision algorithm however start from detecting salient positions in the image and subsequently determine the 3D position of the geometric point to which they correspond. Visual *features* are locations which hold valuable information for characterizing the content of an image. Feature *detectors* are algorithm that recognize features amongst images in the most repeatable way possible and are usually sensible to specific properties.

Corner detectors such as Moravec [95], Harris [61], Shi-Tomasi [125] or FAST [112]

measure the location of intersection of edges in an image with sub-pixel accuracy. Corners are easy to detect and generally fast to compute, however they do not excel in localization accuracy and repeatability, meaning that subsequent detections changing the viewpoint and scale might happen in slightly different locations. Harris and Shi-Tomasi corners are detected by analyzing the image partial derivatives in the u and v coordinates. The partial derivatives I_x and I_y are used to compute the second-moment matrix of I. Its eigenvalues define a response function which dictates if the image patch under consideration is a corner or not. The FAST corner detector instead declares corners those image patch which pass a machine-learned circular test around the patch center. Corner detectors are very suitable for artificial environments, however if the scene does not contain enough edges, *blob* detectors are a viable and more computationally expensive asset. Blobs can be interpreted as regions where a property derived from intensity values remains locally constant and two main detection approaches can be distinguished. The first approach, denoted as *difference of Gaussians* or DoG relies on computing the difference of the same image convolved by two different Gaussian kernels. This step enhances those regions of the image which can be reliably detected at different scales and approximates the Laplacian operator over I. This detector is used by SIFT [84] (Scale Invariant Feature Transform). A second approach is denoted *determinant of Hessian*, where the determinant of the Hessian matrix is computed over multiple scales detecting the extrema. The resulting locations are invariant on scale and rotation. The SURF [8] (Speeded Up Robust Features) detector is based on this principle and computed the determinant of the Hessian in an approximated form using Haar wavelets. Both corner and blob detectors, after computing their specific response function over the image, perform *non maximum suppression* to select only the highest scoring features for the best detection repeatability.

Feature *descriptors* convert the neighborhood of each image feature in a compact descriptor in a compromise between invariance to illumination, scale and viewpoint and computational speed. Simply computing the difference between image patches of fixed size is not in fact invariant to any of the aforementioned factors. The SIFT descriptor is one of the most accurate amongst all *handcrafted* feature descriptors being invariant to scale, illumination, rotation and limited affine transformations. At keypoint detection time, SIFT stores magnitude and orientation of the smoothed image gradient. Those are used to scale and orient a grid centered on each keypoint which define a relative spatial domain. In this domain, orientations and magnitude of gradients are collected in an histogram which constitute the final descriptor. Similarly to SIFT, SURF descriptors are computed by aligning a spatial grid over keypoints given reference scales and orientations. The descriptor entries are derived from Haar wavelet responses computed over subdivisions of the grid. The choice between SURF and SIFT is design driven, while the former maintain a good invariance while being relatively light, the latter achieves usually the highest matching performances at the price of a higher computational effort.

Most recently, deep convolutional neural networks (CNN) have been shown to be a viable option for learning both keypoint detection and description. The earlier DeepDesc [127] and TFeat [94] rely on external keypoints, which in the comparative test given in [120] are computed from SIFT. LIFT [153] instead performs jointly detection and description.

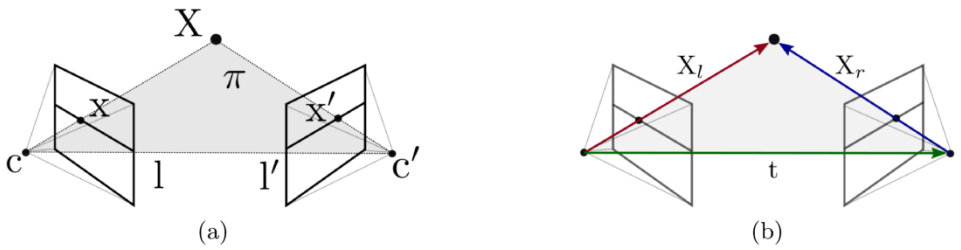


Figure 2.2: Epipolar geometry. c and c' are the camera optical centers. π is the epipolar plane comprising X, c and c' while l and l' are epipolar lines. The line between c and c' is referred to as *baseline* and its intersections with the image planes are called *epipoles*

TILDE [142] and [156] are instead examples of only keypoint detector. In [81] is provided a large scale evaluation of more recent feature descriptors for image matching.

2.3 Multi-view Geometry

Monocular depth perception can be achieved by multiple views taken from different locations. The process of reconstructing simultaneously the 3D structure of an environment from successive images is generally referred to as Structure from Motion (SfM). The projective geometry across two different views is called *epipolar geometry*. Figure 2.2 shows the relationship between the 3D point X with the two cameras (the usage of the term *camera* can denote a single sensor moved in two positions as well as two physical sensors). The point X is coplanar with the optical centers c and c' and its projections x and x' lie on the respective epipolar lines l and l'. The projections x and x' can be found in real images from *feature matching*.

2.3.1 Fundamental Matrix

Let be $t = c' - c$ the translation vector between the two optical centers and R the rotation matrix between the two camera reference frames. Let also be X_r and X_l the coordinates of X expressed in the right and left camera frames and from this follows that $X_r = R(X_l - t)$. As the cross product between X_l and t is always orthogonal to both of them, the following equation always stands:

$$(X_l - t)^T \underbrace{R^T R}_{I} t \wedge X_l = 0 \tag{2.6}$$

Rearranging eq. 2.6 using the relationship between X_r and X_l and expressing $t\wedge$ in matrix form S leads to the following equation

$$X_r^T RSX_l = 0 \tag{2.7}$$

where $RS = E$, the *essential matrix*. Equation 2.7 stands also for X_r and X_l in pixel coordinates though the respective projection matrix, which leads to the following equation, where F is the *fundamental matrix*:

$$x^T F x' = 0 \tag{2.8}$$

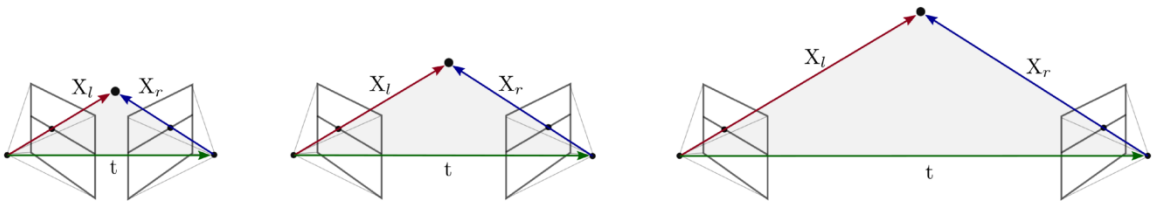


Figure 2.3: *Scale* is a free parameter in Multiple View Geometry from monocular (extrinsically uncalibrated) observations. As can be observed in this figure, the projection of X onto each camera image plane is always the same for each scale value.

Both E and F matrices are rank deficient, in fact their rank is 2. Compared to E, F depends on the additional camera *intrinsics* parameters which can be known from calibration. In this case, the two view geometry of figure 2.2 can be obtained solving eq 2.7 for E using Nister's *Five-Point* algorithm [101]. While E has 6 non-zero parameters, from its rank deficiency only 5 of them are independent, hence the 5 point correspondences required. Using singular value decomposition (SVD), E can be decomposed into the rotation and translation part. However as only 5 degrees of freedom can be recovered out of 6, scale information is **lost**.

Similarly, for non intrinsically calibrated cameras, an 8 point algorithm [63] can be used to estimate directly the fundamental matrix F.

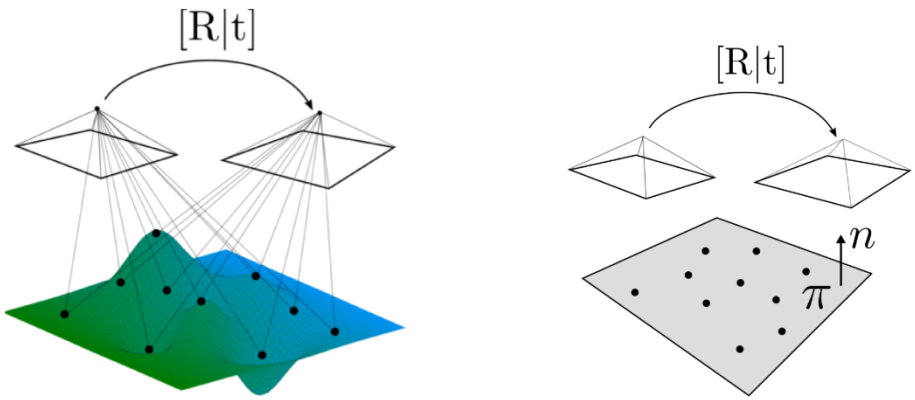
Degeneracies

Estimating the fundamental matrix F can lead to degeneracies for specific spatial configurations of 3D points and cameras where exist multiple solutions for F. In [140] an exhaustive review is given of the sources of degeneracies for the fundamental matrix with respect to the dimension of its null space. If 8 points are collocated in generic positions in front of the cameras, the null space of of F should have dimension 1, representing the scale ambiguity. However, if all points are *co-planar* or in case the camera is performing a *rotation-only* motion, the null space dimension is 3, leading to a degenerate configuration. Higher null space dimensions can be achieved if all points are arranged on a line or on a single point. This leads to important design configurations for all monocular estimations: fundamental matrices can be extracted in case of generic motions and point configurations but in case of a camera observing a flat plane, such as the ground plane, must be treated in different ways.

2.3.2 Homographies

Homographies are isomorphisms of projective spaces and are used to describe the relationship amongst projections of 3D points laying on a planar surface on multiple cameras. Being x and x' matching image features related to the 3D point X, it can be demostrated [9] that the following equation is true:

$$x = \frac{Z'}{Z} K H K^{-1} x' \tag{2.9}$$



(a) Non planar scene (b) Planar scene

Figure 2.4: Two view geometries with non-planar and planar scenes, the first is best described from estimating the fundamental matrix F while the second is best estimated from an homography.

where Z and Z' are the coordinates of X along the optical axis of each camera, K is the intrinsic matrix and H is an homography matrix. H encapsulates information about rotation and translation as it is given by:

$$H = R - \frac{tn^T}{d} \tag{2.10}$$

where n is the normal plane vector and d is the distance between the plane and the camera optical center c. Equation 2.9 can be solved for H knowing K by a Direct Linear Transform algorithm (DLT). As for the fundamental matrix, 8 point pairs are required since H is again rank deficient. Rotation and translation can be retrieved from H through Singular Value Decomposition (SVD) using the Faugeras method described in [40]. This method returns 8 different configurations for R and t from which the correct one must be identified imposing the *chierality* constraint: all 3D points must lie in front of both cameras.

2.3.3 Triangulation

Once the geometry of the two views is known, the 3 coordinates describing the point X can be computed from *triangulation*. This step concludes the process of perceiving depth from monocular cameras moving in space. Given the position and orientation of the two cameras it is possible to determine the *projection* matrices P and P' as $P = K[R|t]$. The projection matrix relates pixel to 3D coordinates using vectors in homogeneous form. For each point X_i it is possible to write the following equations: $x_i = PX_i$ and $x'_i = PX'_i$. Computing the cross product $x \wedge PX = 0$ removes the homogeneous scale factor and returns 3 equations of which 2 independent. Combining two of them with other two from $x' \wedge PX' = 0$ returns

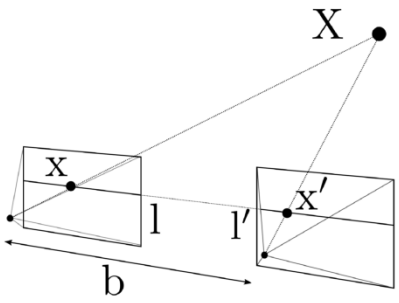


Figure 2.5: Two cameras arranged in a lateral stereo configuration. b is the *baseline* between the optical centers. The epipolar lines l and l' are coincident and parallel in each camera for all points X.

the following system in matricial form $AX = 0$ where A:

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \tag{2.11}$$

The resulting values for X can be further refined with Levenberg-Marquardt.

2.4 Stereo Matching

Stereo vision systems comprise two cameras whose relative configuration is fixed and known. This knowledge allows to just perform feature matching and triangulation to reconstruct the observed environment with the correct scale. In principle in fact, stereo vision setups might be thought as a generic multi-camera setups and handled with conventional multiple view geometry. However, the fact that cameras are usually arranged in a left-right configuration leads to significant simplifications. Figure 2.5 shows an ideal stereo vision setup where the two cameras are only displaced laterally, i.e. in camera coordinates $R = I$ and $t = [t_x, 0, 0]$. The resulting epipolar lines are *parallel*. This means that for all 3D landmarks X_i, given the projections x_i in the left frame, the matching image *feature* lies on the same vertical coordinate on the right image. The search for matching features in this case reduces from 2D (whole image) to 1D (single line). As the left and right cameras are close to each other (10cm - 0.5m) for usual stereo setups, the perspective distortions are minimal. Given the image location x, the matching x' in the right image can be detected using simple distance metrics such as SSD (Sum of Square Distances) or SAD (Sum Absolute Distances) instead of complex handcrafted descriptors.

In the real world, the geometry in figure 2.5 is only an approximation of what a stereo system might be: misplacement of both cameras and sensors result in the epipolar lines to not be parallel anymore. The search for feature matches is then suboptimal. This problem is addressed through calibration [157], which returns the extrinsic parameters $[R|t]$ between the two cameras. The left and right images can then be *mapped* to an ideal stereo setup and used to perform *dense* stereo matching.

2.4.1 Depth from stereo

The simple geometry of an ideal stereo setup lead to a simplified triangulation of matching image features x and x'. Instead of the complex linear approach of section 2.3.3, the 3D coordinates of X are obtained as follows:

$$\begin{aligned} Z &= (b \cdot f)/(u - u') \\ X &= (u \cdot Z)/f \\ Y &= (v \cdot Z)/f \end{aligned} \tag{2.12}$$

where b is the baseline, or the distance between the optical centers, and u, v are horizontal and vertical pixel coordinates. This process requires minimal effort to compute and can easily be parallelized for efficiency. Equation 2.12 requires to know correspondences between features in the left and right images. Dense stereo matching algorithms allow to perform this task on a *per-pixel* basis. Amongst all it is summarized here the Semi-Global Block Matching (SGBM) algorithm [64], which constitute the basis of stereo depth perception in chapter 3.

SGBM

The Semi Global Matching algorithm, also referred to as Semi-Global Block Matching in the OpenCV implementation [1], computes a *disparity* image exploiting Mutual Information [143] instead of intensities and approximates a global 2D smoothness constraint as a combination of discrete 1D constraints over multiple paths across local image patches. A disparity image is defined as

$$D(u, v) = u - u'\big|_v \qquad 0 \leq u \leq \text{cols}(I) \tag{2.13}$$

where u' is the horizontal pixel coordinate in the right image $I'(u', v)$ where the intensities match the ones in $I(u, v)$.

The cost function to minimize is denoted $L(x, d)$ where x is the point in the left image and d the disparity. The first contribution to L is the pixel-wise matching cost $C(x, d)$ which, instead of being simply defined as a difference of local intensities from I' and I, is computed as a function of the mutual information between the two in a local window. Mutual information, as discussed in [143] provides an invariant quantity to illumination. On a side note, the well-known implementation of the function in OpenCV substitutes this step with a simpler intensity difference using the Birchfield-Tomasi metric [12].

A second contribution to the cost function is proportional to the difference between the disparities in a neighborhood of x and the disparity of x. This constraint penalizes regions where the disparity changes rapidly, which is likely due to noise or ambiguous costs computed as part of C. Instead of aggregating this part of the cost in just the horizontal direction, 16 paths are defined crossing the position of x.

[1] https://docs.opencv.org

Chapter 3

Relocalization on Submaps using Bags of Binary Words

3.1 Introduction

A fundamental requirement for long-term exploration of planetary environments is the ability of robotic agents to localize themselves across many consecutive mapping sessions. While the Simultaneous Localization and Mapping (SLAM) problem for single agents equipped with stereo cameras is solved efficiently from many state of the art algorithms, *relocalization* and *loop closure* detection across different mapping sessions with changing viewpoints and lighting conditions is still argument of research. In GPS-denied environments in fact no global localization is available and a robot can only determine its ego-motion referring to an arbitrarily located reference system. In order to refer findings and measurements to a shared map, heterogeneous robotic teams or individual agents over multiple sessions, must be able to localize themselves on a common ground. The task of place recognition using stereo camera systems is traditionally tackled as a similarity search across a set of captured frames, therefore exploiting images instead of 3D structure. Images are in fact a rich source of information and similarity can be detected by matching feature descriptors in a large database through efficient search strategies [31, 30, 97]. Other approaches, following the SeqSLAM paradigm [93, 126, 54], are instead applied at image sequences captured while repeating the same trajectories. In this way, compressed images are matched in limited-size time windows to limit the search effort. Nevertheless, the effectiveness of visual place recognition is undermined in presence of lighting and viewpoint changes as many approaches are difficult to generalize with respect to both of the aforementioned factors. The intuition on which this work builds upon is that, while the environment *appearance* is always subject to changes depending on light source, atmospheric conditions and viewpoint, the *structure* of the environment is usually invariant. Stereo vision systems, a common choice as principal perception system for planetary rovers, allow to measure *statically* the 3D structure of the observed environment. This information can be used to describe the properties of local explored regions to recall them in a robust and consistent way across multiple mapping sessions or from heterogeneous teams of robots. Ignoring the effect of reconstruction noise, 3D information is indeed invariant on the type of structure sensor employed. Relocalization

on point clouds is drawing increasing attention recently also in the autonomous driving field of research, where 3D LiDAR clouds are usually employed. As an example, the authors of [35] segment LiDAR clouds in individual objects such as cars or buildings and associate to them compact and invariant learned descriptors for fast recall. Works such as [29, 72] build global descriptors for single 3D scans while works as [56] exploits local 3D feature descriptors. However, in the context of space missions stereo cameras as preferred to LiDAR systems for their mechanical simplicity as they do not have intrinsic moving parts. This comes with significant consequences as the triangulation accuracy of stereo cameras limits the range at which the 3D structure can be reconstructed without moving the observer. Secondly, while 3D LiDARs provide a complete representation of the environment having usually a 360 degrees lateral Field of View (FoV), exploration is needed with stereo cameras to build a complete representation of local shapes and features.

For all these reasons, in this chapter is presented a relocalization pipeline based on point clouds. The environment is discretized in *submaps* of moderate extent, referred to a global frame through the robot pose at the instant of triggering their creation. Each submap, aggregated from stereo data, contains 3D structural information which is described and compressed through binarized 3D descriptors. With the help of the Bag of Words paradigm, submaps are treated as containers of descriptors, which are transformed in Bag of Words vectors for fast recalling during search of loop closures. In order to combine the information between structure and appearance, binarized 3D descriptors are enriched with texture information by appending a short hand-crafted texture descriptor of limited size inspired by Local Binary Patterns [1]. Finally, submap correspondences selected from different mapping sessions are validated by a novel incremental voting scheme which demonstrates superior robustness to outliers with respect to standard RANSAC approaches. This novel relocalization pipeline is tested both on indoor datasets replicating natural features and on outdoor sequences captured on Mount Etna, designated as a planetary analogous environment.

3.2 Local 3D feature descriptors

Point clouds contain information about the structure of an environment through a discrete set of 3D points which, if triangulated from RGB cameras, contain also color or grayscale intensity information. An efficient way of characterizing the content of a point cloud is through *feature descriptors* which contain local structure information in various form. This is the basis for the relocalization pipeline proposed in this chapter. Here follows an overview of the existing descriptors and a brief discussion on the choice for SHOT [139] amongst all.

Among the most acknowledged descriptors for 3D point clouds, it can be cited firstly the **Intrinsic Shape Signatures** descriptor [158], which also sets the basis for the SHOT descriptor. ISS aims to compute a repeatable local signature of a point cloud by counting the occupancy of a spherical region oriented around a *keypoint*. The first step is then, given a keypoint around which compute the descriptor, define a local reference frame to ensure viewpoint invariance. Being p_i the 3D coordinates of a point belonging to the pointcloud, and p_j each other point in the cloud at a distance lower than r_{frame} from p, the axis of the

reference frame are aligned to the eigenvectors of the covariance matrix M:

$$M(\mathbf{p}_i) = \frac{\sum\limits_{|\mathbf{p}_i - \mathbf{p}_j < r_{frame}|} w_j (\mathbf{p_j} - \mathbf{p_i})(\mathbf{p_j} - \mathbf{p_i})^T}{\sum\limits_{|\mathbf{p}_i - \mathbf{p}_j < r_{frame}|} w_j} \tag{3.1}$$

This step outputs the directions of a reference system which are repeatable but ambiguous in the decision of their direction. After choosing the z-axis direction as the eigenvector with lowest eigenvalues and imposing the right-hand rule for the directions of x and y, there are 4 possible orientations left for the reference system. A spherical region of radius r_{frame} is partitioned in the polar coordinates system to achieve robustness to rotational errors in the reference frame definition. The resulting partitions are designed to have equal solid angle (see figure 3.1). An occupational histogram is built by counting the presence of points in each angular partition. Since 4 possible reference frames can be obtained by aforementioned process, an histogram is create for each of them, resulting in 4 histograms per keypoint. At matching time, the similarity between descriptors is evaluated using a χ^2 test. Being $d_i = \{d_0, d_1, ..., d_K\}_i$ and $d_j = \{d_0, d_1, ..., d_K\}_j$ two descriptors relative to points p_i and p_j, their similarity is computed as:

$$dist(d_i, d_j) = \sum_{k=0}^{K} \frac{(d_{i,k} - d_{j,k})^2}{(d_{i,k} + d_{j,k})} \tag{3.2}$$

The concept and definition of Local Reference Frame as in eq. 3.1 is used also by the authors of [139] for the development of the **SHOT** (**Unique Signatures of Histograms for Local Surface Description**) descriptor. The SHOT descriptor aims at providing an overall robust and accurate description of local surface properties by exploiting both the concepts of *signatures* and *histograms*. In the context of 3D descriptors, *signature* based methods [133, 26, 103] describe geometrical properties of the 3D neighborhood of a keypoint after defining a *support structure*, or a region defined by an invariant local reference frame. While in signature-based methods the selected geometric properties are computed and reported for each point in the support structure, histogram based methods [158, 22, 68] encode the desired geometrical properties into bins. These bins are related to a

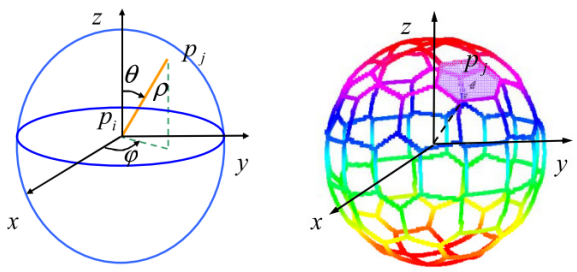


Figure 3.1: Left: polar coordinates system centered on the local reference frame. Right: Subdivision of the spherical support region in sectors with equal solid angles. Figure from [158]

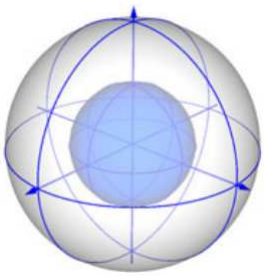


Figure 3.2: Support structure for the SHOT descriptor. Figure from [139]

quantization of the support volume and their definition strongly affects the robustness and descriptiveness of the 3D descriptor. The first major contribution of the SHOT descriptor is the introduction of a unique local reference frame for the cathegorization of the point cloud geometric properties. As for the ISS descriptor, the axes directions are found by computing the eigenvalues of the covariance matrix, or scatter matrix M which is now linearly weighted using the distance of the evaluated point from the support center, or the axis origin:

$$M(\mathbf{p}_i) = \frac{\sum\limits_{|\mathbf{p}_i - \mathbf{p}_j < r_{frame}|} (r_{frame} - r_i)(\mathbf{p_j} - \mathbf{p_i})(\mathbf{p_j} - \mathbf{p_i})^T}{\sum\limits_{|\mathbf{p}_i - \mathbf{p}_j < r_{frame}|} (r_{frame} - r_i)} \tag{3.3}$$

where $\mathbf{p}_i$ is the reference frame center, $\mathbf{p}_j$ is any other point in the neighborhood defined by a maximum distance r_{frame} from $\mathbf{p}_i$. r_i is instead the euclidean distance between $\mathbf{p}_j$ and $\mathbf{p}_i$. The covariance matrix $M(\mathbf{p}_i)$ will be more influenced by the local neighborhood of $\mathbf{p}_i$. The next step is to disambiguate the axes direction in a robust way. For both the z and x axes, the positive direction is selected as the one where the majority of points belong. The y axis is instead determined as the cross product between x and z. In their reference paper, the authors suggest that the combination of the selected weighting scheme and sign disambiguation provides a more repeatable selection of reference axes for different levels of noise and clutter. Inspired by the SIFT descriptor for 2D image features, the definition of the SHOT descriptor involves computing local surface properties and embedding them into a multiple set of histograms placed over the support structure, which delivers a signature-like flavor. The support structure is defined as represented in figure 3.2, where it is represented a partition of space in 4 azimuth, 2 elevation and 2 radial divisions. In reality a 8 part division of azimuth is employed. The geometric property which is encoded in the descriptor is the normal angle computed in the local Reference Frame. Note that normals are computed by solving equation 3.3 for each point in the cloud and selecting z as the eigenvector with lowest eigenvalue. Instead of binning the angle θ_i between each normal and the z axis of the local Reference Frame, the authors decided to build equally spaced bin histograms of the cosine of the angle $cos\theta_i$. This choice also relieve some computational load since the cosine of the angle between two normal vectors can be computed as the dot product of the two normalized vectors:

$$\cos\theta_i = N_i \cdot N_j \tag{3.4}$$

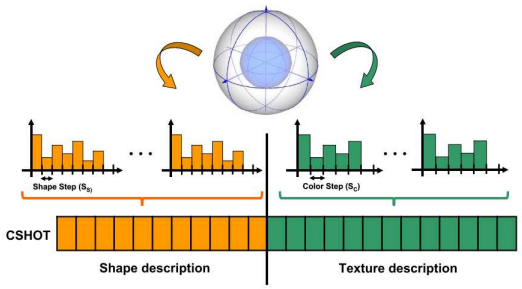


Figure 3.3: Representation of the CSHOT descriptor. Figure from [138]

For less descriptive regions of the point cloud (i.e. flat regions), the difference between descriptors is then limited to lower values since in the angle space, histogram partitions adjacent to the normal angle are coarser, while they are finer for orthogonal directions. Additionally, after populating the histograms, values are interpolated in order to smooth the difference between each neighbor bins. In presence of noise, points could randomly switch the bin where they belong, increasing the difference between descriptors. Invariance to the pointcloud density is then achieved by normalizing the histograms such that the sum of each bin value results 1. The length of the resulting descriptor depends on the number of cosine bins in each histogram. Following the quantization of angles suggested by the authors, given 10 bins per histogram, the length would be 320. Following the guidelines of the PCL library[1], the recommended bin count is 11, which leads to a descriptor of size 352.

An improved version of the SHOT descriptor is also presented by the same authors in [138], where color information is employed for building the descriptor. This version of the SHOT descriptor is called **CSHOT** (or Color-SHOT) and exploits the fact that using stereo cameras or RGB-D sensors, depth information is always accompanied by color information. RGB-D sensors always have RGB cameras to color-map the point clouds, and most of the stereo cameras are built as a rig of RGB cameras. The CSHOT descriptor is built by joining the standard SHOT descriptor and a new ensemble of histograms which encode the color difference between the keypoint k and the points belonging to the support region. As each of the point in the cloud are associated to an RGB triplet $\{R, G, B\}_i$, each of the histogram bins store the frequencies of the following difference:

$$l(R_i, R_k) = \sum_{c=1}^{3} |R_i(c) - R_k(c)| \tag{3.5}$$

where c is the index representing the red, green and blue members of the RGB triplet R. The choice of using an L_1 norm for computing the difference was motivated by the authors experiments.

The **Fast Point Feature Histogram** (**FPFH**) [114] is a speeded up version of the previously presented Point Feature Histogram [115]. The underlying idea is to gather local information about the point cloud without focusing on the center of the support region. Given a keypoint and a reference radius, a reference spherical region defines a neighborhood of points which are selected for computing the descriptor. All the available point pairs in the

[1] http://docs.pointclouds.org/1.8.1/structpcl_1_1_s_h_o_t352.html

support region are selected for computing a set of properties related to their 3D coordinates and normals (see figure 3.4). Specifically, for each pair, a Darboux frame is created such that $u = n, v = (p_j - p_i) \times u, w = u \times v$. The Darboux frame is needed here to create an invariant reference such that all the following properties that compose the descriptor do not depend on the viewpoint. The following angular quantities are quantized and added to an histogram:

$$\begin{aligned} \alpha &= v \cdot n_j \\ \phi &= \frac{u \cdot (p_j - p_i)}{p_j - p_i} \\ \theta &= arctan(w \cdot n_j, u \cdot n_j) \end{aligned} \tag{3.6}$$

As the authors declare, computing the Point Feature Histogram is $O(nk^2)$ where n is the number of points, or keypoints in the point cloud and k is the number of neighbors belonging to a support region, which is proportional to the radius of the support region. The Fast Point Feature Histogram [116] is a faster version of the aforementioned descriptor and relies on a different selection scheme for aggregating the histograms and computing the quantities in eq. 3.7. For computing the FPFH descriptor, instead of considering all the possible point pairs in the support region, first the immediate neighbors of p_i are selected (black lines in fig. 3.4) and the angular quantities of eq. 3.7 are computed on that subset. The selected neighbors are later selected as centers for re-computing the angular quantities until the support region has been traversed. The resulting computational complexity is then $O(nk)$ without significant performance losses.

Choosing a 3D descriptor

Selecting an appropriate descriptor for point clouds registration is not trivial and straightforward. Aiming to recall arbitrary 3D shapes which can be observed with different levels of noise and completeness, different performances in terms of precision and recall can be expected by different choices of 3D descriptors. In the most recent years, a variety of works have been published about place recognition with multiple agents using only 3D information instead of matching visual appearance. The authors of [46] evaluate multiple approaches

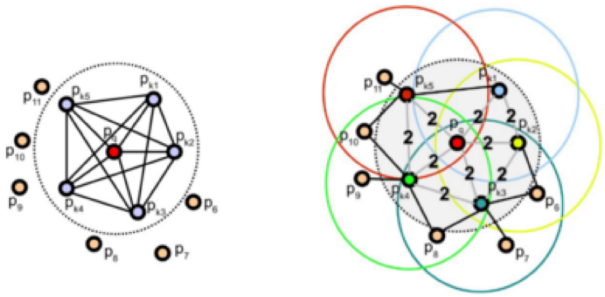


Figure 3.4: Support structure and point pairs for the PFH descriptor (left) and FPFH descriptor (right). Figure from [114]

align heterogeneous clouds representing parts of the same scenario captured by a ground vehicle and an UAV. The ground vehicle builds a consistent map of the environment from an accurate LiDAR SLAM [34] while an UAV performs dense mapping with a monocular camera using a Structure-from-Motion approach as described in [42]. The resulting dense 3D map built by the monocular system is aligned with the UGV map by computing and matching 3D descriptors and is then used for other purposes such as traversability estimation. The authors evaluate multiple choices for 3D descriptors such as SHOT, FPFH and ESF. As ESF is a global descriptor, the points in each cloud are clustered in segments. The authors test their pipeline on an indoor and on an outdoor datasets, choosing the SHOT and FPFH histogram radius empirically after finding plateauing performances. In their experiments, no choice of descriptors guaranteed winning performances in all tests. The usage of FPFH provided better accuracy within simple scenarios but decreasing rapidly with noise, clutter and generally more challenging scenes. SHOT showed however robust performances over all datasets, most of the times achieving the lowest translational and rotational errors after alignment. The authors of [60] evaluate some combinations of 3D descriptors and detectors for the purpose of pairwise point cloud fusion and 3D reconstruction. The feature descriptor involved in this study are SHOT and the C-SHOT variant, PFH, FPFH and a variant of the PFH descriptor which leverages also RGB data similarly to C-SHOT. The authors also evaluate the effectiveness of selecting regions of the cloud to describe which deliver significant information through the usage of 3D keypoint *detectors*. The detectors involved in this work are the Normal Aligned Radial Feature (NARF) and 3D SIFT. The NARF detector [132] extracts keypoints in regions of the point clouds which are stable and can provide accurate surface normals and in proximity of borders. The authors of [132] suggest infact that borders in point clouds, such as discontinuities between foreground and background objects, provide stable and repeatable information if observed from varying viewpoints. The SIFT 3D keypoint detector [23] is an adaptation of the well known SIFT detector for image features. Instead of image intensities, the principle curvature of points within the clouds is used to create a Gaussian scale space. Keypoints lie at the extrema of the DoG (Difference-of-Gaussians) function as for the 2D SIFT detector. Regarding the descriptor evaluation, the authors of [60] observe that for all the descriptors, as they rely on computing difference between normals, the radius for computing normals have a significant impact on the performances. Greater values for the support region radius deliver robustness to noise at the price of a lower uniqueness and greater computational effort. Increasing the radius leads to a quadratic increase in description time for PFH and linear for FPFH, SHOT and C-SHOT. As far as the success rate in pairwise registration is concerned, the results in the author's datasets show that in most of the scenarios, replacing 3D keypoint detectors with simple voxel grids increased the success rate in matching point clouds. In their experiments, the FPFH descriptors outperforms all the other descriptors in most of the dataset. The SHOT descriptor however shows consistent performances even if not the highest. It must be noted that the descriptors were evaluated on RGB-D scenes, captured using a Microsoft Kinect. As it was mentioned before, the quality, resolution, completeness and noise characterizing a pointcloud are critical aspects for the descriptor uniqueness and robustness. Performances are expected to vary significantly between de-

scriptors when different clouds and shapes are evaluated. The authors of the previously cited [34] infact observed a distinct superiority of the SHOT descriptor for LiDAR generated clouds captured by a moving robot. In [57], the authors test a wide variety of feature descriptors including SHOT and FPFH as well as other descriptors which are not implemented in the Point Cloud Library. The set of descriptors is ranked for accuracy, influence of noise and occlusions and finally computational efficiency. To give a fair insight on their performances, multiple datasets are used ranging from synthetic models with high details and low to no noise, to LiDAR scans and dense stereo point clouds. The authors observe that the SHOT descriptor deliver similar performances to the FPFH descriptor, with some fluctuations between datasets. For some datasets with high quality point clouds, SHOT delivers higher precisions than FPFH and for LiDAR scans FPFH outperforms SHOT. However, the performances are not favoring one particular descriptor versus the other in all the sequences therefore no final decision can be made. Where SHOT excels over FPFH is instead on robustness. The authors infact report a better tolerance of SHOT of Gaussian noise applied to the cloud, which can approximate the noise in dense stereo camera clouds while increasing the distance. It is then tested the influence of keypoint selection using Harris3D and ISS, reporting better performances in all tests as compared with uniform sampling. This results is in contradiction with [60]. Finally, the authors suggest to use FPFH for "time-crucial" applications with sparse pointclouds and to use SHOT in generic applications with dense clouds as a trade-off between descriptiveness and computational efficiency.

From this brief review of descriptor performances what emerges is that it not straighforward to predict what level of performances can be achieved by selecting a 3D descriptor. The characteristics of the clouds and the parameters choice for tuning the descriptors have a strong influence on matching accuracy and robustness to noise. For these reasons, the SHOT descriptor will be used in the following experiments as its performances, while not always the highest, do not fluctuate much in different datasets. This and its robustness to noise suggest that it can be trusted in a wide variety of applications.

3.2.1 Binarized SHOT descriptors (B-SHOT)

The SHOT descriptor, as already stated in the previous section, is a vector of 352 floating point numbers resulting from discretizing the support region in 32 sectors and computing 11 bins histograms of point cloud normals for each spatial quantization. Such data structure have a significant impact both on memory and computational effort. A single descriptor requires 1408 bytes of memory which multiplied by thousands of descriptors for a single or several clouds represent a considerable amount of data. While modern computing devices for robotics can store several hundreds of gigabytes of storage, inter-robot communication and data sharing for mapping and localization can suffer if regular SHOT descriptors are shared over a network. For this reason, the authors of [107] introduce a binarization scheme for SHOT which convert the standard floating point descriptor to a binary vector of the same length. While binary descriptors have been used regularly in the past years for 2D images, this work represents the first implementation of binary 3D descriptors for point clouds. In this dissertation, this binarized version of SHOT will be referred to as B-SHOT.

Binary descriptors (such as BRIEF for images) do not directly encode spatial properties of the object to describe, such as intensity derivatives or filter responses. They instead encode the outputs of some checks over the appearance and distribution of the input values contained in a standard SHOT descriptor. Values are analyzed in tuples of four members. Let be S_i a SHOT descriptor and B_i the relative binary version. $\{B_0, B_1, B_2, B_3\}$ are computed from $\{S_0, S_1, S_2, S_3\}$ and so on. To compute these four binary values, five discrete conditions are analyzed. Let be S_{sum} the sum of four SHOT values such that $S_{sum} = S_0 + S_1 + S_2 + S_3$. The following 5 discrete cases are considered in a recursive way. Each case is checked if the previous do not hold:

- Case A: all values in $\{S_0, S_1, S_2, S_3\}$ are zero. Then all values in $\{B_0, B_1, B_2, B_3\}$ are also set to zero.
- Case B: one value from $\{S_0, S_1, S_2, S_3\}$ exceed the 90% of S_{sum}. Then the same position of that particular value is set to 1 in the binary vector while the others are set to 0.
- Case C: the previous cases do not hold and the sum of two of the values in $\{S_0, S_1, S_2, S_3\}$ exceed the 90% of S_{sum}. The position of those two numbers are set to 1 in the binary vector
- Case D: the previous cases do not hold. There are 3 values whose sum exceeds the 90% of S_{sum}. Those positions are set to 1 in the binary version
- Case E: the previous conditions do not hold. All values in binary vectors are set to 1.

This sequence of checks is repeated for every other set of 4 contiguous positions in S_i. This procedure is effective and the resulting binary string is descriptive enough to generate true correspondences between local regions of matching point clouds. This procedure however comes with a slight loss of information. In cases B and D in fact, as it is considered only the sum of the values and not the values themselves, looking at the binary descriptor it is unknown which of the values contributed most to the sum. As an example, if the tuple from the SHOT descriptor is $\{0.3, 0.7, 0, 0\}$, the corresponding binary version would be $\{1, 1, 0, 0\}$. But this result would originate also from $\{0.7, 0.3, 0, 0\}$ which belongs to a different descriptor. However this little loss of information is compensated by a the great advantage of fast matching using the Hamming distance.

3.2.2 Embedding Texture cues (B-Tex-SHOT)

As previously mentioned, many 3D descriptors include also color information beside structure. This results in improved uniqueness of the descriptor vector and by extension a higher matching accuracy. Most of the existing approaches for embedding color information however tend to overpower the color contribution, such as in the case of C-SHOT where the 75% of the descriptor contains color information. Additionally, while being normalized with respect to the central intensity values, the color part is not invariant to non-uniform

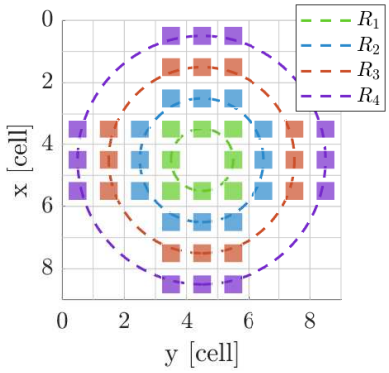


Figure 3.5: 9x9 cell grid of the texture descriptor. Colored patches define the intensity values compared with the center to build the binary descriptor

changes in lighting and appearance. In case of monochrome image sequences, the presence of three different color channels is also unnecessary as the information will be repeated.

In order to increase the precision of B-SHOT we leverage local texture information in a way inspired by Local Binary Patterns [1], where binary strings are produced by a series of checks over intensity differences. We propose an original short descriptor, designed to not overpower the 3D structure part, and as invariant as possible to illumination differences and misplacements of keypoints and local reference systems.

To compute this texture descriptor, the first step is to recall the axes of the local reference systems and all points of the cloud contained in the support region. Points are supposed to be characterized by 4 coordinates: $P = \{x, y, z, I\}$ where I is a monochrome intensity value ranging from 0 to 255. The z axis of the reference system is aligned to the local normal vector, while the x and y axis will identify a plane which for most natural 3D features can be considered to be approximately tangent to the local surface. The xy plane is divided in a coarse 9x9 cell grid where the cell size is computed with respect to the radius of the support region R:

$$l_{\text{cell}} = \frac{2R}{9} \tag{3.7}$$

All the 3D points belonging to the support region are projected in the xy plane and for each cell an intensity value is determined as the average of all point intensities who lie on the cell. For cell ij, being n the number of points projected on it,

$$I_{ij} = \frac{1}{n} \sum_{k=0}^{n} I_k \qquad i, j \in [1, 9] \tag{3.8}$$

To provide illumination invariance, the descriptor is assembled by comparing the intensities in each cell to a local reference value I_{ref}, which is defined as the average intensity of the central 3x3 cells:

$$I_{\text{ref}} = \frac{1}{9} \sum_{i,j=4}^{6} I_{ij} \tag{3.9}$$

It is tested the intensity of cells laying on 4 increasing radiuses from the center, as the

pattern in figure 3.5 shows. Each value of the descriptor is then defined as:

$$d_{\text{int}}(k) = \begin{cases} 1 & \text{if } I(k) > I_{\text{ref}} + t \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where the index k ranges from 1 to 44, which is the number of tested cell intensities and t is a small threshold accounting for noise usually set to 2. As an effect of the coarse cell division, the resulting descriptor tolerates small misplacements of the keypoint location and small rotations of the local reference frame. Finally, a compact binary descriptor combining structure and texture is defined by appending d_{int} to B-SHOT, hence the name B-Tex-SHOT:

$$d_{\text{B-Tex-SHOT}} = d_{\text{B-SHOT}} \cup d_{\text{int}} \tag{3.11}$$

3.2.3 Hamming Distance

Typically, in order to determine the pairwise similarity between descriptors d_1, d_2, in case they are generic floating point vectors, the L_2 norm is computed on their difference (or Euclidean norm), which represents the length of the difference vector.

$$dist(d_1, d_2) = (\sum_{i=0}^{n} (d_{1,i} - d_{2,i})^2)^{\frac{1}{2}} \tag{3.12}$$

If d_1 and d_2 are similar, because they represent regions of similar appearance, then their difference would be small and tend to 0. Therefore matching descriptors between two sets can be found by a Nearest Neighbor search which can be summarized as:

$$argmin_{d_1,d_2}(dist(d_1, d_2)), \qquad \forall (d_1, d_2) \in \{\mathbf{S_1}\}, \{\mathbf{S_2}\} \tag{3.13}$$

being $\{\mathbf{S}\}$ submaps, or containers of descriptors. $dist(d_1, d_2)$ is an operation of computational complexity $O(n)$ with n the number of elements in d_1 and d_2. If d_1 and d_2 are floating point vectors, then the computational effort can be significant and sometimes requiring several seconds or minutes in single-threaded applications. For binary vectors, this task is order of magnitude faster if $dist(d_1, d_2)$ is computed applying the Hamming distance on the binary vector pair.

In Information theory, the Hamming distance between two strings or two vector of equal length is the number of element in which they differ. As an example, let be B_1 and B_2 two binary vectors:

$$\begin{aligned} B_1 &= \{1, 1, 0, \mathbf{0}, 1, 0, 1, 1\} \\ B_2 &= \{1, 1, 0, \mathbf{1}, 1, 0, 1, 1\} \\ H_d &= 1 \end{aligned}$$

They differ in just one member, highlighted in bold, therefore their Hamming distance is exactly 1. For two binary vectors, the Hamming distance is 0 if they are equal in all their members and can be at most the number of elements in case they are all different. From

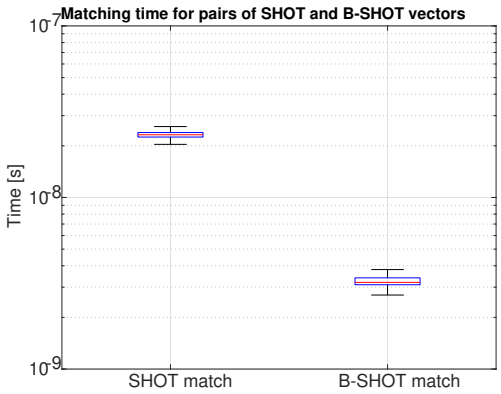


Figure 3.6: Comparison of the matching time between SHOT descriptors and B-SHOT descriptors. The time refers to single pairwise matching of descriptor using eq. 3.12 and the Hamming distance. For a true comparison, eq. 3.12 is applied without computing the square root since it is not needed for just finding minima. The boxplot is evaluated for approximately 10^6 descriptor pairs.

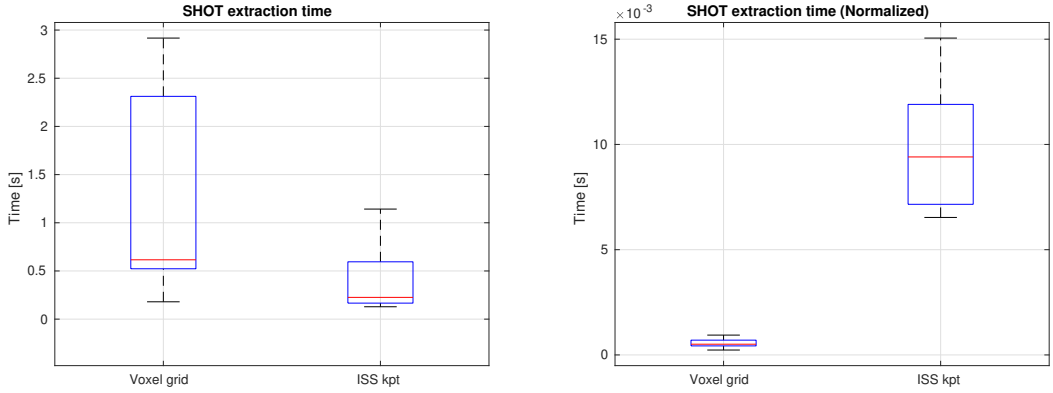
the computational perspective, the Hamming distance is extremely fast to compute since it is performed as an XOR operation which is part of the intrinsics operations included in many processor instructions. Figure 3.6 shows the results of a timing evaluation for computing matches of SHOT and B-SHOT descriptors. Matching times are evaluated for approximately 10^6 descriptor matches. Eq. 3.12 is employed without computing the square root since as only minima are searched within pairs. Computing also the square root does not alter the resulting pairs and saves computational time. The results show that the Hamming distance between two binary descriptors requires one order of magnitude less time to compute than the squared L_2 distance between float vectors.

3.2.4 Keypoint Selection

Having selected the appropriate feature descriptor, the remaining problem to solve is how to extract relevant points where to compute the descriptor. As discussed previously, the advantage of using keypoint detectors is not clear and sometimes simple voxel grids deliver better performances. Multiple reasons can be stated on why not relying on detectors and instead prefer other approaches:

- Computational complexity. Keypoint detectors tend to require a significant effort to compute, in the orders of 10^0 to 10^1 seconds for an average submap. See figure 3.7 for an analysis of the total time spent in detection and description alternating the ISS detector to a simple voxel grid.
- Their efficiency is strictly related to a variety of parameters to tune and their impact is not always clear, limiting the robustness of the approach.

For these reasons in this work we opt for two simple strategies for keypoint selection, the first is through appyling a voxel grid to segmented *obstacle data*, which will be used in the indoor datasets, while the second approach relies on detecting high curvature regions in



(a) Absolute

(b) Normalized

Figure 3.7: Comparison of times for keypoint extraction and SHOT descriptor computation. The test refers to 11 pointclouds captures during robot operation in mixed indoor / outdoor scenarios. Keypoints result from voxel grid downsampling of the clouds with leaf size equal to 2 times the cloud resolution. ISS keypoint extraction is performed with a salient radius of 6 times the cloud resolution. Figure [b] shows the same timings normalized on the number of extracted features. It is very visible the overhead imposed by the ISS feature extraction step. The average numbers of keypoints per cloud are 2111 for the voxel grid approach and 36 for the ISS approach.

the cloud and will be used in outdoor datasets, where the distinction between obstacle and traversable regions is not always clear.

During exploration, the point clouds that can be built by a moving stereo system are typically incomplete. The borders of the pointclouds therefore deliver false information about the appearance of the environment. Their existence is infact related to the sensing methodology, such as the camera field of view, and not to the actual geometry. Matching local information based on the cloud borders is then prone to errors. An important preprocessing step of the clouds before attempting registration is to remove those regions from descriptor evaluation. This problem is solved here with a very light and efficient solution which jointly detects borders and exclude invalid keypoints. Two assumptions are here considered:

- Points in each cloud represent surfaces and not 3D volumes
- All point clouds are firstly processed with a Voxel grid, which discretizes the space in 3D cubes of given size and for each volume it is computed the centroid of the points. This step is performed as a filtering stage and not for the purpose of keypoint selection. The resulting cloud have uniform point density without losing 3D information.

Figure 3.8 depicts the keypoint filtering process. Instead of computing the cloud boundaries and discarding points in their proximity, each point in the cloud is investigated about the number of its neighbors in the radius of the support structure for computing the descriptor. If the previous conditions hold, being P_{res} the point cloud resolution and $R_{support}$ the radius of the support volume, each point located far from the boundaries should have a number

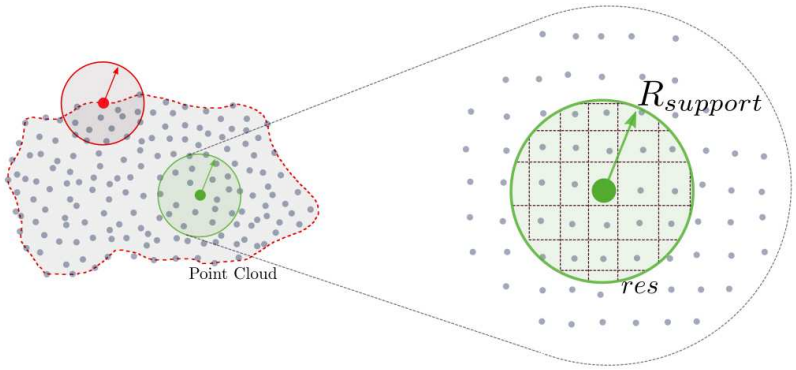


Figure 3.8: Depiction of the keypoint selection strategy for computing SHOT descriptors. The point cloud on the left represents a patch of points measured from the stereo camera belonging to the environment. The borders of this patch are highlighed with a dashed red line and do not represent the actual borders of the object therefore are not descriptive of its structure. If keypoints are selected such as the support region intersects the cloud borders (red circle), the obtained descriptors do not encode a repeatable signature. The keypoint in the cloud center however captures complete information about the cloud and therefore delivers valid information.

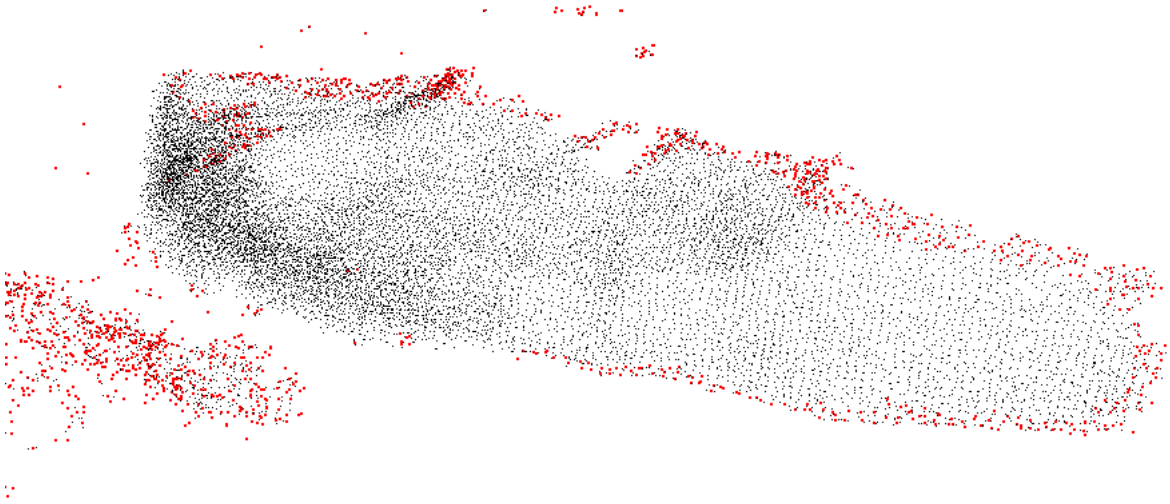


Figure 3.9: Example of keypoint rejection. Black dots are used keypoints, red points are highlight the rejections.

of neighbors n equal in average to:

$$n \simeq \frac{\pi R_{support}^2}{P_{res}^2} \tag{3.14}$$

where $\pi R_{support}^2$ corresponds to the area inside the support radius and P_{res}^2 corresponds to the area of a square centered on a keypoint of side length the cloud resolution. To test a keypoint for being valid or not, the number of its nearest neighbors inside a radius $R_{support}$ is compared with n. If it is lower than n then the keypoint most likely lies on the cloud edges or on isolated patches due to noise, otherwise it is considered valid for computing SHOT descriptors. Figure 3.9 shows the outcome of a keypoint filtering process, representing with red colors the points which are discarded.

Keypoints from Obstacle Data

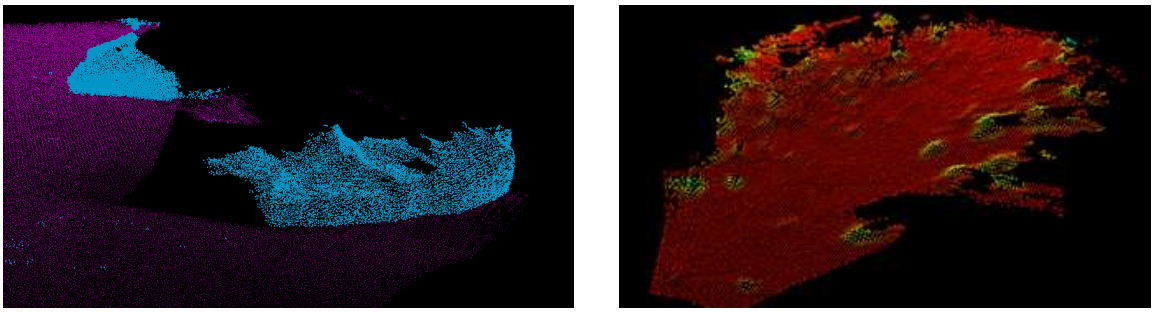
The first approach to extract keypoints is to apply a coarse voxel grid on segmented obstacle data from stereo clouds (see figure 3.10a). In [16] is presented an approach to distinguish non-traversable regions from stereo disparity images which is very efficient if the transition from ground to obstacles is very sharp. The remaining step is to apply a voxel grid filter, selecting usually boxes with 5x5x5cm dimensions, and computing the average coordinate of points belonging to each voxel. This approach provides the advantage of being computationally very light while the downsides resides on the fact that keypoint locations are not repeatable because they are not identified from salient 3D properties. Nevertheless, the underlying assumption is that obstacle information is significative and the radius of support regions for computing descriptors are multiple times higher than the voxel grid size.

Keypoints from High Curvature

In natural planetary-like environments, the distinction between traversable ground and obstacles is not very clear. In this case keypoints are extracted by detecting and downsampling regions in the clouds presenting high curvatures, which are assumed to contain significant 3D information. The first step to obtain a curvature value for each point in the cloud is to compute the *scatter matrix* from equation 3.1. As previously discussed, the three eigenvalues of the scatter matrix $\lambda_0 < \lambda_1 < \lambda_2$ indicates the density of points in each direction. An approximation of the curvature is given by the formula:

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{3.15}$$

The points resulting from thresholding the cloud (see figure 3.10b) according to the value of σ are then downsampled using a voxel grid.



(a) Obstacles (b) High curvature regions

Figure 3.10: Regions in the clouds for keypoint extraction. (a) Obstacle data in light blue computed in an indoor environment replicating natural features in form of rocks. (b) Submap extracted in an outdoor environment. The colormap highlight curvature starting from red which corresponds to zero.

3.3 Bags of Binary Words for Place Recognition

The *Bag of Words* paradigm consists in a strategy to classify documents by the frequency of the words that they contain. While firstly applied for classification of text documents [6], the Bag of Words approach has been widely applied to the field of computer vision for performing place recognition tasks [128] and loop closure detection. In computer vision, the task of recalling similar images from a highly populated set, which can be a video captured from a mobile robot, can be very onerous from a computational perspective depending on the approach employed. As images are represented as matrices of millions of elements, it is unconvenient both to store them as such and to compare them pixel-wise for detecting similarity. For this reason, the seminal works [102, 33, 44] showed how to summarize the information contained in images by relying on using sparse sets of feature descriptors as *visual words*. The first step is to generate off-line a *vocabulary* of ordered visual descriptors in a partially supervised manner using selected descriptors which should reflect all the observable feature types in images. An image is converted into a *Bag of Visual Words* by counting the frequency of the detected descriptors associated to words in the vocabulary. This way, multiple images can be compared for their similarity by computing the similarity between their Bag of Words representation, allowing loop closure detection up to the frame rate of a video feed. As this approach is both effective and fast, several state of the art visual SLAM systems relies on Bag of Words (or Bag of Visual Words) for loop closure detection, such as ORB-SLAM2, S-PTAM, RTAB-MAP and others.

In this work, the Bag of Words paradigm is utilized in a novel approach for relocalization and loop closure detection. Instead of using visual words, a vocabulary is generated from *binary 3D descriptors* which encapsulate geometric information about discretized sections of the mapped environment, called *submaps*. Each submap, which can contain up to hundreds of thousands of 3D points, is summarized by a Bag of Word vector which encode the frequency of 3D descriptors. Similar submaps are found by comparing the their Bag of Words vectors in a very fast and efficient way, finding candidate matches in the order of milliseconds. As the vocabulary contains binary vectors, the memory footprint of the data

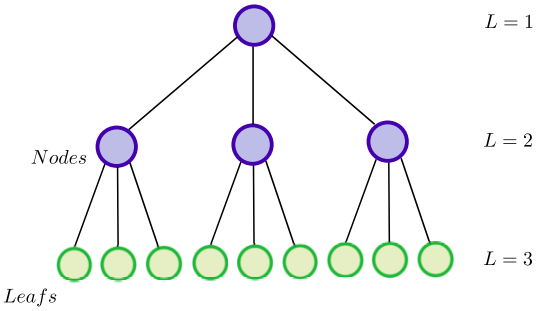


Figure 3.11: Simple representation of a tree structure with $k = 3$ and $L = 3$. Green dots represent *words*, purple dots represent *nodes*

structure is extremely reduced compared to vocabularies of float descriptors, facilitating storage and communication of the vocabulary in resource constrained platforms.

3.3.1 Building the Vocabulary Tree

The *vocabulary* is an ordered container of reference elements, or *words*, whose presence is detected for building Bag-of-Words vectors. The vocabulary must therefore be built in advance so that each vector to be compared refers to the frequency of the same set of words. In a vocabulary of text words, each element can be efficiently queried as they are ordered alphabetically. An *order* is therefore necessary also with generic word types such as visual descriptors or 3D descriptors. Generic feature descriptors can be ordered using distance metrics since it is always defined a function which returns a pair-wise similarity score. A vocabulary of descriptors can be built by sub-dividing the descriptor space in W clusters as in [129] so that Bag of Words vector can be built by counting the frequencies of indexes of clusters. A more efficient approach however, is to build the vocabulary as an *hierarchical* set of clusters, which enclose descriptors more and more similar as the clustering level increase. The vocabulary can be then be represented as a *tree*. Let be k the number of choosen clusters per each level and L the number of levels, the number of words, or the vocabulary nodes at the deeper level is:

$$W = k^L \tag{3.16}$$

The clustering process is based on the *k-means++* algorithm [5] which aims at grouping descriptors based on their proximity in such a way that the distances between descriptors and their cluster center are minimum. Following the notation the author's paper, let χ denote a set of descriptors, initially the entire set. Let be x_i the i-th descriptor in this set and $D(x)$ the preferred distance metric between the descriptor and the closest cluster center to x. $D(x)$ can be an Euclidean distance as well as a Hamming distance in case of binary descriptors. The *k-means++* algorithm serves to give an optimal first estimate of the cluster centers before using a regular *k-means* algorithm for growing the cluster. The procedure is structured as follows:

1. Select the first cluster center c_1 by randomly extracting a point from χ

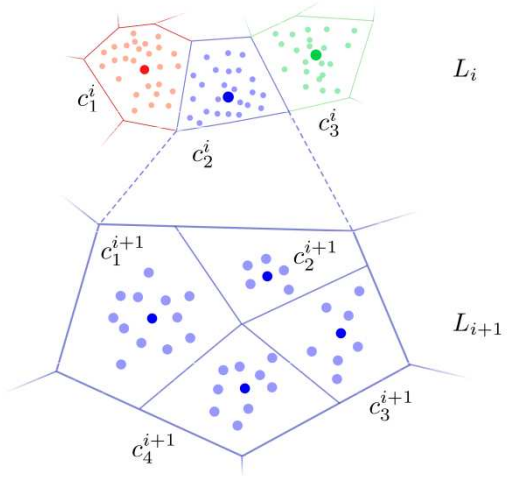


Figure 3.12: Schematic representation of the vocabulary hierarchical tree structure. Each represents a descriptor in the descriptor space, here represented as a 2D space. Each bigger dot represent a cluster center, clusters are higlighted with different colors. Each cluster is subdivided itself with the same *k-means* approach for a finite number of levels L

2. Select each other cluster c_i by sampling x with probability

$$\frac{D(x)^2}{\sum_x D(x)^2}$$

   where $D(x)$ is the distance from x to the *closest* cluster center c. The number of clusters grow after each iteration

3. Repeat the previous point until the desired number of clusters has been initalized

At this point, the standard *k-means* algorithm can be used to cluster the descriptors:

1. Associate each descriptor x with the closest cluster, i.e. the cluster c_i for which the distance with x is minimum

2. Update the position of each cluster c_i by re-computing the centroid of the descriptors belonging to the i-th cluster

3. Repeat the above steps until convergence

This procedure defines the clusterization of a single *level*, a hierarchical tree structure of nodes is generated by repeating the same procedure for each of the k clusters. Figure 3.12 depicts this process, and figure 3.11 represents the tree structure of the resulting vocabulary.

Word weighting

In text documents, not all words have the same impact on defining their category. Articles or prepositions are too common across different texts and therefore are not discriminative enough. With descriptors of 3D clouds the same principle applies as well, local properties which are present across several point clouds are not useful to identify one of them

in particular. In order to mitigate this inefficiency, words are associated to a weight term as suggested in [**zissermann2006**] using a *term frequency-inverse document frequency* approach. Weights are determined by two contributions: the *term frequency* and the *inverse document frequency*. The first contribution can be computed as follows:

$$tf(i, S) = \frac{n_{i,S}}{n_S} \tag{3.17}$$

Where $n_{i,S}$ is the number of occurrences of the word i in submap S and n_S is the number of words contained in submap S. This weight contribution belongs to the interval $[0; 1]$ and favors words which have a high occurrence in a point cloud suggesting that they give a significant character to that entity. However, as previously mentioned, if the same highly frequent words are present in all point clouds, they have a very low discriminative power and must be accounted less. For this purpose, the *inverse document frequency* is introduced:

$$idf(i) = log(\frac{N}{n_i}) \tag{3.18}$$

where N is the number of clouds used for generating the vocabulary and n_i is the number of clouds which contain the word i. This term is high if the smallest number of clouds possible contains a certain word. The combination of the two terms results in the final weigth:

$$w_i = tf(i) \cdot idf(i) \tag{3.19}$$

and it is high when a point cloud contains multiple occurrencies of a word i while the other clouds have none. Other options are available for computing weights, such as using the plain term frequency or a simpler binary scheme where the weigth is 1 if the cloud contains that words, regardless of the frequency, or 0 otherwise. In our experiments the combined approach of term frequency and inverse document frequency leads to the best results.

3.3.2 Bags of Binary Words Vectors

After the vocabulary is generated in a first training sequence, it is possible to convert new point clouds in Bag of Words vectors $\mathbf{v}$. Figure 3.13 depicts the process. For each submap, the extracted binary descriptors are fed into the trained vocabulary. At each level, the node which results in the lowest Hamming distance is selected until the lowest level is reached. The leaf index and correspondent weight is added to the BoW vector $\mathbf{v}$. After processing all the descriptors in the target submap, the Bag of Words vector is complete and represents a signature for the submap, storing its identity in terms of indexes and weigths instead of a stack of descriptors. As a final step, all weights comprised in $\mathbf{v}$ are normalized therefore accounting for the different number of words that two Bag of Words vectors might have. This representation allows for a very fast matching between submaps by computing a pair-wise score of Bag of Words vectors instead of descriptor to descriptor distances. As suggested in [102], a similarity score between two Bag of Words vectors $\mathbf{v}$ and $\mathbf{w}$ can be computed as:

$$s(\mathbf{v}, \mathbf{w}) = 1 - \frac{1}{2}\|\mathbf{v} - \mathbf{w}\|_1 \tag{3.20}$$

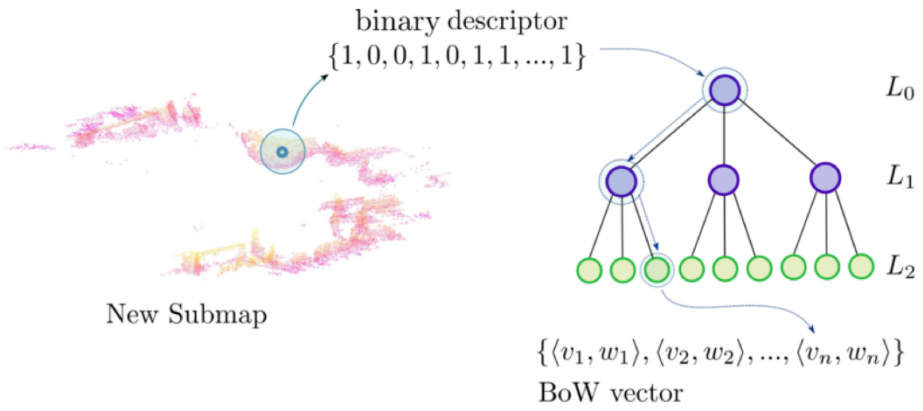


Figure 3.13: Generation of a *Bag of Words* vector from the set of binary descriptors belonging to a submap, or container of descriptors. Each binary descriptor *traverses* the vocabulary tree, choosing at each level the closest node (cluster center) according to the Hamming distance. Once arrived at a *leaf* node, the contribution is added to the BoW vector

The score $s(\mathbf{v}, \mathbf{w})$ is comprised in the interval $[0; 1]$ where 0 is total difference and 1 means complete similarity. Note that $\mathbf{v}$ and $\mathbf{w}$ must be seen as ordered *map* containers containing pairs of indexes and weights as in this representation:

$$\mathbf{v} = \{\langle id_0, w_0 \rangle, \langle id_1, w_1 \rangle, ..., \langle id_n, w_n \rangle\} \tag{3.21}$$

therefore the difference $\mathbf{v} - \mathbf{w}$ involves only the weights associated to common indexes.

3.3.3 Word thresholding

As each descriptor belonging to a submap traverses the vocabulary tree (see figure 3.13), the path is always defined by the cluster center scoring the lowest Hamming distance with the descriptor. However, when the descriptor arrives at the lowest level or the leaf level, the Hamming distance can still be high. This is the case when the training phase of the vocabulary did not involve similar descriptors to the ones detected at relocalization phase. If this condition is not addressed properly, the Bag of Words vectors will be populated with words that do not belong to the submap in consideration, even if they are the closest to the descriptors which are fed into the tree. In [102], it is shown as high performances in terms of image recalls are obtained for number of leafs up to 1 or 10 millions. It is implicitly supposed however that the image features used for building the vocabulary are variegated and representing a complete spectrum of the detectable features in images. This is easily achieved by choosing carefully the images to build the vocabulary in such a way that they represent various types of environments and objects. With point clouds it is challenging since 3D descriptors are usually viewpoint but not scale invariant and dependent on the choosen parameters. Furthermore, a single point cloud, in the form of a submap, requires multiple observations of an environment to be built. In terms of images used, a likely number would be hundreds or thousands of frames to build a single point cloud delivering up to about a thousands of SHOT descriptors. Additionally, while collecting

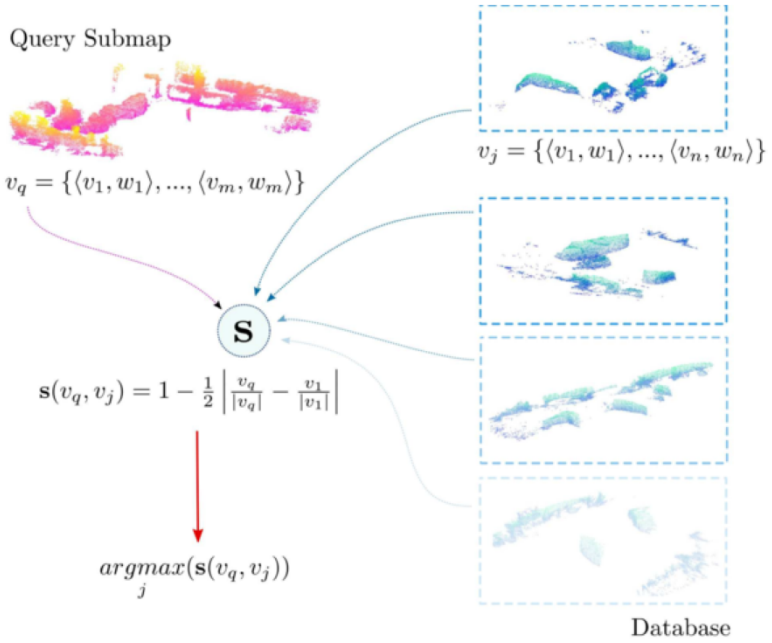


Figure 3.14: Representation of the similarity evaluation between submaps by scoring Bag of Words descriptor pairs

images representing different environments could take surfing the web for a few minutes, obtaining point clouds for training would require to explore the actual environment with the target sensor setup and therefore would be infeasible in many situations. For this reason, the relocalization pipeline described in this work, uses previous observation of an environment as the training set for further multi-session mapping or inter-robot loop closures. The training setup is very likely to be lacking observable descriptors, and a special care is required for building consistent Bag of Words vectors.

In this work we propose a thresholding scheme in order to add to the Bag of Words vectors only leaf nodes representing the actual local appearance of a point cloud. However, instead of applying a hard-coded threshold on the Hamming distance at the lowest level, The Bag of Words vector generation scheme described beforehand is modified to apply a re-weighting of the original *tf-idf*. Let be $w^* \in [0; 1]$ an additional multiplicative term such that the final weight associated to the word i is:

$$w_i = w^*(H(c_i, d_i)) \cdot tf(i) \cdot idf(i) \tag{3.22}$$

where w^* is a function of the Hamming distance H between the cluster center c_i and the descriptor d_i at the lowest level of the tree (i.e. level L recalling equation 3.16). From empirical evidences during the experiments, an effective weight function for w^* is selected to be:

$$w^*(H) = \begin{cases} \frac{\alpha}{1+\lambda H} + \beta, & \text{if } H > H_{thresh} \\ 1, & \text{if } H \leq H_{thresh} \end{cases} \tag{3.23}$$

Where H is the Hamming distance between the leaf c_i and the descriptor d_i. The parameters

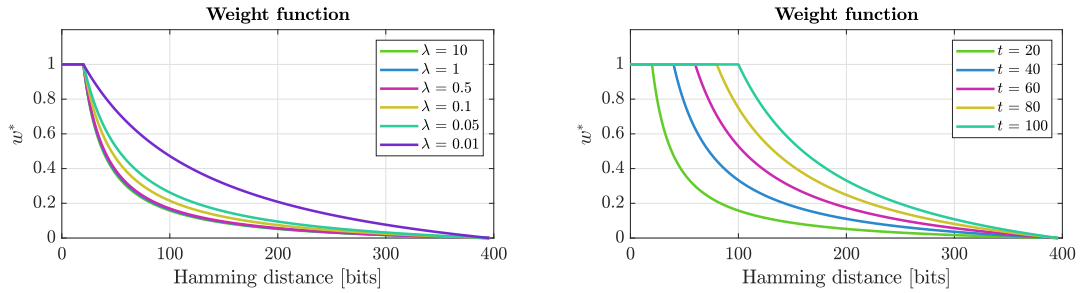


Figure 3.15: Plots of the weights as a function of the Hamming distance. In this example, Hamming distances are contained in the interval $[0; 396]$ where 396 is the length of the B-Tex-SHOT descriptor. Left: plot of the function with fixed $H_{thresh} = 20$. Right: plot of the function with fixed $\lambda = 1$

α, β ensure that, for a given value of λ, the following constraints are satisfied:

$$\begin{aligned} \frac{\alpha}{1 + \lambda \cdot H_{thresh}} + \beta &= 1 \\ \frac{\alpha}{1 + \lambda \cdot \text{length}(d)} + \beta &= 0 \end{aligned} \tag{3.24}$$

The first constraint ensures that if the Hamming distance is in the interval $[0; H_{thresh}]$, the additional weighting term is 1, otherwise if the Hamming distance is equal to te length of the descriptor, the weight is 0. The Hamming distance is in fact maximum when each of the bit fields in the binary descriptors are different from each other. Selecting a value for λ, which defines the steepness of the function, ad H_{thresh} which defines the maximum distance for which no additional weigths are applied, the other parameters can be determined as:

$$\begin{aligned} \beta &= \frac{1 + \lambda H_{thresh}}{\lambda(H_{thresh} - \text{length}(d))} \\ \alpha &= -\beta - \lambda \cdot \text{length}(d) \cdot \beta \end{aligned} \tag{3.25}$$

the function is then fully determined and its shape can be observed in figure 3.15.

3.4 Relocalization

3.4.1 Candidates Selection

The scoring scheme described in section 3.3.2 is intended to provide a metric proportional to the pair-wise similarity of submaps reflecting the fact that they might or might not contain common words. Under the hypothesis that the vocabulary is descriptive and well-formed, meaning that it contains words that belongs to the observed environment and the frequency of those words as they appear in the vocabulary is as uniform as possible, high correlation between vectors suggest that the submaps match. However, a challenge arises when good candidates for submap matching must be selected from a set because the absolute values

of the scores are very dependent on a variety of factors such as the vocabulary size and composition. Traditional means of finding best matches between vectors, may they be visual descriptors or other types of descriptors, can be divided in two families of approaches:

- Nearest Neighbor search: let be $\{\mathbf{D}\}_{i=1...n}$ a set of database, or train descriptors and $\{\mathbf{Q}\}_{j=1...m}$ a set of query descriptors for those it is to be found the best match in $\{\mathbf{D}\}$. Let also be $\mathbf{s}(\mathbf{q}_i, \mathbf{d}_j)$ a scoring function which computes the similarity of the two descriptors. In many applications, this scoring function might be an L_2 norm of the descriptor difference, or the Euclidean distance. The Nearest Neighbor search consists then in finding, for each $\mathbf{q}_i$, the database descriptor $\mathbf{q}_i$ for which $\mathbf{s}$ is minimum. This test is usually performed a second time by comparing not only the query descriptors with the train descriptors but also the train with query descriptors, consistent couples are then selected as matches. If the two sets of descriptors are very different, the Nearest Neighbor search will eventually deliver matches with are false in reality. Such test should be then accompanied with a threshold check so that only consistent matches in the forward and backward pass have a distance between the descriptors which is lower than a threshold. This threshold is usually handcrafted and must be applicable for all the tests. This scheme can not be applied in the submap matching scenario as there is no clear and repeatable way to select a fixed threshold value, as it must be dependent on the vocabulary usage.

- Ratio test: originally proposed by Lowe in [84], it does not consider any absolute value of the descriptor pairs scores or distances. The ratio test instead rewards the uniqueness of scores. For all query descriptors $\mathbf{q}_i$ and for all train descriptors $\mathbf{d}_j$, a matching pair is selected if the highest score from $\mathbf{q}_i$ to each train descriptor is higher than the second highest multiplied by a given value. This means that a matching pair is selected if its score "spikes" with respect to all the others. In the context of submap matching this cannot be applied since many train submaps can have very similar scores with the query if they share observations of the same parts of the environment.

The impossibility of choosing fixed score thresholds and the fact that often recalls do not spike in a unique way, requires to develop a custom selection scheme for this context. The procedure developed for this matter is summarized in algorithm 1. The candidate selection scheme proposed in this section is intended to run online at submap generation time. Given a first set of submaps from the same robot or another robot obtained in a previous SLAM session, the goal is to select a number of submaps to be fed in the further steps of the relocalization pipeline such that they are minimal and most representative of good candidate matches.

As a new SLAM session begins, the candidates selection scheme waits to receive a minimum number of submaps. The first step is, in fact, to evaluate the spectrum of scores which can be expected during the mapping session. That is done in order to be able to threshold the scores based on the expected range between maximum and minimum values.

After having received n_{start} submaps, a matching threshold is initialized as:

$$T = t_{rel} * (bestMax - bestMin) + bestMin \tag{3.26}$$

where $bestMax$ and $bestMin$ are the highest and lower scores received so far and t_{rel} is the only parameter which must be manually decided prior to execution. t_{rel} is a relative threshold and ranges from 0 to 1. The pipeline, initialized as described, picks matching

Algorithm 1: Selection of candidate Submap matches from BoW scores

Input :

- $\{\mathbf{s}_i\}$: BoW scores between i-th submap and database submaps
- t: relative matching threshold over BoW scores, $t \in [0, 1]$
- n_{start}: minimum number of submaps to trigger a relocalization attempt

Output:

- $\{\mathbf{C}_i\}$: set of candidate matches from current submaps to database

```
bestMax = 0
bestMin = Inf
if i < n_start then
    update bestMax
    update bestMin
    store {S_i} for future use
end
else if i = n_start then
    update bestMax
    update bestMin
    T = t * (bestMax - bestMin) + bestMin
    foreach stored {s_j} do
        add to {C_i} each entry k in {s_j} such that {s_j}_k > T
    end
end
else if i > n_start then
    update bestMax
    update bestMin
    T = t * (bestMax - bestMin) + bestMin
    add to {C_i} each entry k in {s_i} such that {s_i}_k > T
end
```

candidated based on their relatively high scores with respect to the others. If the first submaps have very low absolutes scores and therefore represent different scenarios, wrong candidates would be still passed to further checks and hopefully rejected. However, at each new submap reception, the values $bestMax$ and $bestMin$ are updated, so that when actual matching submaps are received, the relative threshold T_{rel} is updated to higher values and from that time on, only more and more correct matches are sent to the next stages of the pipeline. This selection scheme is therefore designed to initialize to some performance level

and update iteratively adapting to the score ranges which are observed at each submap receival.

3.4.2 Match Validation

Once candidates submap matches are extracted looking at high BoW scores, a validation scheme must be implemented in order to discard all the false matches and only keep the correct ones, in case they exist. The false matches rejection scheme implemented in the single robot SLAM pipeline of [122] is not applicable in this case, as it relies on a strong prior about the pose of submaps, considering also the uncertainty over the pose estimates. For relocalization, no pose prior exists between consequent runs as estimating a first transformation between the two is the actual objective of the system. While the *candidate selection* scheme relies on bag of binary words created using binary descriptors derived from SHOT, the validation stage employs the original SHOT descriptors which are stored in memory. Given a pair of submaps $\mathbf{S}_1$ and $\mathbf{S}_2$ selected as relocalization candidates, the first step is to find correspondences between the two sets of SHOT descriptors $\{\mathbf{D}\} \in \mathbf{S}_1$ and $\{\mathbf{Q}\} \in \mathbf{S}_2$ by performing a Nearest Neighbor search in two directions. For each $\mathbf{d}_i \in \{\mathbf{D}\}$ the descriptor $\mathbf{q}_j \in \{\mathbf{Q}\}$ is a match only if $\mathbf{d}_i$ is Nearest Neighbor of $\mathbf{q}_j$ and $\mathbf{q}_j$ is Nearest Neighbor of $\mathbf{d}_i$. Numerous correspondences amongst the ones computed are expected to be wrong. This is because 3D information can be challenging to match if captured by stereo vision systems instead of precise 3D LiDAR sensors. Traditionally, outlier rejection schemes when aligning point clouds or images rely on RANSAC [41] which is prone to fail if the number of outliers exceeds the number of inliers in estimating a *model*. In this case, the *model* to estimate would be a 4D transformation from two submaps $\mathbf{T}_2^1$ excluding the roll and pitch angles which are directly *observable* from IMU measurements, while $\{x, y, z, \psi\}$ can drift. The probability of success for the RANSAC algorithm is infact:

$$P(\text{success}) = (1 - w^n)^k \tag{3.27}$$

where w is the fraction of inliers, n is the number of samples selected to compute the model in the current iteration and k is the number of iterations performed. As visible in figure 3.16, the probability of failing for RANSAC is almost 1 for the expected fractions of inliers which can likely be lower than 0.3 for submaps which share actual descriptor matches. The only way for the RANSAC algorithm to return a set of inliers is to lower the minimum consensus parameter, which defines the fraction of input data that must comply with the candidate model. By doing so, there would be no obvious way to distinguish between wrong and correct models in the case where wrong descriptor matches outnumber the correct one, if present. Therefore the RANSAC algorithm alone can not be applied to this scenario.

The process of selecting correct matches between point clouds is modeled as a *voting* process of 4D transformations. The underlying hypothesis is infact that wrong descriptor matches should vote for 4D transformations randomly placed in the x y z and yaw space, while correct transformations, if present, should be voted repeatedly by multiple matches both across the same submap pair as well as consequent submap pairs.

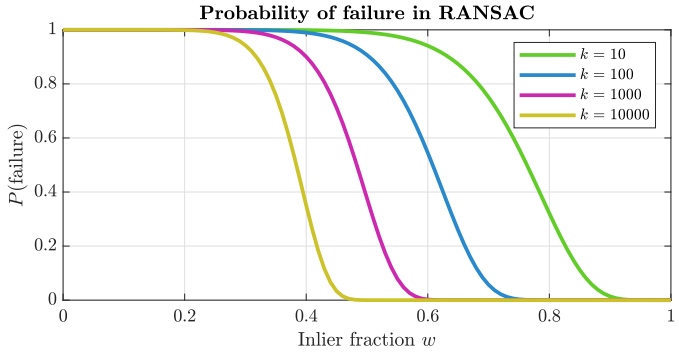
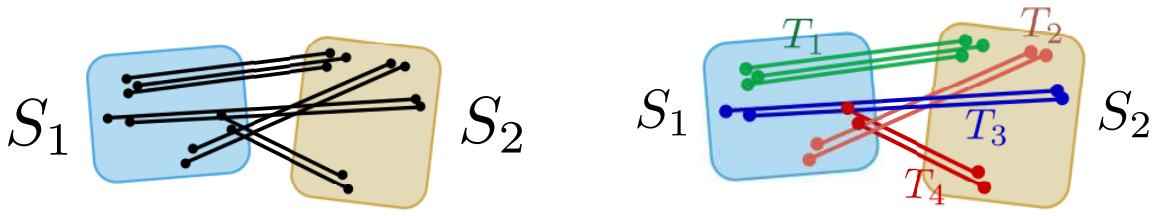


Figure 3.16: Probability of failure in the RANSAC algorithm to find a good model between the dataset corrupted by outliers. As more outliers are present, the fraction of inliers is lower and the probability of failing is steeply increasing. More iterations help to select and validate good models however the computational time is $O(k)$ where k is the number of iterations

Hough3D Clustering

Given a submap pair candidate for match verification, the matched SHOT descriptors are clustered such that each resulting group of matches suggest the same rigid transformation and furthermore resembles the correspondences between the same 3D *model* (such as a rock, or in general an agglomerate of close keypoints) across the two submaps. The grouping scheme is presented in [137] and it is here very briefly summarized. Inputs to the algorithm are matched keypoints (regardless of the descriptor employed) and the Local Reference Systems associated to each of them. Given a model instance, which in generic input scenes can be a cluster of neighbor keypoints over a rock, a unique reference point for that model is computed as the centroid of the set of keypoints. It is also stored the set of vectors pointing from each keypoint in the model to the reference point (blue vectors in figure 3.18). Each blue vector connecting keypoints to the reference point is stored in the local coordinates of each reference frame. After descriptor matching, each corresponding keypoint in the scene casts a vote in a 3D Hough space regarding the coordinates of the reference point (pointed by green lines in fig 3.18). If multiple scene keypoints vote for the same position of the reference point, then they likely represent the same object instance. While the aforementioned process is related to just one model and a target scene comprising



(a) Descriptor Matches

(b) Hough3D Grouping

Figure 3.17: This figure represents the effect of grouping descriptor matches across two submaps using Hough3D. Groups of matches who suggest the same rigid transformation are labeled with different colors.

multiple models, the PCL[2] implementation of the Hough3D voting scheme embeds the algorithm in a RANSAC process to extract multiple models from a first pointcloud and grouped correspondences with matched keypoints in a second point cloud. This process helps to forward to the next validation step only groups of consistent keypoint matches driving the selection of good transformation candidates between pairs of candidate submap matches.

3.4.3 Transformation Clustering

The previous step shows how the whole set of individual keypoint matches are grouped in a discrete set of clusters identifying the correlation of aggregated keypoints which resemble the same *object* and define individual hypotheses of rigid transformations between the two submaps. In order to validate a match between two submaps, each keypoint group after Hough3D votes for a specific transformation that aligns the origins of the first and second mapping sessions. As depicted in figure 3.19, the rigid transformation between two keypoint clusters identifies the relative pose of the two origins through the local pose of each submap. Under the hypothesis that the query and database SLAM sessions are not characterized by excessive localization errors, all the reference frames of matching submaps should differ by a constant rigid transformation, or more generally their difference should be bound in a small interval. Let be $\mathbf{T}_i$ and $\mathbf{T}_j$ the rigid transformations between the local reference frame and the respective map origins of two matching submaps from a database and query SLAM session (see figure 3.19). The transformation $\mathbf{T}_j^i$ is computed during the Hough3D grouping step for each keypoint cluster and does not account for the pose of each submap respectively to their global origin. Instead it is derived from the local coordinates of each keypoint which are expressed in the local origin of each submap. However, the reference frame of each submap is expressed in the respective origin from the most recent 6D SLAM pose estimate $\mathbf{T}_i^q$ for the second session (or relocalization session) and $\mathbf{T}_j^{db}$ for the submap in the first session. As each submap is *rigid*, any transformation applied to their points applies also to their local reference frame, therefore it can be easily shown with the help of figure 3.19 that:

$$\mathbf{T}_q^{db}\big|_{cl_k} = \mathbf{T}_j^{db} \cdot \mathbf{T}_i^j \cdot (\mathbf{T}_i^q)^{-1}\big|_{cl_k} \tag{3.28}$$

where $\mathbf{T}_q^{db}$ is the transformation between the origins of the first and second SLAM session, and it is the ultimate target of the relocalization pipeline. Eq. 3.28 is however evaluated

[2] http://pointclouds.org/

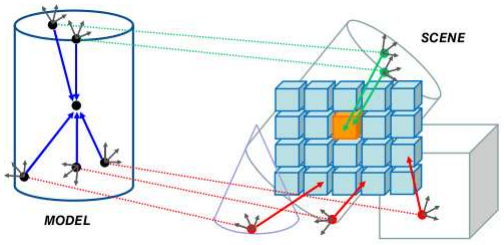


Figure 3.18: Depiction of the Hough3D voting scheme to recognize instances of an object model in a scene where multiple objects are present. Figure from [137]

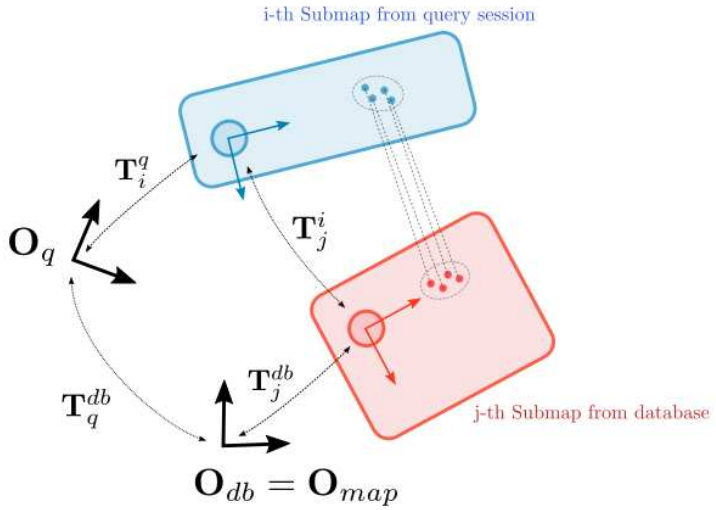


Figure 3.19: Schematic representation of the rigid transformations between pairs of matching submaps and the respective SLAM session origins. Dashed lines highlight the keypoint matches grouped from Hough3D voting across the two submaps. The rigid transformation that they identify leads to the transformation between the two session origins through the local pose of each submap

for the inter-submap transformation $\mathbf{T}_i^j$ which is suggested by the $k-th$ keypoint group after Hough3D, and can be either a cluster of correct keypoint matches or not. In order to select only the transformations suggested from clusters of correct matches, a voting scheme is established on the 4D coordinates of $\mathbf{T}_q^{db}\big|_{cl_k} = \{x, y, z, \psi\}$. Should multiple groups of correct keypoint matches be present, the transformation between the reference systems that they vote for should gather more votes than the wrong ones because it can safely be assumed that wrong matches vote for random transformations. This voting process is described in detail and validated over synthetic tests in the following paragraph, and it is performed across several submap matches until a set of correct transformations is safely selected as winner.

An Incremental Clustering Scheme for 4D Transformations

The naive solution of discretizing the transformation space into bins of given dimension is very likely to lack uniqueness. If for example the center of a set of good transformations is located between two adjacent bins, their similar probability would render the choice between the two ambiguous. An incremental clustering technique is therefore developed in order to:

- Group into clusters a growing number of transformations
- Adapt the cluster boundaries without relying on fixed bins

The first transformation fed in the incremental clusterer defines the first cluster center. For each other new transformations, the closest cluster center is found by minimizing the L_2 distance between the current transformation and the cluster origins. It is also minimized

Algorithm 2: Incremental Clustering Algorithm for computing consensus over a candidate set of transformations

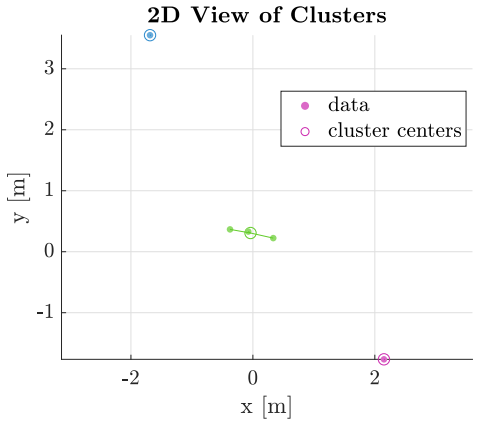
Input :

- $\mathbf{T}_q^{db}\big|_{cl_k}$: 4D Transformation between the origins of the two sessions as suggested by Hough3D cluster k ($\{x, y, z, \phi\}$)
- t_{xyz}: spatial threshold for grouping transformations
- t_ϕ: yaw threshold for grouping transformations
- t_p: ratio threshold for accepting a cluster

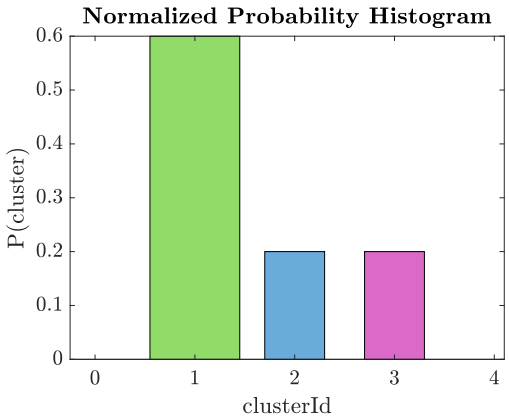
Output:

- $\mathbf{T}$: consensus transformation for relocalization

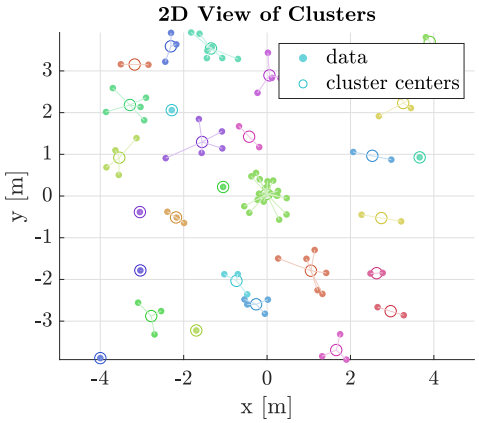
```
if first transformation then
    initialize clusters {C}
end
else
    search for closest cluster C_i
    if T_i < t_xyz & t_φ then
        add T_i to cluster C_i
        recompute C_i as new centroid
    end
    else
        add new cluster to {C}
    end
end
compute P_i for each cluster
normalize P_i
if 1 − p_2nd / p_max > t_p then
    return T_i
end
```



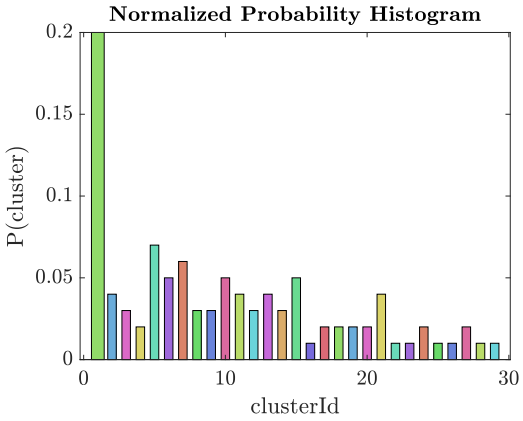
(a) 5 tf 2D sample



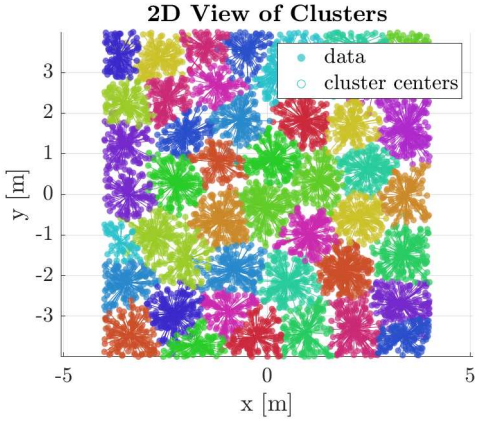
(b) 5 tf histogram



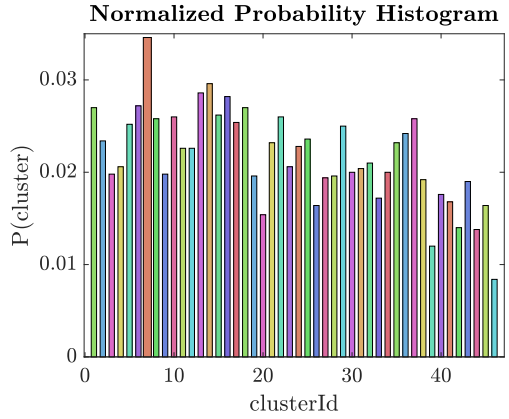
(c) 100 tf 2D sample



(d) 100 tf histogram



(e) 1000 tf 2D sample



(f) 1000 tf histogram

Figure 3.20: Results of the iterative clustering technique described in algorithm 2. The full 4D transformations are simplified as 2D translations for an easier visualization. (a) and (b) show the outcome of clustering a realistic number of 5 transformations, (c) and (d) are related to a number of 100 transformations. Samples are randomly drawn according to a Gaussian distribution centered on the origin, representing **true positives** and a second uniform distribution representing noise. (e) and (f) show clusters from an uniform distribution of 5000 samples, just for testing purposes

the difference in yaw angles:

$$argmin_i \begin{cases} ||\mathbf{T}_{xyz} - \mathbf{c}_{xyz}||_2 \\ |\mathbf{T}_{\psi} - \mathbf{c}_{\psi}| \end{cases} \tag{3.29}$$

where $\mathbf{T}_{xyz}$ and $\mathbf{c}_{xyz}$ denote the translation component of the current transformation and the i-th cluster, while $\mathbf{T}_{\psi}$ and $\mathbf{c}_{\psi}$ are related to the yaw. If the spatial and angular distances between the closest cluster center are lower than the thresholds t_{xyz} and t_{ψ}, the current transformation is added to the i-th cluster. The value of c_i is then updated by averaging all the cluster member, therefore representing the *centroid* of all members. In case the distances exceed the thresholds then a new cluster is initialized. Figure 3.20 represent the results of this clustering scheme by simplifying the transformations to 2D planar translations. Two scenarios are generated by sampling values from two distribution, a Gaussian distribution with zero mean and unitary sigma and an uniform distribution in $x \in [-5, 5]$ and $y \in [-5, 5]$. The uniform distribution serves at representing outliers. In the first scenario 100 samples are drawn while in the second one just 5 samples are considered. The second scenario is most representative of the number of relocalization matches that can be found in a real-world SLAM session. In both sessions, the distance threshold is set to 1. From the results, it emerges that the clustering scheme is effective in grouping the highest number of elements belonging to the "true" set of transformations. The normalized histograms on the right are generated by dividing the cluster member count by the total number of candidates:

$$P_i = \frac{n_c}{n_{all}} \tag{3.30}$$

A good cluster of transformations must also be unique, meaning that its probability should be much higher than all the others. To ensure that the most probable transformations are correct, the highest probability value is tested against the second highest in a *ratio* test.

$$\mathbf{c}_i \text{ is correct } \iff \left(1 - \frac{P_{2nd}}{P_{max}}\right) \geq 0.5 \tag{3.31}$$

The value of 0.5 discriminates in fact the winning cluster in case of the lowest number of candidates possible. This is the case of a cluster of size 2 and just one more cluster of size 1. The trivial case of a single cluster is not considered. For situations where more inliers are present, the ratio test should score higher values than the imposed threshold.

3.5 Experiments: Inter-Robot and Multi-Agents Loop Closure

3.5.1 The Lightweight Rover Unit (LRU)

The relocalization pipeline discussed in this chapter, while being potentially employable by any flying or ground exploration vehicle, focuses on the usage of the LRU (Lightweight Rover Unit) as the experimental setup. In figure 3.21 two rovers, namely LRU and LRU2, are depicted operating in cooperation during the test campaign on Mount Etna for the

(a) LRU on mount Etna (b) LRU and LRU2 on mount Etna

Figure 3.21: The LRU and LRU2 rovers during the test campaigns on Mount Etna in 2016. Pictures from [123]. The pictures show LRU with a Kinova arm attached to the rear for manipulation and LRU2 carrying a hyper-spectral camera along with the stereo head and monocular color camera.

ROBEX (Robotic Exploration of Extreme Environments)[3] mission. A detailed description of the LRU rovers as well as the SLAM system which provides localization for each individual mapping session follows in the next sections.

The Rover

The LRU rover is designed to be a lightweight exploration vehicle relying on a robust locomotion system which can adapt to harsh terrain condition and using only vision systems for navigation. The dimensions of the rover are approximately 1090 mm in length and 730 mm in width. Four individually powered wheels, which can be turned in-place independently from each other, guarantee travel velocities around 1 meter per second. Each actuated joint, including wheel drives, is moved with a brushless DC motor built and validated at DLR called *ILM* motors [58], also employed for the MASCOT (Mobile Asteroid Surface Scout) lander [109] [110]. The wheels are attached to two bogies which are connected to the rover body by a *Serial Elastic Actuator (SEA)*. This solution allows to benefit from both active and passive suspensions. The passive element allows damping of the travel surface irregularities for the sake of a better visual estimation and pose filtering as well as safety of internal electronic components. The active counterpart allows instead to adjust the center of mass of the rover body in order to maximize traction in slopes or to have

[3] https://robex-alliance.de

Figure 3.22: Picture of the *pan-tilt* unit of the LRU rover. Figure from [149]

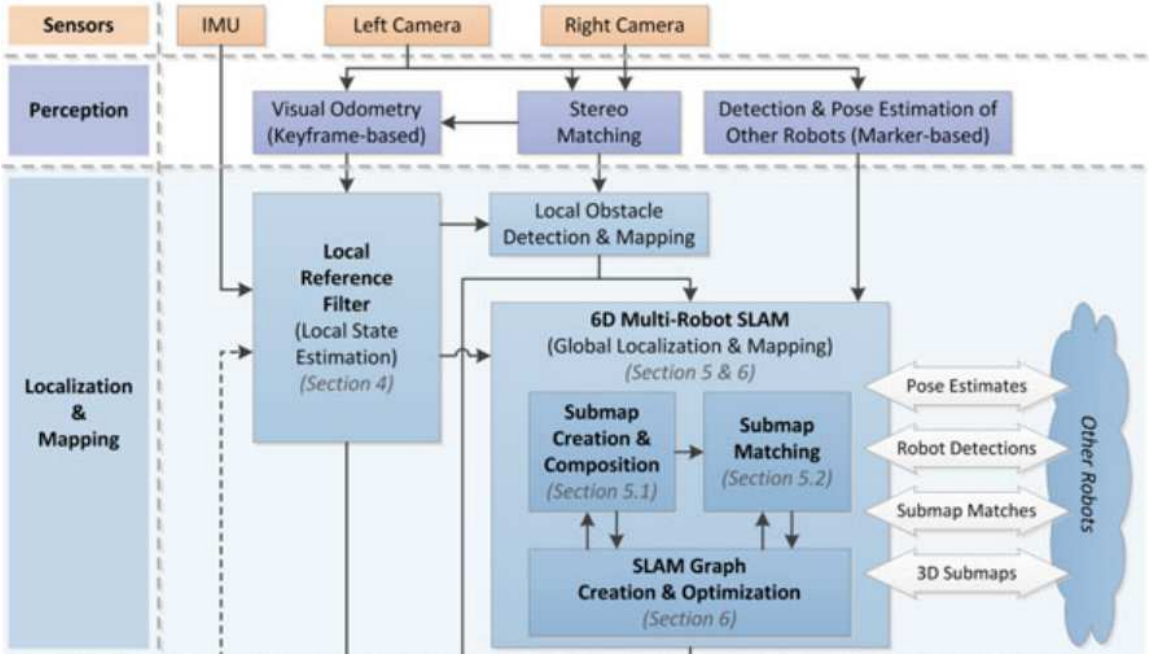


Figure 3.23: Schematic representation of the localization and mapping system for the LRU and LRU2 rovers. Picture modified from [122]. This scheme shows how sensor data (stereo camera and IMU) is used to provide localization through Visual Odometry and filtering, and how the relevant 3D information in from of obstacle data is used in the SLAM system along with the filter estimates to produce a globally consistent 3D map. Inter-robot detections through visual markers allow to merge two maps from collaborative robotic teams.

different views of the environment using the onboard pan-tilt head. As for the sensor setup, the LRU rover employs two stereo camera heads (figure 3.22), one for visual localization and other, rear-mounted for supervising the manipulation. Each stereo head employs two monochromatic Guppy PRO-F125B cameras with 1/3" sensor size and resolution 1292x964. The main stereo head has a 9 centimeter baseline while the secondary head has a 6 centimeter baseline. The central camera is a Guppy PRO-F125C with identical sensor size and resolution. Linear accelerations and angular velocities for filtering and attitude are provided by a Xsens MTi-10 IMU which is mounted in the body barycenter. Processing of stereo frames to produce disparity images is provided by a Spartan-6 LX75 FPGA, which applies the *Semi-Global Block Matching* (SGBM) [64] algorithm of each incoming stereo pair and produces disparity images of 1024x508 pixels resolution with a frame-rate of 14.6 Hz. All the further processing of data is performed by the on-board PC which is an Intel NUC equipped with an i7-3740QM quad-core processor (CPU clock 2.70 GHz). Power is provided by a couple of Li-Ion batteries with a nominal capacity of 208 Wh supplying a voltage ranging from 23V to 30V.

6D SLAM

Figure 3.23 shows an overview of the localization and mapping architecture running onboard the rovers [122]. Sensor data are processed in a hierarchical and decoupled way to both provide instantaneous localization as well as consistent mapping for individual robots as

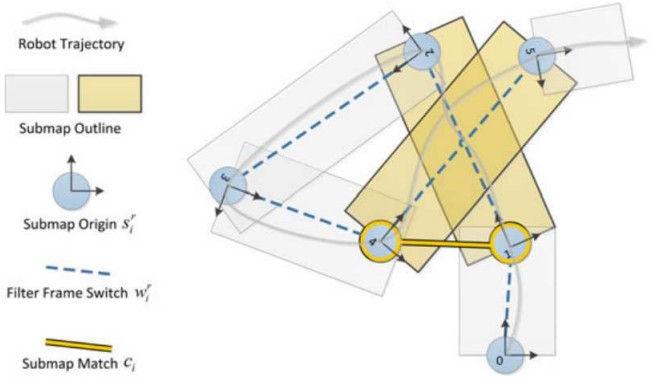


Figure 3.24: Simplified example of SLAM graph. Each reference frame is a Local Reference Frame for the localization filter and serves as origin for the respective 3D submap (grey boxes). Yellow boxes denote submaps whose bounding boxes overlap for a significant amount and therefore are likely to match. Picture from [122]

well as cooperating teams. Stereo Visual Odometry provides visual localization using a fixed window keyframe-based approach [65]. Camera poses estimated from VO are fused with IMU data to provide accurate localization with respect to a Local Reference System [118], switched periodically to the current robot location according to user-defined metrics such as excess of travelled distance or excess of pose uncertainty. For a detailed discussion on the visual-inertial visual estimation pipeline the reader is referred to [122].

Switching of Local Reference Frames partition the travelled environment in *submaps*, partial point clouds which are assumed to be locally accurate given the low drift rate of the visual-inertial localization. Within each submap, points belonging to obstacles are distinguished from the full point cloud using an obstacle detector based on disparity data [16]. Obstacle points are used for keypoint extraction and description through SHOT features. 3D information is used to establish *loop closure* constraints in the pose graph to correct localization errors and drifts. As suggested in [17] in fact, 3D information from obstacle data provide unanbiguous and informative geometrical feature which can be exploited for matching submaps, in other words finding corresponding submaps from 3D similarity and estimating the rototranslation between them. An exhaustive search for pairwise matching submaps is infeasible, as for n submaps the number of possible combinations is:

$$\frac{n!}{2(n-1)!} \tag{3.32}$$

In the rover SLAM system, the method for detecting potentially matching submaps relies examining the overlap between bounding boxes of submaps in the $x-y$ plane, gravity aligned (figure 3.24). Given the most recent optimized poses from the pose graph, the uncertainty of each submap origin (origin of Local Reference Frames) inflates each submap bounding box:

$$\begin{aligned} \text{overlap}'_x &= \text{overlap}_x + 2 \cdot \Delta\sigma_x \\ \text{overlap}'_y &= \text{overlap}_y + 2 \cdot \Delta\sigma_y \end{aligned} \tag{3.33}$$

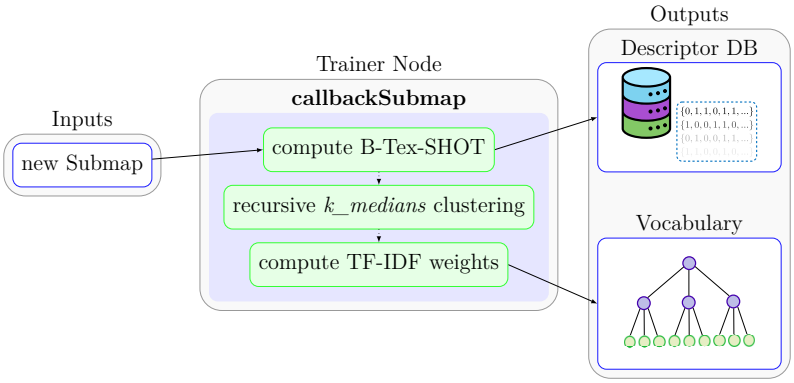


Figure 3.25: Functional scheme of the *trainer* ROS node. This node is used during the first exploration stage, when the robot is constructing a 3D map for the first time. Inputs to the node are *submaps*, discretizing the 3D space based on the covariance of the local localization filter estimates. The trainer node receives new submaps, extracts SHOT descriptors and binarize them. When a sufficient number of submaps is received, a *kd-tree* organizes the descriptors in a hierarchical architecture, which will represent the *vocabulary*. The vocabulary is then saved to disk as a *yaml* file as well as the set of binary descriptors which represent the database information.

where overlap_x and overlap_y are the overlaps between a pair of submaps in x and y directions. The inflated overlap values rank each pair of potential matches for which $\text{overlap}_{x,y} > 2m^2$. Each submap pair undergoes validation as previously described in section 3.4.2, this stage is shared with the relocalization pipeline.

This paragraph gave a brief overview of the localization and mapping system used by the rovers to build a map and localize themselves. It is highlighted how the current system performs loop closures and re-localization, which is by relying on *prior* information about the possible location. It is then clear how the relocalization pipeline, focus of this chapter, improves the mapping capabilities of the system by defining a selection scheme of matching submaps without any loss of generality.

3.5.2 ROS Architecture

The relocalization pipeline is implemented in the ROS (Robot Operating System)[4] environment, running on a Linux platform with an Intel Xeon E5-1620 v3 and 8 Gigabyte of RAM. The performances observed with this workstation are similar to what can be expected from the hardware running in the LRU robot. Each functional part of the relocalization pipeline is embedded in ROS *nodes*, which are stand-alone applications receiving and publishing custom timestamped messages to be shared along the robotic network. Two nodes are dedicated to relocalization, one for the original run and the second for the relocalization session.

[4] https://www.ros.org

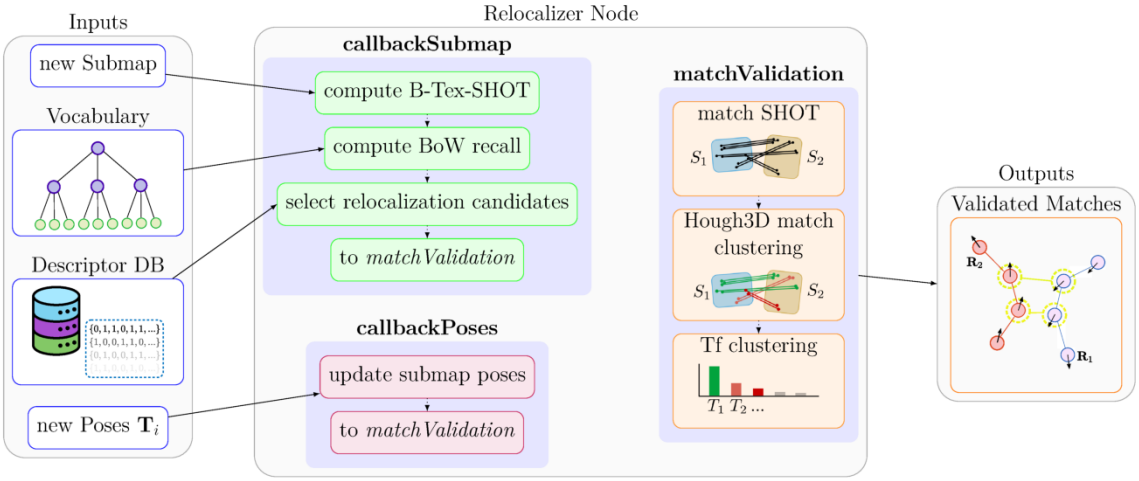


Figure 3.26: Functional scheme of the *relocalizer* ROS node. This node is run during the relocalization session, where map information collected during the training session is re-used to quickly localize the robot with respect to the first map. Inputs to the node are *submaps*, which can be constructed with different parameters from the training session, the binary descriptor vocabulary as well as the descriptor database. Candidate matches computed with the relocalization pipeline are fed to a matching verification stage which discards wrong candidates and select valid ones. A set of consistent relative transformations between submaps is pushed to a pose graph

Trainer Node

The first node is referred to as *trainer_node* (overview of the node in Fig. 3.25). Inputs to this node are the submaps published in the current session by the rover. Each submap represents a portion of the explored map and is referenced to the world coordinate system by the position of the robot where the submap was firstly triggered. The contained 3D point cloud undergoes a series of procedures with the goal of extracting meaningful 3D descriptors in binary form. The role of this node is to build a vocabulary of descriptors enabling to compare submaps though bag of words vector similarity.

Once new submaps are received, a voxel grid filter is applied on a fine grid (2x2x2 cm box size) to reduce noise in the 3D coordinates of points.

The following step is **normal estimation**. Normal vectors are associated to each point of the cloud and can be visualized in first approximation as the normal of a locally tangent plane to each location in the cloud. The correct definition however is the eigenvector with lowest associated eigenvalue of a local covariance matrix computed as:

$$\mathbf{M}_i = \frac{1}{n}\sum_{j=0}^{n}(\mathbf{p}_j - \bar{\mathbf{p}})^T(\mathbf{p}_j - \bar{\mathbf{p}}) \tag{3.34}$$

where $\mathbf{p}_j$ are points in the neighborhood of a point $\mathbf{p}_i$ and $\bar{\mathbf{p}}$ is the *centroid* of the neighborhood, which is defined as:

$$\bar{\mathbf{p}} = \frac{1}{n}\sum_{j=0}^{n}\mathbf{p}_j \tag{3.35}$$

Table 3.1: General parameters for the relocalization pipeline

Context	Parameter Name	Value
Normals	$r_{support}$	10 - 15 cm
Keypoints	l_{grid}	5 - 10 cm
SHOT	$r_{support}$	10 - 15 cm
Vocabulary	k	9
	L	6
	weight scheme	TF-IDF
BoW recall	t_{rel}	0.3
	n_{wait}	3
	H_{thresh}	50
	λ	1

From SVD decomposition of the covariance matrix $\mathbf{M}_i$, the eigenvector with the lowest associated eigenvalues defines the *direction* of the local normal. The spatial distribution of normals will be exploited by the SHOT descriptor to compute local features. The following step is **keypoint extraction**, where a discrete set of 3D points is selected as the anchor positions from which compute the SHOT descriptors. As already discussed in section 3.2.4, two approaches can be switched depending on the scenario. If the traversable and non-traversable areas are easy to identify, obstacles are segmented and downsampled. Otherwise curvatures values, already computed while extracting the normal vectors, are thresholded to select meaningful areas from the 3D standpoint and downsampled.

SHOT descriptors (section 3.2) are then computed on the selected keypoints using support radiuses typically ranging from 10 to 15 centimeters. The resulting set of descriptors, defined by arrays of 352 floating point values, is **binarized** using the approach discussed in section 3.2.1. Both the original SHOT descriptors as well as the binary counterpart are stored as part of the submap object, the first will be used for the purpose of submap matching in the context of the current SLAM session, the second is used instead to build a **vocabulary tree** of binary descriptors [44] [33] for 3D place recognition in the context of multi-session and multi-robot mapping. The vocabulary is build as a kd-tree using the k-means algorithm, the general parameter values for this task are reported in table 3.1. At the end of the training or first session, the vocabulary is saved to disk, reporting all the nodes and respective weights in a compressed YAML file. The choice of using compact binary descriptor reflects in very low dimensions of the vocabulary file, which reaches dimensions of few megabytes after training with $\sim 10^6$ binary descriptors.

Relocalizer Node

The relocalizer node is run during the second mapping session in order to recognize previously visited places and localize the rover on the previous map. As the node is started, the vocabulary is loaded from the system as well as a database of B-SHOT descriptors extracted from the previous set of submaps. This allows to build at runtime the bag-of-words vectors for the previous submaps according to the desired weighting scheme (section 3.3.3). As new submaps are received and processed as described for the trainer node, the

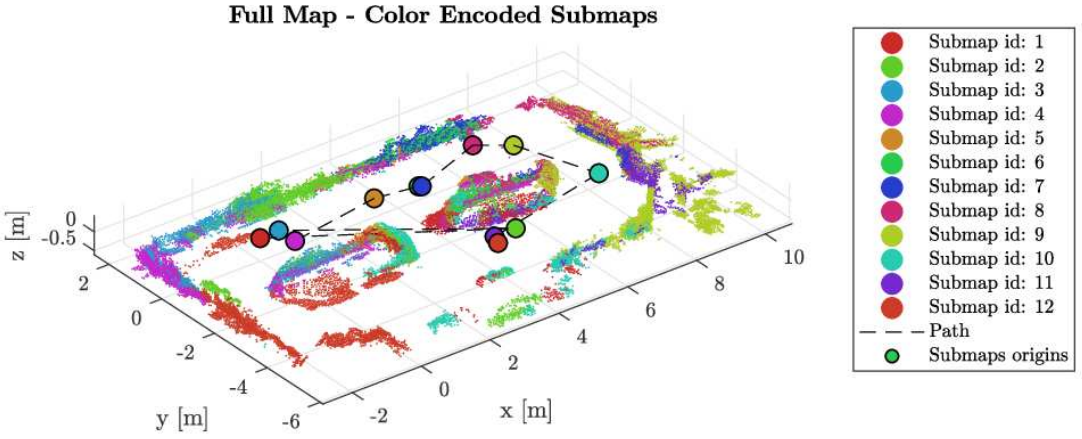


Figure 3.27: Outcomes of a mapping session in the laboratory environment of the Robotics and Mechatronics center at DLR in Oberpfaffenhofen near Munich. The environment comprises mock-ups of rocks and natural features. The dimensions of 8x12 meters are constrained by the field of view of a Vicon system to measure the true position of the LRU robot. Different colors highlight the discretization of the full map in *submaps*. Colored dots connected by a black dashed line represent the origin of submaps of the same color. Only *obstacle* clouds are shown here, the ground is removed.

vocabulary is invoked to transform binary descriptors in bag-of-words vectors. Candidates for relocalization are searched by computing the pairwise similarity scores as explained in section 3.4.1. Each submap is referred to the current world reference system from the first robot position at submap switch time, submap position are subjected to updates during the SLAM session as inter-session loop closures are found. For this reason, a second thread in the relocalization nodes listens to submap positions updates for computing alignment with the database from the most recent positions. When more than n_{wait} new submaps are processed, the candidate selection pipeline elects a set of candidate matches between current and database submaps which exibit high similarity scores. This measure of similarity does not guarantee the correctness of a match, therefore each pair of candidate matches is fed to a match validation stage. Here, the original SHOT descriptors are matched and grouped using the Hough3D scheme. Transformations from the Hough3D groups are clustered in an ordered probability histogram triggering relocalization if a consensus is met.

3.5.3 Preliminary Validation of BoW Recalls

As a first investigation of the effectiveness of bag-of-words scoring for pointcloud recall, the relocalization pipeline is run using the same data for the training and localization stages. This test serves as ensuring that similar pointclouds, or in this case *identical* pointclouds, exhibit large bag-of-words scores. In order to not facilitate the recall process, no thresholding schemes (section 3.3.3) are applied during the generation of bag-of-words vectors. This way, any possible ambiguity between submap recalls due deficiencies of words in the vocabulary tree would manifest in the results. In addition, even if the submap pointclouds between the training and relocalization stages are identical, a rotation of 180 degrees is applied for just the relocalization stage. If the submaps were compared without

any perturbations between them, the pair-wise scores would just result in perfect recalls: scores of 1 for the same submaps otherwise 0. However, rotating the submaps for recalling results in a different selection of keypoints, which are arranged in a *voxel grid* scheme. Shifting the positions of keypoints results consequently in different *local reference frame* orientations therefore different values for the SHOT descriptors. This trick mimics the consequences of mapping the same environment in different times.

Figure 3.27 shows the outcomes of a mapping session where the LRU (Lightweight Rover Unit) was driven around the test environment in the laboratories of the Robotics and Mechatronics center at DLR in Oberpfaffenhofen near Munich. The test area for this run comprise some mock-up surfaces of natural rocky features. A ceiling-mounted Vicon tracking system provides continuous ground truth during the rover motion covering an area of 8x12 meters, which results in a possible travel area of approximately 96 m^2. During the exploration session, the stereo camera is used to observe as much of the environment as possible using the pan and tilt motion of the camera head. During this session, which is approximately 600 seconds long, 12 submaps are collected, some of them also overlapping to some extent. Thanks to the Vicon tracking system, the origin of each submap is known with millimeter level accuracy, and a precise grade of overlap between the different submaps can be computed.

Computing ground truth for the relocalization pipeline

The results of this relocalization pipeline can be observed as the results of a classifier. Given a set of submaps for training and one for relocalization:

$$\begin{aligned} \mathbf{S}_{db} &= \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n, \}_{db} \\ \mathbf{S}_{reloc} &= \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n, \}_{reloc} \end{aligned}$$

the relocalization pipeline tries to predict to which of the submaps in $\mathbf{S}_{db}$ each submap in $\mathbf{S}_{reloc}$ is most similar. For this problem, a reference measure for performance evaluation can be a similarity metric based on spatial overlap. Given the true submap origins measured by the Vicon tracking system, each submap pair can be scored as follows. Let be n_1 and n_2 the total number of 3D points in each submap of the pair $\langle \mathbf{s}_1, \mathbf{s}_2 \rangle$. For each point p_i in $\mathbf{s}_1$, submap $\mathbf{s}_2$ can be searched for a *Nearest Neighbor* which is the point satisfying the following relation:

$$\arg\min_{p_j} ||p_i - p_j||_{L_2} \qquad p_i \in \mathbf{s}_1, \quad p_j \in \mathbf{s}_2 \tag{3.36}$$

For all sets of points in 3D space this search would produce an output. In order to guarantee that the points in the resulting pair actually matches, a threshold on the Euclidean norm must be set.

$$||p_i - p_j||_{L_2} < thresh \tag{3.37}$$

Reasonable values of this threshold should be related to the pointcloud noise, therefore in first approximation the threshold is set to a relaxed value of 0.5 meters. Let be n_{pairs} the number of Nearest Neighbors pairs that satisfy eq. 3.37, a scoring scheme for grade of

		True Value	
		Value Positive	Value Negative
Predicted Value	Predicted Positive	True Positive	False Positive
	Predicted Negative	False Negative	True Negative

Table 3.2: *Confusion Matrix* representing the possible outcomes of a classification task

overlap is defined as:

$$\mathbf{o}(\mathbf{s}_1, \mathbf{s}_2) = \frac{2 * n_{pairs}}{n_1 + n_2} \tag{3.38}$$

where the multiplicative factor 2 relates to the fact that each pair involves 2 points. This score is comprised between 0, when no pairs are closer than the set threshold, and 1 when all points are closer than the threshold.

BoW recall vs True overlap

Figures 3.28 and 3.29 report the results of comparing the BoW scores for each submap pair with the true overlap score defined in the previous section. Each submap of the set of 12 visibile in figure 3.27 is compared against the rotated set. Sub-plots in the two figures refer to one individual submap and contain the BoW scores computed for all the other submaps in the rotated set (red line). The blue lines refers to the true overlap computed by a Nearest Neighbor search (equation 3.38). It is evident how for about half of the submaps, spikes in the bag-of-words scores completely match spikes in the true overlap curves. For submaps 1-7 the correlation between bag-of-words scores and spatial overlap is very unique suggesting that in most cases BoW scores are a valid index for recalling correct matches between point clouds. In some cases, local maxima suggest the possibility of detecting false matches. BoW scores in fact measure the similarity between descriptors, which could be present in more than one submap in the mapping session. Another explanation for these local maxima is in the non-completeness of the vocabulary tree: the small perturbations introduced by the 180 degrees rotation of the submaps from the relocalization set introduce notable differences between the new and old descriptors. When the vocabulary tree is queried with the new set of descriptors, the bag-of-words vectors are populated with word indexes which do not match with the new descriptors. In other words the cluster centers at the lowest level of the vocabulary tree are still quite different from the new descriptors, ending in wrong 3D matches. In conclusion, these results suggest that the proposed approach for relocalization using bag of binary words is a feasible solution but the BoW vector building scheme must take into account the possibility of having to deal with an incomplete vocabulary. This motivates the need of a re-weighting scheme for the contributions to the bag of words vectors (section 3.3.3) and its effectiveness is highlighted in the next section.

3.5.4 On the effect of BoW re-weighting

Traditional applications of bag-of-words based image retrieval make use of dense vocabularies of features up to 10^6 elements. In [102], the authors show the performances of an image retrieval algorithm which is most efficient when using a vocabulary tree with $k = 6$

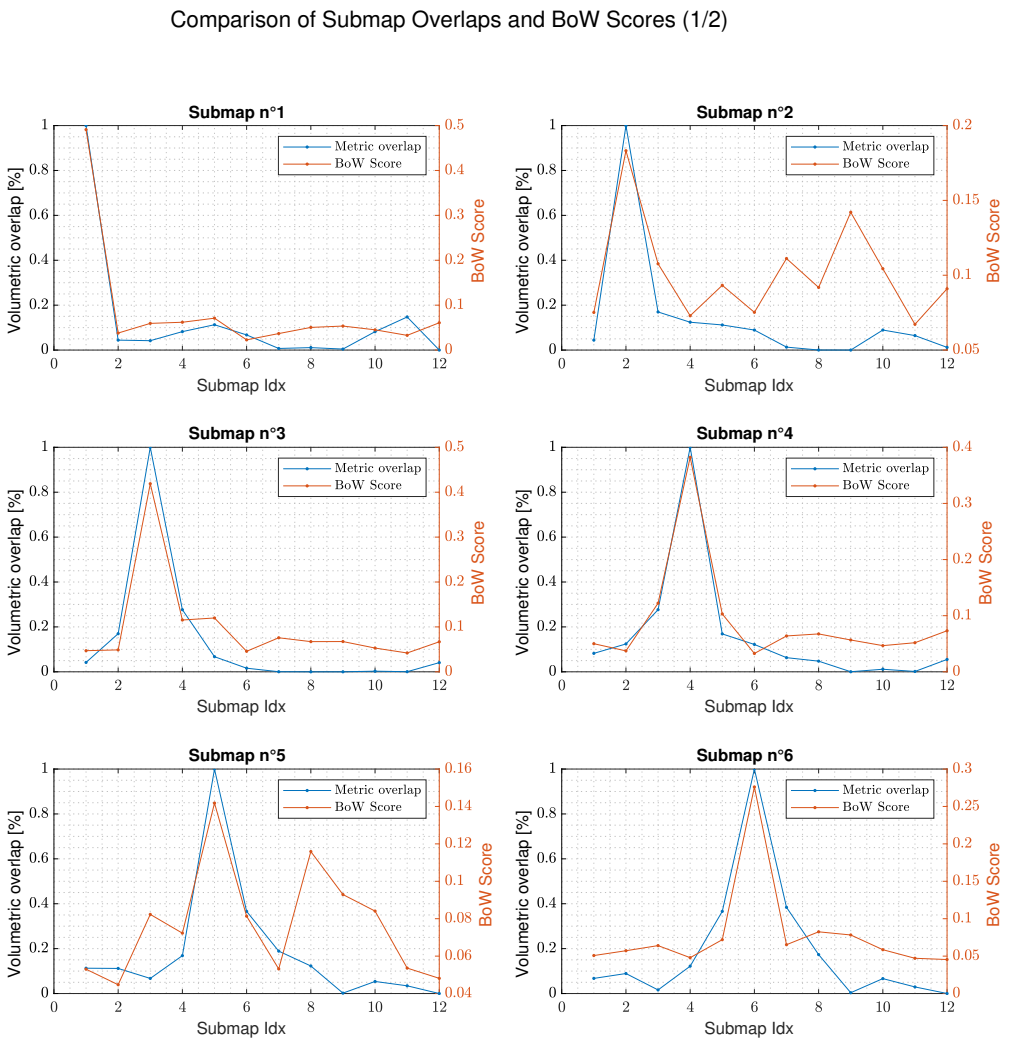


Figure 3.28: Comparison of the bag-of-word scores between submaps and true spatial correlation. Submaps in the mapping session shown in fig. 3.27 are tested against each other. The BoW recall is computed as in eq.3.20 without any thresholding scheme applied. Volumetric overlaps are computed as the fraction of 3D neighbors over the total number of points in the submaps pair.

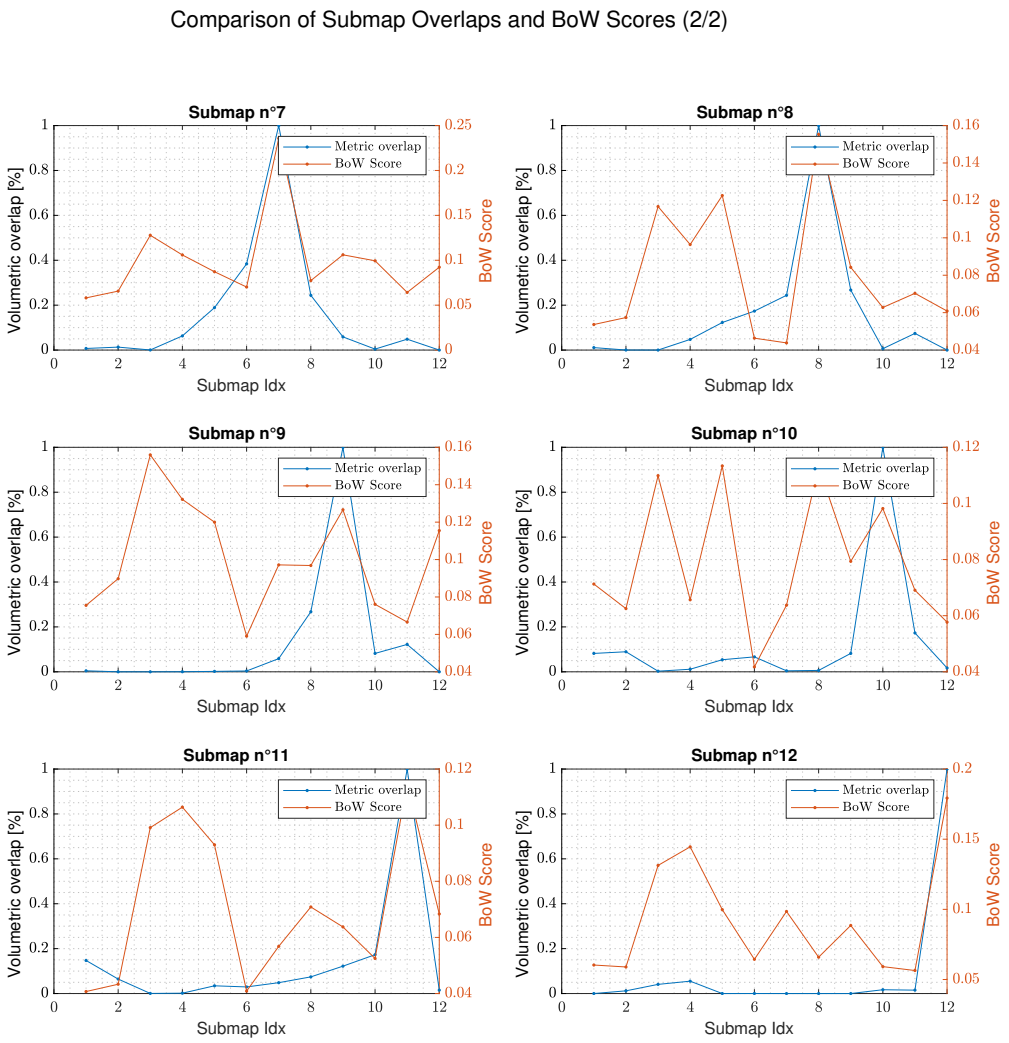


Figure 3.29: Comparison of the bag-of-word scores between submaps and true spatial correlation. Submaps in the mapping session shown in fig. 3.27 are tested against each other. The BoW recall is computed as in eq.3.20 without any thresholding scheme applied. Volumetric overlaps are computed as the fraction of 3D neighbors over the total number of points in the submaps pair.

and $L = 10$ (for the meaning of these parameters, the reader is referred to section 3.3.1). In their work, a MSER (Maximally Stable Extremal Regions) keypoint detector [32] is used followed by SIFT feature description. This results in a well structured and descriptive vocabulary. The well-known Visual SLAM algorithm ORB-SLAM [96] is provided with a dedicated vocabulary of ORB feature descriptors whose size is again close to 10^6 elements. Visual information is both fast to retrieve and densely packed in images. From a single image, thousands of unique descriptors can be extracted if enough visual information is present. This allows to easily build a visual vocabulary from a relatively limited amount of data. Point clouds on the other hand such as the submaps here involved, require notable efforts to build and the amount of 3d information that stereo images can deliver is limited. From thousands of images observing a restricted environment, especially comprising natural features, only little numbers of 3D descriptors can be extracted. This challenge both the uniqueness of the vocabulary that can be built as well as its size, which is usually limited. In this section, the effects of re-weighting 3D words are investigated. Figure 3.30 provides an overview of the tasks involved in this evaluation.

Given two input submaps belonging respectively to the relocalization and training stage (blue and red), the procedure of extracting binary descriptors and transforming to bag of words vectors is identical. These tasks involve the same parameters such as support region radiuses for normals and descriptors extraction as well as the vocabulary parameters. In particular, the threshold H_{thresh} on the Hamming distances between descriptors and leafs (eq. 3.23) is applied to both submaps. This parameter influences the final values contained inside each BoW vector, affecting the final pair-wise similarity scores. This effect is then *systematic* for each session. After scoring each of the bag-of-words pairs, it is needed to select a threshold value such that all scores higher than this threshold are considered correct, otherwise false. This value is the relative threshold t_{rel} in eq. 3.26. Modifying these two parameters, H_{thresh} and t_{rel}, affect the distribution of true and false submap matches.

The performances of this relocalization pipeline can be interpreted as those of a *classifier* of correct matches using a *precision-recall* test. Submaps pairs can belong to two classes, *matching* or *non matching*. As summarized in table 3.2, a ground truth for the submap

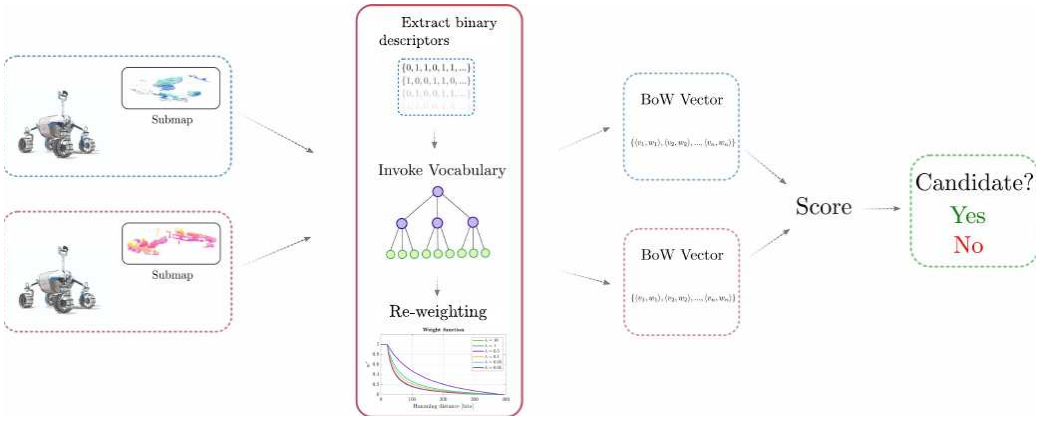


Figure 3.30: Schematic representation of the tasks involved for evaluating the effect of thresholds.

locations deliver information about the *true values*. In other words, the True value is the value that a perfect classifier should be able to predict. The value that is actually predicted by the classifier, which can be mistaken, is the *predicted value* highlighted on the first column. If the classifier commits no errors, each of the predictions are either true positives (TP) or true negatives (TN), meaning that the classifier predicts correctly if two submaps match or not. When the classifier is wrong, it can either trigger a match between non-matching submaps (FP) or not trigger a match when two submaps actually match (FN).

To score the performances of such classifier, two metrics are particularly useful: *precision* and *recall* [104]. Precision is defined as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3.39}$$

and describes the predictive accuracy of the classifier. In relevant words to this evaluation, this metric contains information about how many of the predicted matches are actually correct. A value of 0 would describe a classifier that just predict wrong values. On the contrary, a value of 1 describes a perfect classifier. This metric however does not include information about the completeness of the prediction: positives which are not detected by the classifier do not influence the precision score. For such information, the *Recall* metric is useful and is defined as:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.40}$$

The denominator of the Recall metric involves all the true values that the classifier should detect. It is then a measure of how many of the true values are actually detected by the classifier, regardless of the false positives. These two metrics are visualized at the same time through the *Precision-Recall* curve. The threshold $t_{rel} \in [0, 1]$ is varied along equal intervals from the extrema of its domain in order to evaluate the performances of the relocalization pipeline. Just as a reminder, this relative threshold acts as a discriminator of similarity between submaps in terms of bag-of-words scoring: let be $\mathbf{v}_i$ and $\mathbf{v}_j$ bag-of-words vectors of $\mathbf{s}_i$ and $\mathbf{s}_j$:

$$s(\mathbf{v}_i, \mathbf{v}_j) > T(t_{rel}) \ \rightarrow \mathbf{s}_i \text{ and } \mathbf{s}_j \text{ match} \tag{3.41}$$

where $T(t_{rel})$ is defined in eq. 3.26. A single value for t_{rel} defines a certain behavior of the relocalizer which is translated in a point along the Precision-Recall curve.

In the following sections the candidate selection scheme is evaluated firstly without embedding texture information, therefore using the B-SHOT descriptor to build the vocabulary and bag-of-words vectors. Finally, in section 3.5.5 we highlight the benefits of using our novel B-Tex-SHOT.

IN_OUT_RUN

The classifier performances are evaluated on two multi-robot datasets. The first, denoted as IN_OUT_RUN involves the LRU and LRU2 rovers exploring a partially overlapping environment. Both robots start inside the test environment at the Robotics and Mechatronics

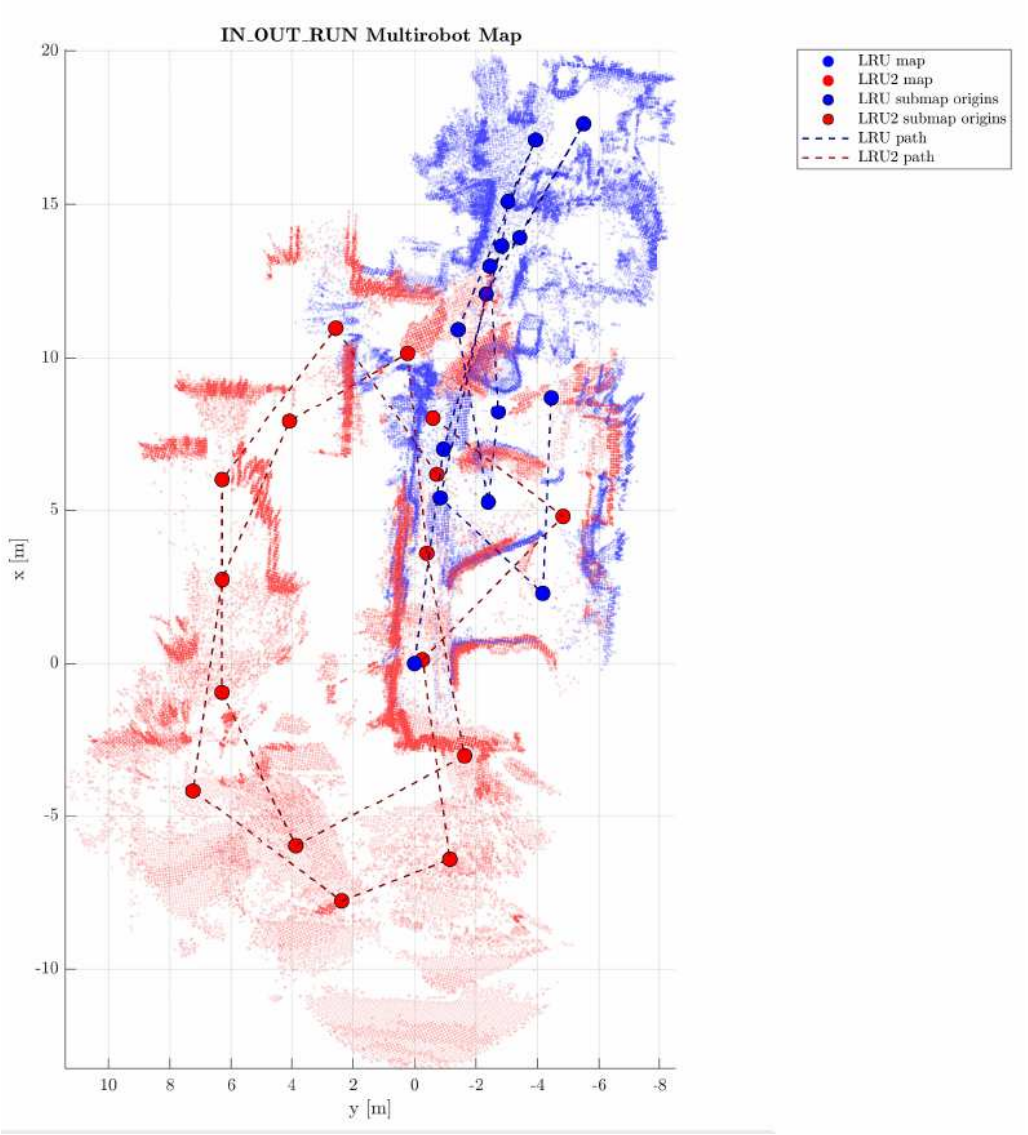
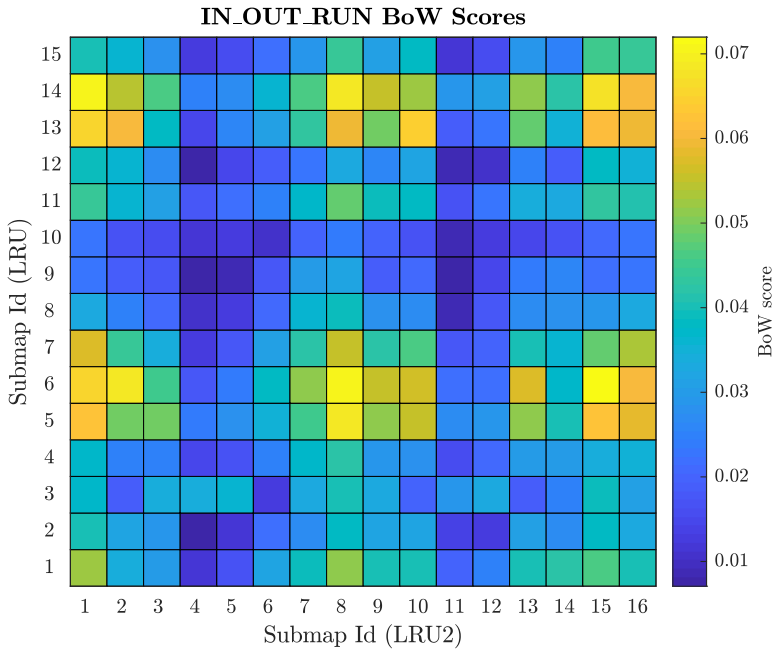
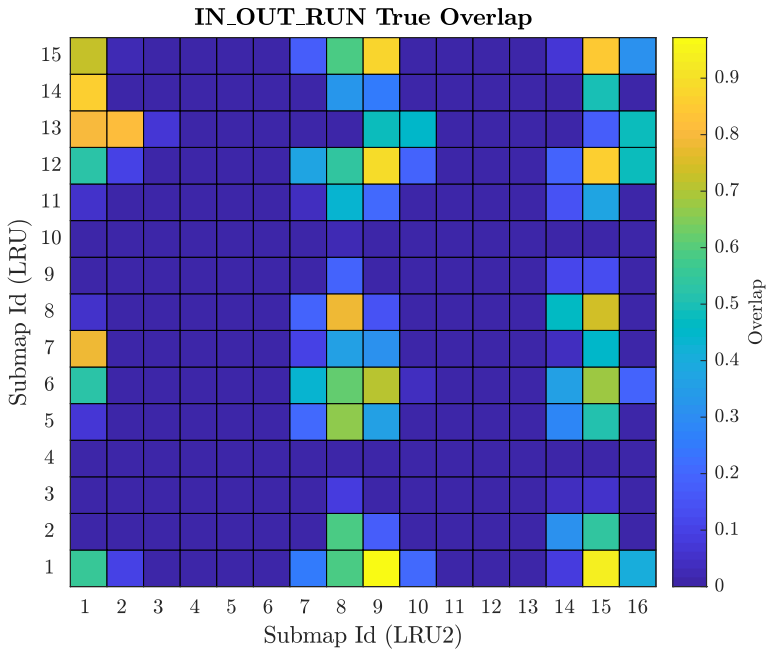


Figure 3.31: Map and submap origins for the IN_OUT_RUN multi-robot mapping session. In this session, LRU explores the laboratory environment (three rocks are visible in the proximity of the coordinates $[-3, 5]$) and a workshop area (upper right) behind the Vicon-covered zone. LRU2 instead explore partially the Vicon-covered zone, then moves towards the hallway of the Robotics and Mechatronics center as well as the exterior of the building. Trajectories and maps are manually aligned. As both maps are rigidly aligned, some misalignments are visible due to inaccuracies in the outcomes of the SLAM system.

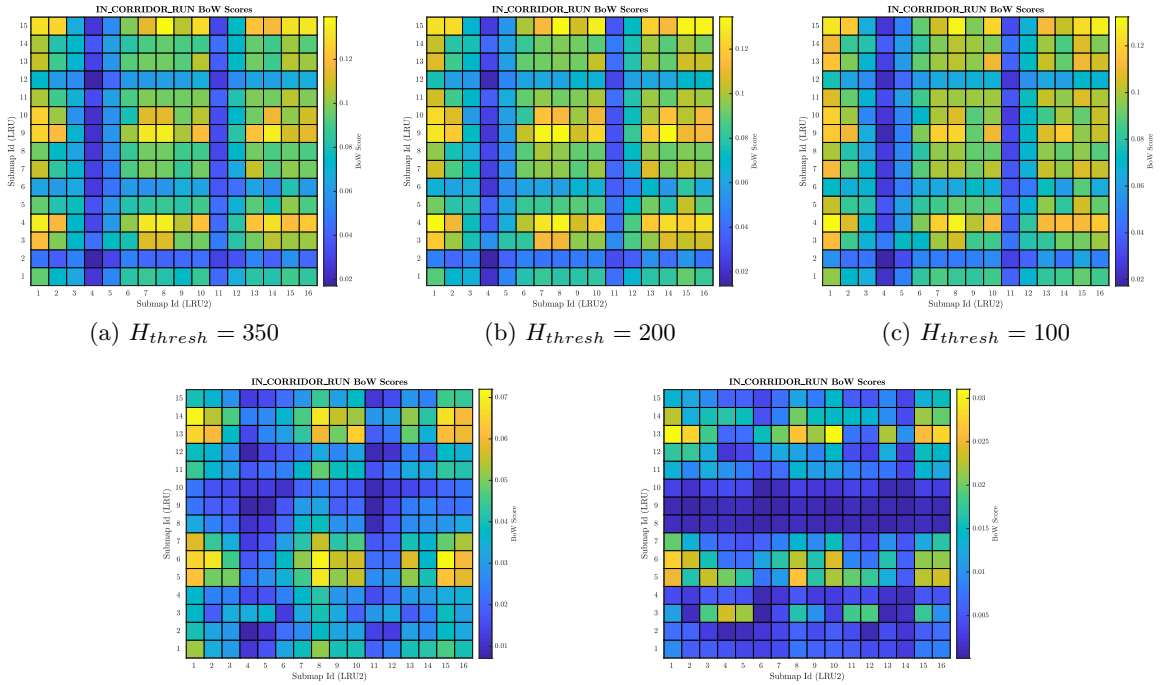


(a) BoW scores - $H_{thresh} = 50$



(b) True overlap

Figure 3.32: Heatmaps representing the BoW scores and true overlap between the LRU and LRU2 submaps for the multi-robot session in figure 3.31. For each of the two plots, the x direction contains the indexes of submaps from the *relocalization* session performed by LRU2 while the y direction encodes the indexes from the training session performed with LRU



(a) $H_{thresh} = 350$ (b) $H_{thresh} = 200$ (c) $H_{thresh} = 100$

(d) $H_{thresh} = 50$ (e) $H_{thresh} = 20$

Figure 3.33: Heatmaps of BoW recalls as in figure 3.32a where the Hamming threshold for bag-of-words vector creation is varied in 5 decreasing values spanning from the maximum 352 to 20. This figure shows how decreasing the value for H_{thresh} helps discriminating good candidates.

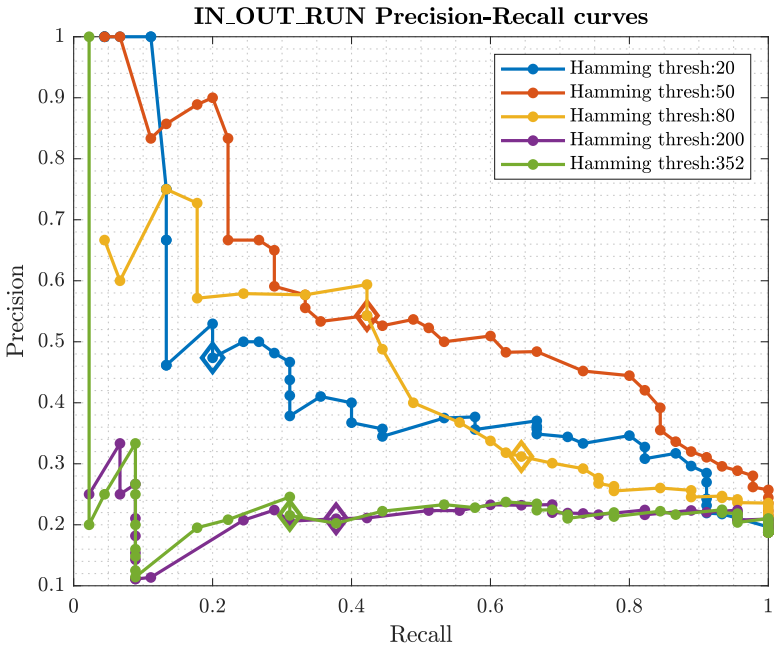


Figure 3.34: *Precision-Recall* curve for the IN_OUT_RUN session. Each curve is plotted for different values of H_{thresh} which significantly affect performances. PR curves are defined by increasing t_{rel} from 0 (lower right part) to 1 (upper left part). The diamond marker on each of the curve represents the t_{rel} value of 0.7, increasing t_{rel} generates the upper left side of the curves. Heatmaps for the curves reported in this graph can be observed in figure 3.33

center where Vicon coverage is available. Maps and trajectories are visible in figure 3.31: blue and red colors are related respectively to LRU and LRU2. During the run, both rovers use their stereo SLAM system (section 3.5.1) in order to build a consistent map and store their submaps to be processed *offline*. Here, only classifier performances in terms of precision-recall are evaluated, timings and run-time performances are reported in the following sections. LRU explores part of the test environment and the workshop (upper right area). LRU2 instead explores briefly the test area and exits the building for some outdoor exploration. In figure 3.31 submaps are plotted in terms of their obstacle content. The more densely colored zones represent quasi-vertical surfaces which are actually the detected obstacles where descriptors are evaluated. The submap count is 15 for LRU and 16 for LRU2. In order to align the two maps for ground truth, the available Vicon data is used.

In these experiments, LRU submaps are used for building the vocabulary while LRU2 submaps are used for relocalization. Figure 3.32a reports, in form of a heatmap, the bag-of-words scores between each pair of submaps from the training and relocalization sets. Higher values reflects similarity between the contained descriptors while low values indicate dissimilarity. Figure 3.32b reports instead a reference value the overlap based on the metric proximity of submaps. These values represent ground truth for the relocalization pipeline, valid candidate matches should at least be submaps actually overlapping. Generally, the opposite is not valid since overlapping submaps might not have 3D similarity rendering relocalization impossible for this pipeline. For this evaluation, true positives are defined by overlaps higher than 0.1. Given a value for t_{rel} therefore, each candidate is a true positive if the true overlap is higher than 0.1 otherwise it is marked as a false positive. Viceversa, high overlaps which do not correspond to relocalization candidates are considered as false negatives. Recalling the *confusion matrix* in table 3.2, all the possibilities in terms of submap recalling are listed here:

$$\begin{aligned}
\{i,j\}\ \text{TP} &\iff s(\mathbf{v}_i,\mathbf{v}_j) > T(t_{rel}) \text{ and } overlap(i,j) > 0.1 \\
\{i,j\}\ \text{TN} &\iff s(\mathbf{v}_i,\mathbf{v}_j) < T(t_{rel}) \text{ and } overlap(i,j) < 0.1 \\
\{i,j\}\ \text{FP} &\iff s(\mathbf{v}_i,\mathbf{v}_j) > T(t_{rel}) \text{ and } overlap(i,j) < 0.1 \\
\{i,j\}\ \text{FN} &\iff s(\mathbf{v}_i,\mathbf{v}_j) < T(t_{rel}) \text{ and } overlap(i,j) > 0.1
\end{aligned} \tag{3.42}$$

where the threshold T is computes as a function of t_{rel} as in eq. 3.26. Varying the threshold t_{rel} on the bag-of-words scores, the values for *precision* and *recall* are computed as in eq. 3.39 and 3.40 drawing a curve in the *precision-recall* space. Figure 3.34 reports various curves each related to a given Hamming threshold H_{thresh}. Only thresholds in the set $\{20, 50, 80, 200, 352\}$ are shown to avoid cluttering the plot. Firstly it is evident how changing this threshold vary the precision of the tentative matches: the highest threshold of 352 is the one that gives the lowest performances since for each value of the relative threshold on bag-of-words scores t_{rel} (which defines the curve), the precision is very low. This means that the majority of recalls are false positives i.e. wrong candidates. By lowering the threshold, the precision-recall curve reaches higher values meaning that less false positives are produced. Notably, the highest curve is obtained using a Hamming

threshold of 50 which allows some flexibility in building bag-of-words vectors. By imposing a much lower Hamming threshold, the contributions of the extracted descriptor in each submap is too weak, suggesting that noise sources in computing the descriptors likely leads to false associations. Relative thresholds t_{rel} around 0.7 in conjunction with H_{thresh} around 50 leads to the best performances in terms of recall accuracy (upper left corner in figure 3.34). These values are generally confirmed also by further experiments such as the one that follows.

IN_CORRIDOR_RUN

A second multi-robot mapping session is used to evaluate the relocalizer precision is selecting candidate matches. This session involves the exploration of the laboratory environment also explored in the IN_OUT_RUN as well as in the session shown in figure 3.27. The rover then proceed to explore the corridors in the first floor of the Robotics and Mechatronics building. Their trajectory (and 3D maps) overlap inside the laboratory as well as along two corridors. Figure 3.35 show the maps of both LRU and LRU2, again the red and blue colors denote the two agents. As for the previous test session, the map is represented for the 3D content classified as *obstacle* and colored dots represent the Local Reference Frame of each submap (see 3.5.1). A black dashed line connects each consecutive submap origin to reconstruct the rovers trajectories. Both maps are here aligned manually since Vicon tracking is available only for a very limited region of travel inside the laboratory environment. Drifts in the localization and mapping system however make the trajectories diverge slightly, especially during the 90 degrees turns between the intersecting corridors. Visual alignment was the best way to average drifts and allow to compute reference overlaps between submaps (figure 3.36b). Figure 3.36a represents again in a heatmap form the bag-of-words scores between each submap from the LRU and LRU2 mapping sessions. All the pair-wise scores are computed using a Hamming threshold parameter $H_{thresh} = 50$. Compared with the reference overlap given by spatial proximity, the matching regions detected by high bag-of-words recalls are concentrated in the beginning and end of the trajectories: those are the regions inside the laboratory environment where actual 3D information is discriminative of the scenario. The overlaps in the corridors, highlighted as diagonal traces in figure 3.36b are not detected by BoW recalls. This is expected since no useful information is provided by those submaps, where obstacle clouds are just two narrow bands representing the side walls, therefore ambiguous and not informative enough.

Figure 3.38 report the *precision-recall* curves generated for multiple values of H_{thresh} and t_{rel}. As for the previous session, t_{rel} is varied from the minimum observed bag-of-words score *bestMin* to the highest observed score *bestMax* and diamond markers represent the value 0.7 along the curves. Also in this session it is evident how increasing the value of the Hamming threshold helps in increasing the accuracy of the classifier, doubling the *precision* score for most of the *Recall* values by switching to a H_{thresh} of 50. A decrease in performances is observed while lowering the threshold to 20 since most of the valid candidates contain non perfectly matching binary SHOT descriptors and are therefore neglected by BoW scoring with such a low threshold. Curiously, with Hamming thresholds of 200 to 352, the *Precision* drops to 0 for higher t_{rel} thresholds, this is most likely due

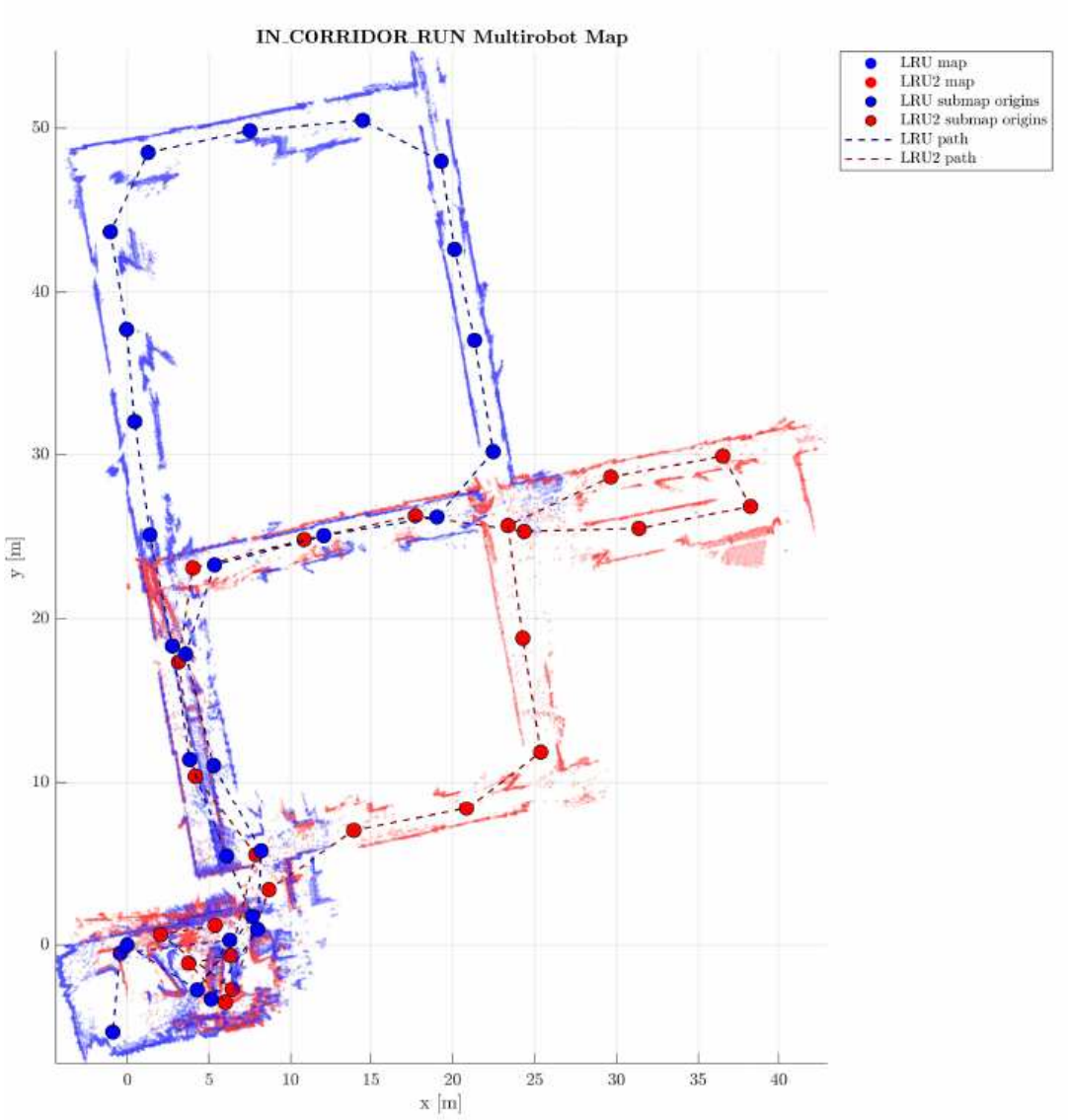
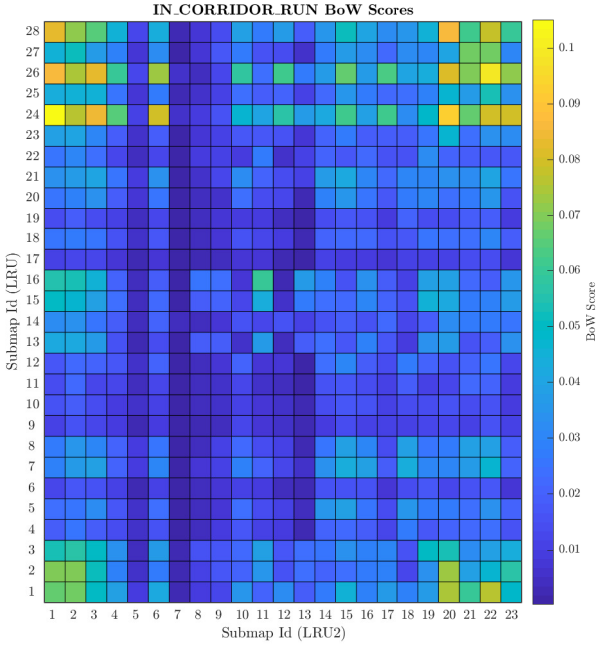
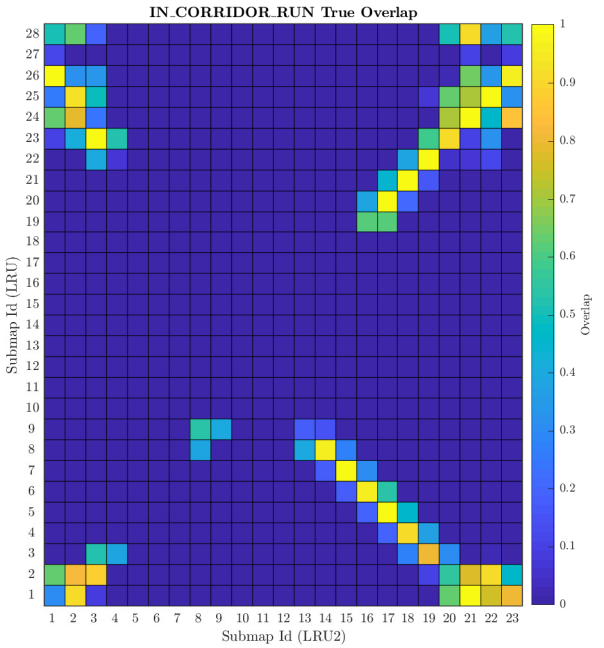


Figure 3.35: Map and submap origins for the IN_CORRIDOR_RUN multi-robot mapping session. In this session, the LRU and LRU2 rovers explore both the laboratory environment visibile in the lower left as well as the first floor corridors of the Robotics and Mechatronics center. Their trajectories and maps partially overlap on the corridors sector and fully overlap inside the laboratory. In the corridors, the rovers view captures only a few centimeters of side wall, while the majority of the field of view comprises the ground. Only the point cloud in the laboratory (lower left) contain usable 3D information for relocalization. The total submap count for LRU is 29 while for LRU2 is 23. Vicon coverage is available only inside the laboratory, which constitutes a minimal part of the travelled ground, therefore the two sessions are manually aligned

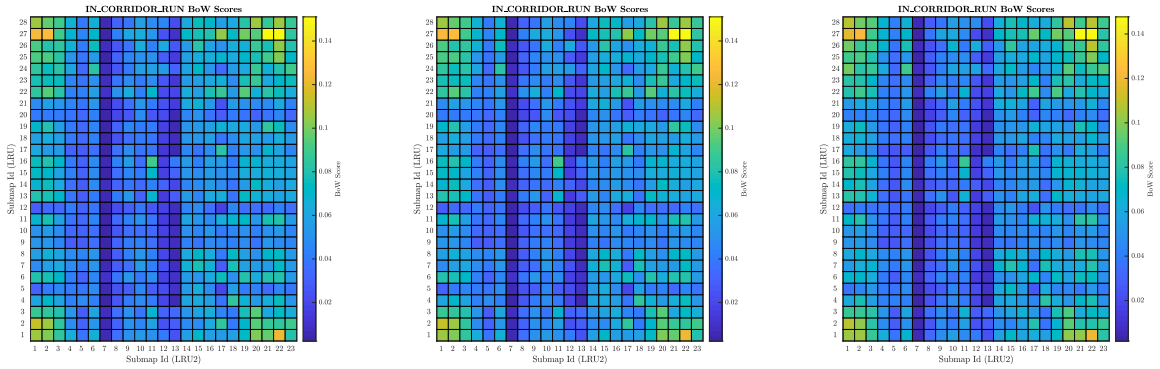


(a) BoW scores - $H_{thresh} = 50$

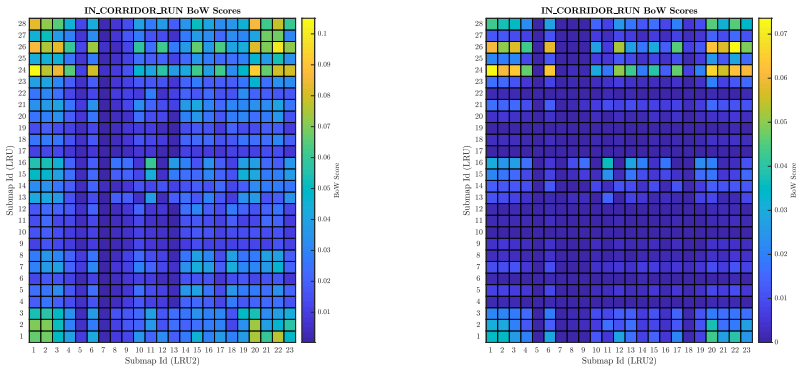


(b) True overlap

Figure 3.36: Heatmaps representing the BoW scores and true overlap between the LRU and LRU2 submaps for the multi-robot session in figure 3.35. For each of the two plots, the x direction contains the indexes of submaps from the *relocalization* session performed by LRU2 while the y direction encodes the indexes from the training session performed with LRU



(a) $H_{thresh} = 350$ (b) $H_{thresh} = 200$ (c) $H_{thresh} = 100$



(d) $H_{thresh} = 50$ (e) $H_{thresh} = 20$

Figure 3.37: Heatmaps of BoW recalls as in figure 3.36a where the Hamming threshold for bag-of-words vector creation is varied in 5 decreasing values spanning from the maximum 352 to 20. This figure shows how decreasing the value for H_{thresh} helps the selection of good candidates.

to the presence of too many high recalls for submap pairs, most of which are wrong. It is then very likely than only very few wrong matches are selected causing the precision to drop to zero. This effect is confirmed also by figure 3.34, strongly motivating the need of a thresholding scheme as the proposed one. To complement figure 3.38, also figure 3.37 reports the full heatmaps for different values of H_{thresh}.

3.5.5 Benefits of texture-enriched descriptors

Figure 3.39 shows the precision-recall curves on the IN_OUT_RUN and IN_CORR_RUN datasets where the candidate selection scheme uses B-Tex-SHOT instead of the structure-only B-SHOT. In this case the maximum allowed value for the Hamming threshold H_{thresh} is 396, which is the descriptor length. The curves are computed again by varying t_{rel} from 0 to 1 with and without texture (therefore B-SHOT vs B-Tex-SHOT) on the same Hamming thresholds. Generally, for H_{thresh} close to 80 and 100, employing texture increases the precision of the classifier. In addition, figure 3.39 reports also a baseline value which is the precision of a random classifier. In this case, the precision is computed as:

$$P_{random} = \frac{n_{positives}}{n_{sub,reloc} \cdot n_{sub,db}} \tag{3.43}$$

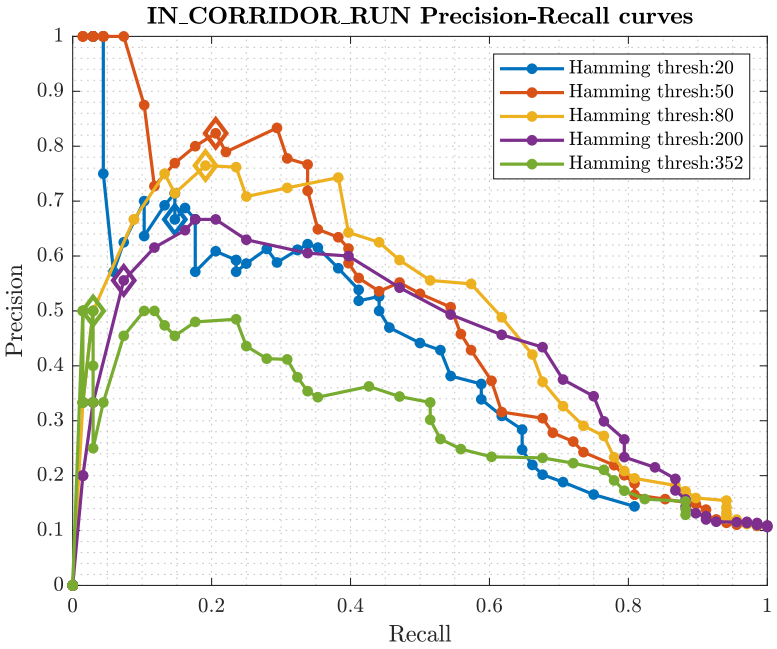


Figure 3.38: *Precision-Recall* curve for the IN_CORRIDOR_RUN session. Each curve is plotted for different values of H_{thresh} which significantly affect performances. PR curves are defined by increasing t_{rel} from *bestMin* (lower right part) to *bestMax* (upper left part). The diamond marker on each of the curve represents the t_{rel} value of 0.7, increasing t_{rel} generates the upper left side of the curves.

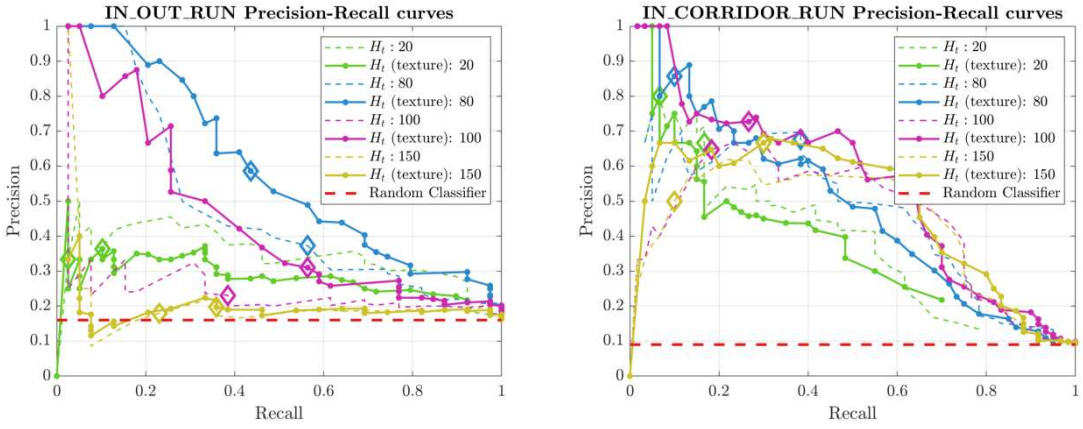


Figure 3.39: Precision-recall curves for the IN_OUT_RUN and IN_CORR_RUN. In this case results are compared with (solid lines) and without (dashed lines) adding texture using the same H_{thresh} for both cases. A red dashed line shows the outcomes of a random classifier to highlight baseline performances.

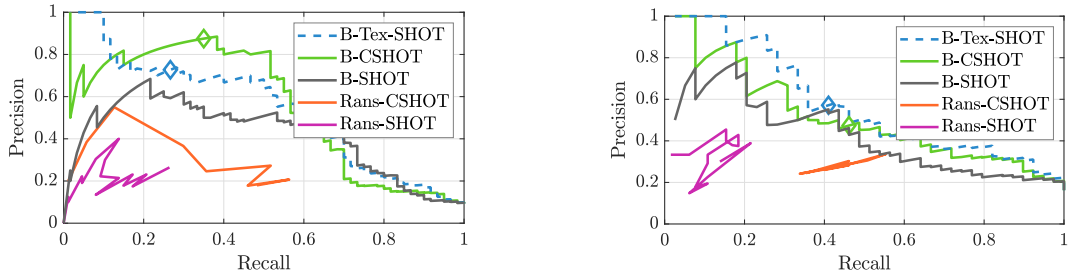


Figure 3.40: Precision-recall curves for IN_OUT_RUN and IN_CORR_RUN. In this figure we evaluate the benefit of using B-Tex-SHOT instead of a color SHOT (C-SHOT) binarized extending the approach in [107]. Here we also compare the performances with a common RANSAC validation approach, which computes a consensus on the roto-translation between every submap pair. If RANSAC exits with a transformation model then the submap match is consideres as a positive.

where $n_{positives}$ is the number of correctly matching submaps while $n_{sub,reloc}, n_{sub,db}$ are the number of submaps in the first and second dataset therefore their product is the total number of possible matches.

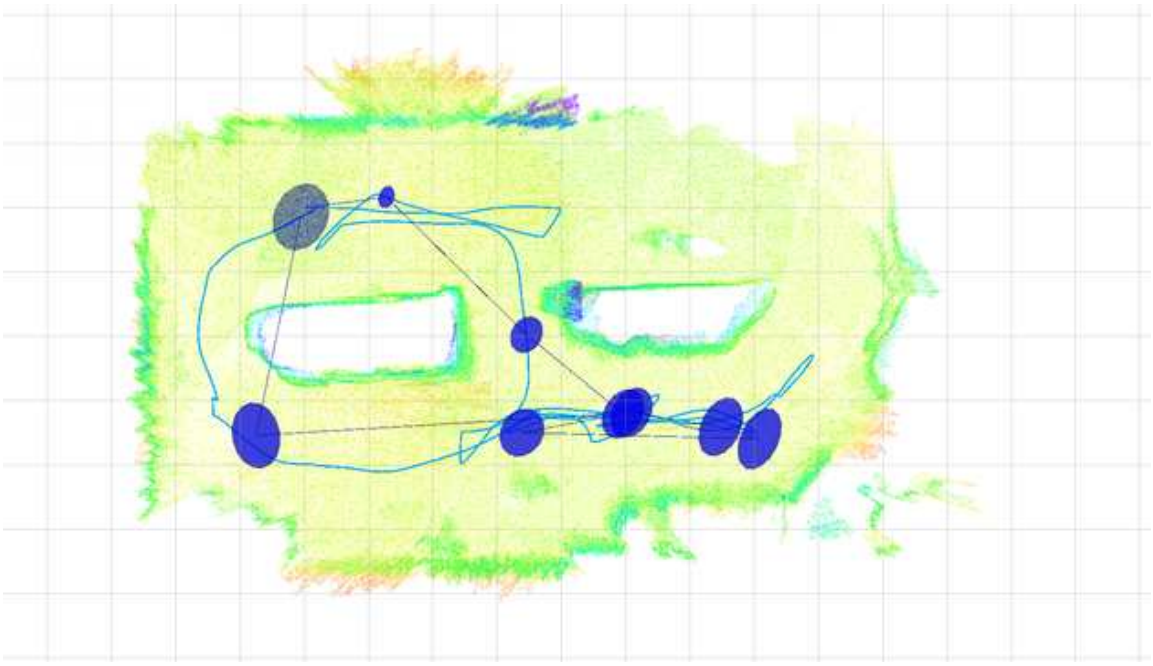
Comparison with RANSAC

In figure 3.40 we evaluate how our approach stands in comparison with common RANSAC-based approaches, motivating the impossibility of using RANSAC discussed in 3.4.2. To extract a candidate match using RANSAC we first match both SHOT and C-SHOT (color SHOT) between every possible pair of submaps from the two sessions. Then a RANSAC registration algorithm is used[5] to align the submaps rejecting wrong descriptor matches and labeling them as outliers. For the RANSAC tests the probability parameter P is set to 0.99. P correspond to the probability of selecting a rototranslation model given the number of descriptor matches and determine the number of iterations needed. RANSAC curves are generated by varying not the algorithm parameters but the descriptor matching thresholds instead. Both SHOT and C-SHOT are matched by minimizing L_2 distances and selecting those for which the distance is lower than a threshold. As both descriptors are normalized on their norm, the possible distances belong in the interval $[0, 1]$. The curve denominated B-CSHOT is instead built by replacing the B-SHOT descriptor with a binarized C-SHOT adapting the method used for SHOT to a longer descriptor. The figure shows that the performances between B-Tex-SHOT and B-CSHOT are very similar, therefore motivating the usage of a shorter compact descriptor as B-Tex-SHOT (of length 396) instead of B-CSHOT which has length 1344. The precision of RANSAC is lower than our approach in every case, failing to discriminate inlier from outlier keypoint matches.

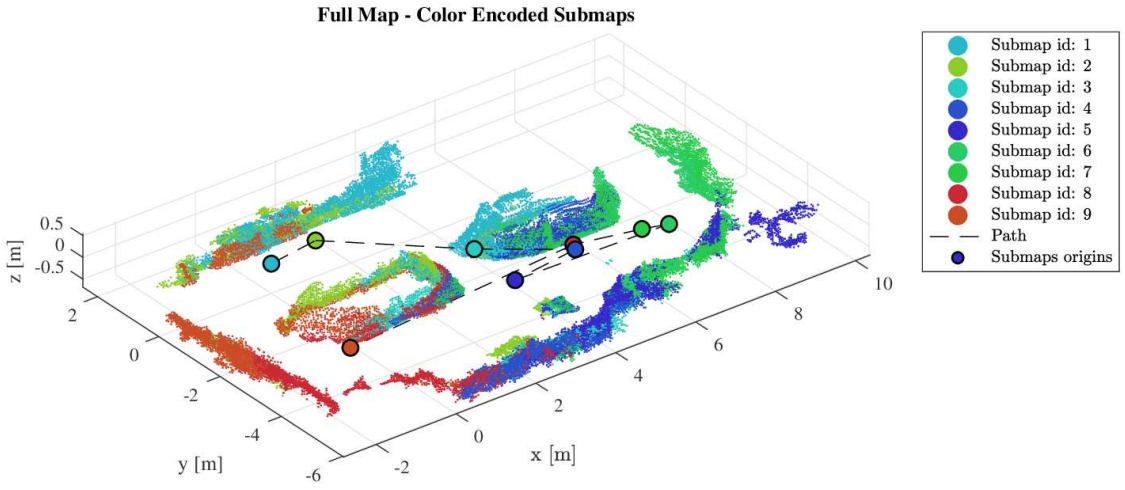
3.5.6 Indoor Multi-session relocalization

Having evaluated the performances of the relocalization candidate selection scheme (section 3.5.4), the full pipeline is here evaluated in the context of a multi-session mapping scenario.

[5] http://pointclouds.org/documentation/tutorials/random_sample_consensus.php



(a) RVIZ view

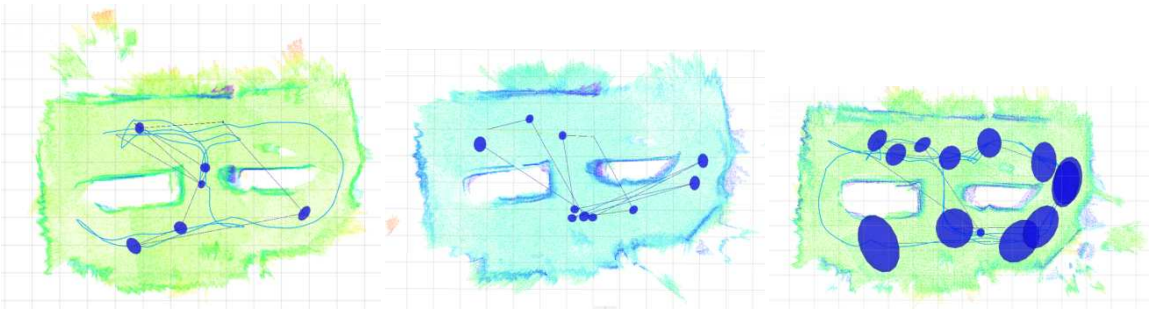


(b) Submaps view

Figure 3.41: SLAM results for the database run of the multi-session experiment IN_RUN_DB. Figure (a) reports the outputs of the RVIZ visualizer provided by the Robot Operating System. The full 3D point cloud is shown color-mapped to highlight z-axis coordinates (or elevation). Blue ellipses represent the position of each submap with the uncertainty estimated from the iSAM2 optimizer. The full trajectory of the rover is shown (*unoptimized*) with a solid blue line. The light grid has size 1 meter-. Figure (b) highlights instead the division in submaps of the same SLAM session. Each dot represents the Local Reference Frames which constitute the origin of each submap and are associated to each respective point cloud using the same colors.

	Database	Relocalization
new_submap_distance [m]	**7.0**	**9.0**
new_submap_rotation [deg]	480	480
new_submap_position_σ [m]	0.2	0.2
new_submap_orientation_σ [deg]	5.0	5.0

Table 3.3: Submap trigger params



(a) IN_RUN_1 (b) IN_RUN_2 (c) IN_RUN_3

Figure 3.42: 3D maps, trajectories and Local Reference Frames for the relocalization sessions IN_RUN_1, IN_RUN_2 and IN_RUN_3. The figures are related to single mapping sessions, where SLAM was run on LRU data during multiple explorations of the same recreated natural environment. The figures highlights how the three SLAM sessions are characterized by different submaps both in number and therefore appearance. As an example, IN_RUN_3 have more submaps in the upper right side than the others. For metric reference, grid cells are 1 meter.

The LRU is driven in the laboratory environment, the same used for the preliminar investigation in section 3.5.3. The environment is prepared with the rock in similar configuration to what can be observed in figure 3.27. This multi-session mapping experiment comprises one database run, denominated IN_RUN_DB, and three consecutive mapping sessions for relocalization performed in a time frame of two days. This way, slight changes can appear in terms of environment structure and lighting. The full map as well as submap division respective to the IN_RUN_DB is reported in figure 3.41. This database run comprises 9 submaps, and observes partially the test environment. Relating again to figure 3.41, the upper right side of the map lacks infact some structural detail which is actually present in some relocalization runs. In order to further perturbate the conditions of the relocalization scenario, aside from capturing other sessions with different travelled paths and lighting, also the parameters for submap switching were changed such that the submaps will differ also in size from database and query. Table 3.3 reports the parameters used by the submap switching scheme in the different sessions, only the *new_submap_distance* parameter was changed since it was the most influent in this scenario.

Figure 3.42 show the outcomes of the 3 individual SLAM sessions that constitute the *query* set, or the *relocalization* set. The three runs are denominated IN_RUN_1, IN_RUN_2 and IN_RUN_3 and the differences in submaps, environment and coverage of the area are visible in the figure. As an example, the sessions IN_RUN_1 and IN_RUN_3 have coverage of the upper right side of the map which is lacking in some parts in IN_RUN_2 as well

Run name	n_submaps	captured	area covered $[m^2]$	time duration
IN_RUN_DB	9	-	~ 96	
IN_RUN_1	8	same day	~ 96	
IN_RUN_2	11	same day	~ 96	
IN_RUN_2	13	next day	~ 96	

Table 3.4: Summarized parameters of the database and relocalization sessions

as the database run IN_RUN_DB. The left side of all obstacle maps also might exhibit differences as it is represented by a tent covering the glass walls of the building. Also note that in the run IN_RUN_1 the rover does not travel completely around the left rock while it does in IN_RUN_3. All these maps also show various level of metric accuracy in the 3D reconstructions, several slight misalignments are visible and enhanced by the top-down viewpoint. Different levels of uncertainty over the submap origins are observed since different loop closures where detected. In run IN_RUN_1 a single loop closure is performed, in IN_RUN_2 two loops are detected and none in IN_RUN_3. Loop closures, or submap matches, are visible in figure 3.42 as yellow lines connecting submap origins. However, for an overview of the pose tracking performances of the individual SLAM system the reader is referred to [122]. In addition, apart from pose accuracy not being the focus of this chapter, it is not relevant to relocalization performances since this pipeline relies on a submap basis. In other words, finding relocalization candidates and validating matches is not related to the global pose of submaps. As far as the ground truth is concerned, with the available Vicon tracking system, a true path for the rover is available during its individual SLAM session but not for consecutive sessions. For this reason, in order to have a reference transformation from each database-query session pairs, the rover started in approximately the same position such that the true alignment between maps should be the identity $\{x, y, z, \phi\} = \{0, 0, 0, 0\}$

Candidate and Validated Submap Matches

As the environment observed in this dataset is small and rich of 3D information, only B-SHOT are used to evaluate the effectiveness of a basic configuration. The usage of B-Tex-SHOT is investigated in the following sections where the scenario is more complex and the pipeline benefits from an increased classifier precision. Figure 3.44 show the results of the relocalization pipeline in terms of both selected candidate matches (section 3.4.1) and validated matches (section 3.4.2) for the run IN_RUN_1. Each row contains the results of the matcher by varying the relative threshold t_{rel} in the set $\{0.3, 0.5, 0.7\}$. For every run also, the Hamming threshold H_{thresh} is always set to 50 given the findings on section 3.5.4. This is done in order to verify how changes in thresholds affect the number of candidate and validated matches. This aspect was previously investigated with the *precision-recall* curves in figures 3.34 and 3.38, however this time it is shown how increasing the thresholds might reduce the number of candidates passed to the validation stage but also the number of validated matches as well. As previously stated infact, not always the bag-of-words recall based on descriptor similarity is related to the fact that submaps actually match,

and good matches can then be lost. The left and right columns of figure 3.44 report the candidate matches (red tiles) and the set of validated matches (green tiles) over the bag-of-words recalls and true metric overlaps respectively. It can be observed how increasing the threshold t_{rel} leads to less selected candidates (red tiles) but also less validated matches which can then be used to align the two maps. For this run also the bag-of-words recalls and true overlaps matched quite accurately since between the left and right columns the color tiles exhibit similar patterns. In all of these run, the full pipeline was able to relocalize, since even with very selective thresholds, one match was found.

Figure 3.45 shows the relocalization results for the run IN_RUN_2, where a run of 11 submaps is relocalized over the database. The same considerations made in the context of the previous run apply also here. In this case, however, increasing the threshold t_{rel} have a bigger impact over the proportion of validated matches over total candidate matches. For a value of $t_{rel} = 0.3$ almost half of the possible submap pairs were considered as good candidates, which leads inevitably to wasting computational resources. By increasing the threshold to 0.7 many of the false matches were discarded and a very high proportion of validated matches to candidate ones is achieved. Still, for query submap of Id 2, a high recall was experienced with most of database submaps even if not one of them was validated. As the pipeline ends when presenting a transformation for relocalization of the full map, figure 3.43 shows a simplified representation of the transformation clusters. Only the translational components are shown here neglecting the yaw contribution to the cluster. As no false positives in relocalizing were observed, all transformations are contained in a single cluster. The cluster center is close to the origin of the 3D space because the LRU rover started in very similar locations for all these runs.

Figure 3.46 show the results related to the run IN_RUN_3 where the query set is composed of 13 submaps. Similar considerations can be made also here about the thresholds. In this run as well as the others, a high threshold of $t_{rel} = 0.7$ was sufficient to provide enough validated matches for relocalization. While the three validated matches with submaps 9, 10 and 12 were lost while increasing the thresholds, submaps 1, 3 and 4 were validated for value of t_{rel}. Notably, submap 1 was not validated for the lowest value of t_{rel}, this can be explained by the non-deterministic nature of the validation scheme, where RANSAC is involved and therefore introducing randomness in the results.

Computational Performances

Figure 3.47 reports a detailed analysis of the time required for performing each step of the relocalization pipeline. The three plots refer to the three relocalization session IN_RUN_1, IN_RUN_2 and IN_RUN_3. Additionally, each step compares also the computational effort required in the context of the three values for $t_{rel} = \{0.3, 0.5, 0.7\}$. The different tasks which are examined start from ground removal, or obstacle detection, to find 3D regions where to compute descriptors, then keypoint selection and filtering as well as SHOT descriptor extraction. The combination of these steps can be interpreted as a *preprocessing* phase of handling each submap. Preprocessing is followed by *recall* tasks which involve binarization of the SHOT descriptors, computing BoW vectors and scoring BoW vectors with the database maps to generate candidates. The final step is the *validation* stage which

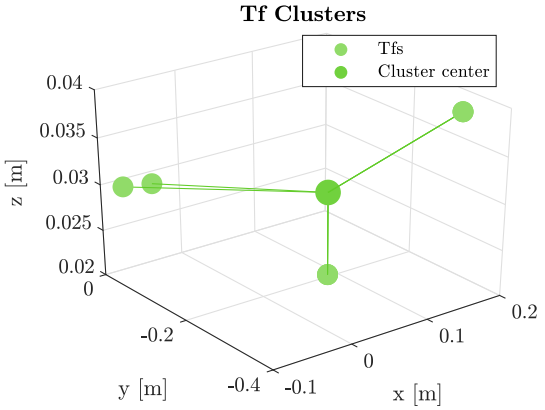
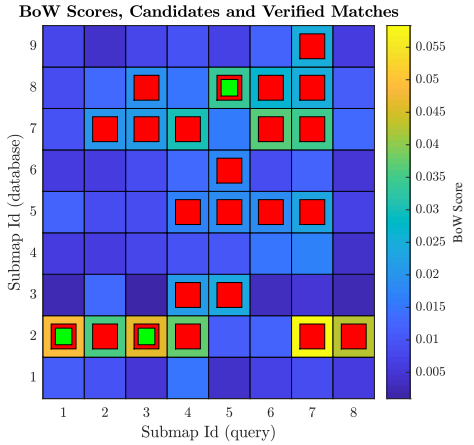


Figure 3.43: Visualization of the 3D coordinates of the winning transformations between validated submap matches for the run **IN_RUN_2**. The yaw value is neglected in this visualization. The 4 matching submaps are related to the t_{rel} of 0.7 in figure 3.45.

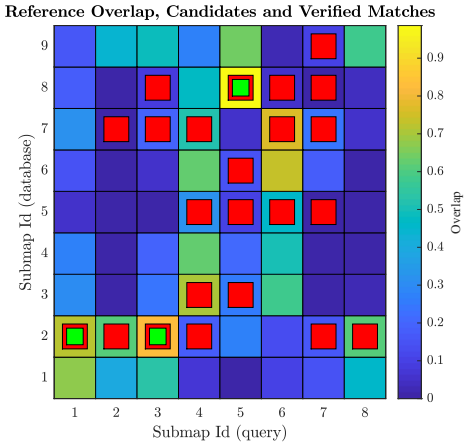
involves just validating each candidate match and growing a consensus of transformations. Figure 3.47 reports timings in logarithmic scale in order to provide an easily understandable comparison of the contributions of each tasks in the global timing budget of the relocalization thread. From the preprocessing stage, the most demanding task is the extraction of SHOT descriptors, which can require in average around half a second for submaps with 1000 to 2000 keypoints. Keypoint detection and filtering require a negligible computational effort as it is implemented as a simple voxel grid approach, however as already pointed out in figures 3.7 this approach results in a much higher number of descriptors to compute. The time required to compute SHOT descriptors on such a high number of keypoints is however tolerable and beneficial for grasping as much 3D information as possible on each submap. Timings for the *recall* part are very low and they motivate the benefits of implementing the proposed approach with bags of binary words. The highest effort required in this step is for the binarization of SHOT descriptors which requires in average less than 10 milliseconds. Building bag-of-words vectors by traversing the vocabulary requires similar but lower efforts and BoW scoring is negligible. The real bottleneck in this case, as it is for the complete SLAM framework of [122], resides in the match validation stage. As each plot shows, match validation requires in average a few seconds per submaps therefore validating a high number of submaps can be an issue. However, as the boxplot shows for different values of t_{rel}, the proposed re-weighting scheme for candidate selection is effective in decimating the computational time required to validate each query submap versus all the selected database ones.

3.5.7 Multi-robot relocalization

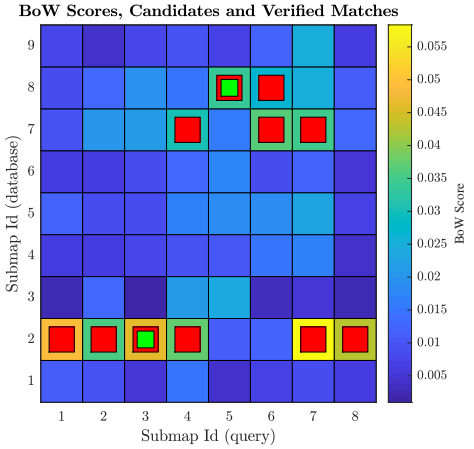
In this section the full relocalization scheme is tested in the multi-robot exploration datasets IN_CORRIDOR_RUN and IN_OUT_RUN, used previously with manually aligned maps to validate the candidate selection scheme from bag of binary words similarity (section 3.4.1). As previously mentioned, in these datasets the LRU and LRU2 rovers explore a mixed indoor environment consisting of both replicas of natural features as well as artificial objects. In the course of their exploration session, the rovers partially share observations of



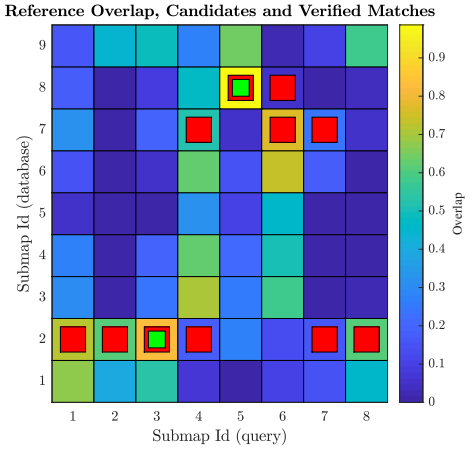
(a) $t_{rel} = 0.3$



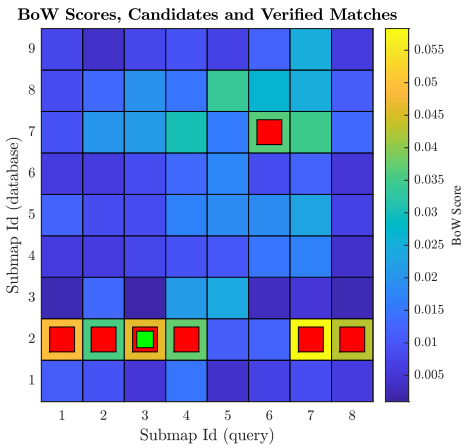
(b) $t_{rel} = 0.3$



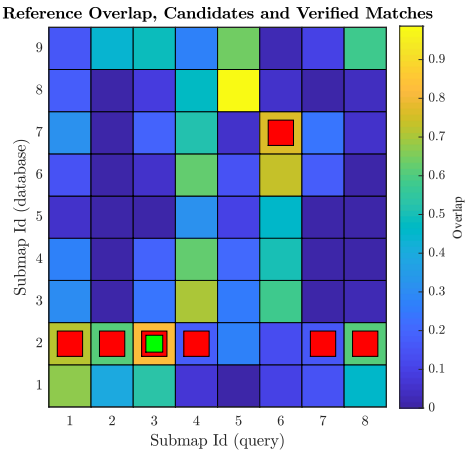
(c) $t_{rel} = 0.5$



(d) $t_{rel} = 0.5$

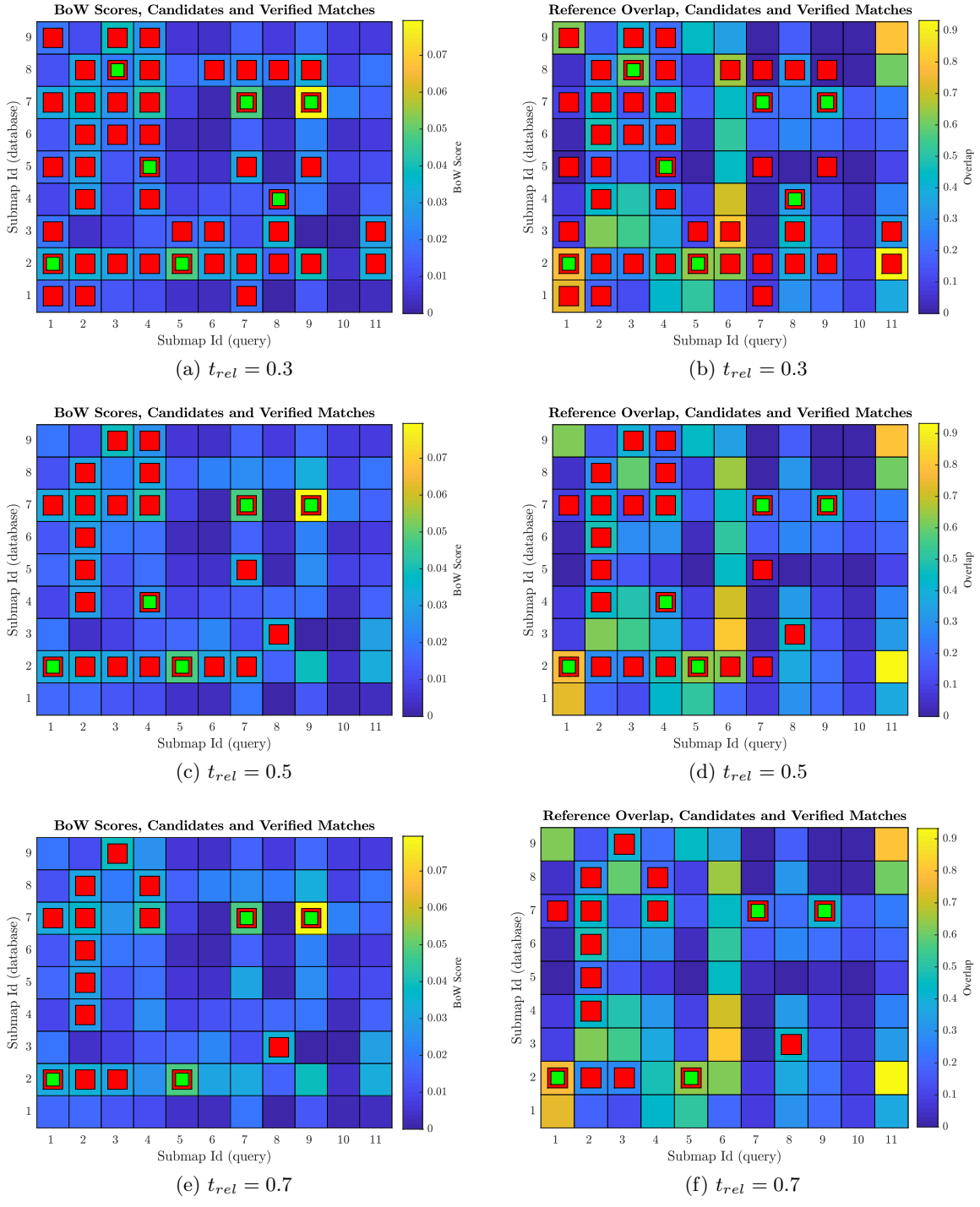


(e) $t_{rel} = 0.7$



(f) $t_{rel} = 0.7$

Figure 3.44: BoW recalls, ground truth overlaps, relocalization candidates and validated for the session **IN_RUN_1**. The three rows refer to three parameter settings in term of t_{rel} for pushing relocalization candidates (red squares). Green tiles denote validated matches from the candidate set.

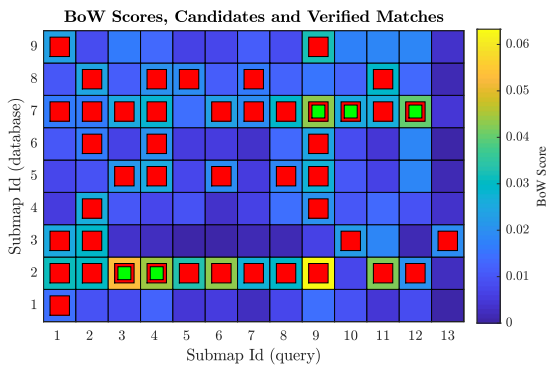


(a) $t_{rel} = 0.3$ (b) $t_{rel} = 0.3$

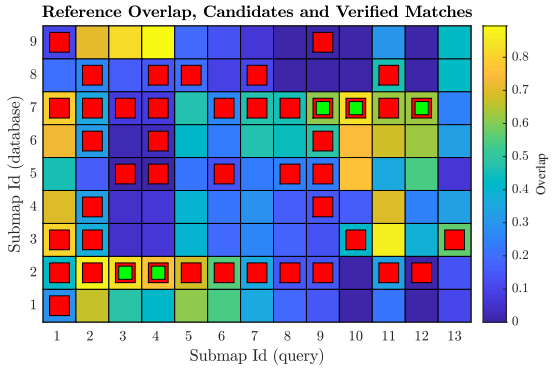
(c) $t_{rel} = 0.5$ (d) $t_{rel} = 0.5$

(e) $t_{rel} = 0.7$ (f) $t_{rel} = 0.7$

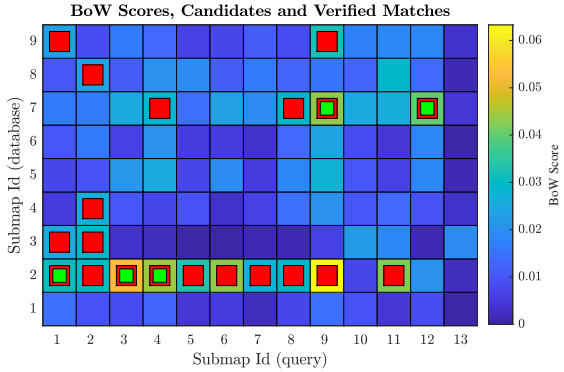
Figure 3.45: BoW recalls, ground truth overlaps, relocalization candidates and validated for the session **IN_RUN_2**. The three rows refer to three parameter settings in term of t_{rel} for pushing relocalization candidates (red squares). Green tiles denote validated matches from the candidate set.



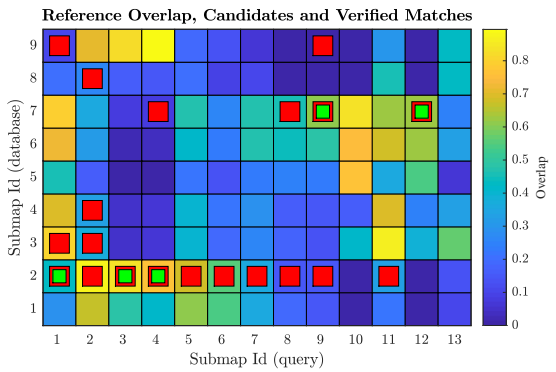
(a) $t_{rel} = 0.3$



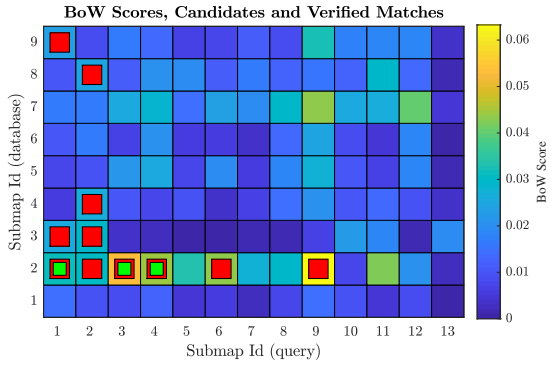
(b) $t_{rel} = 0.3$



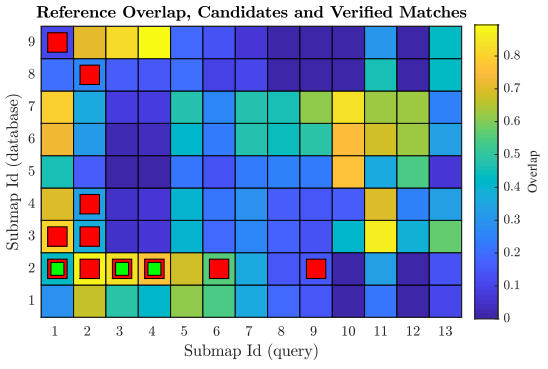
(c) $t_{rel} = 0.5$



(d) $t_{rel} = 0.5$

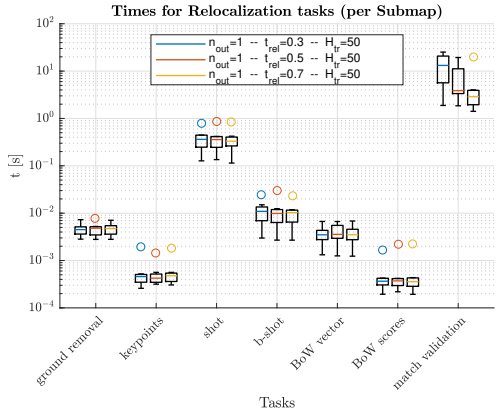


(e) $t_{rel} = 0.7$

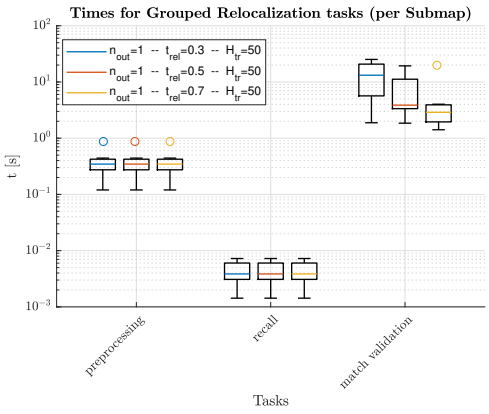


(f) $t_{rel} = 0.7$

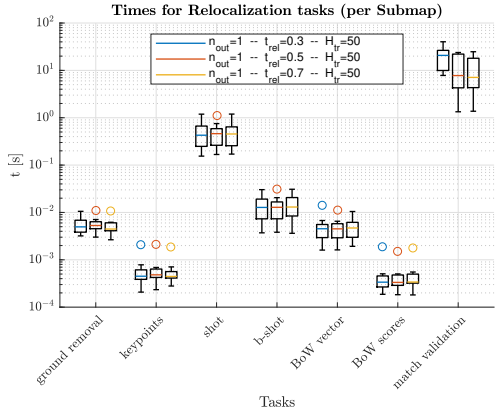
Figure 3.46: BoW recalls, ground truth overlaps, relocalization candidates and validated for the session **IN_RUN_3**. The three rows refer to three parameter settings in term of t_{rel} for pushing relocalization candidates (red squares). Green tiles denote validated matches from the candidate set.



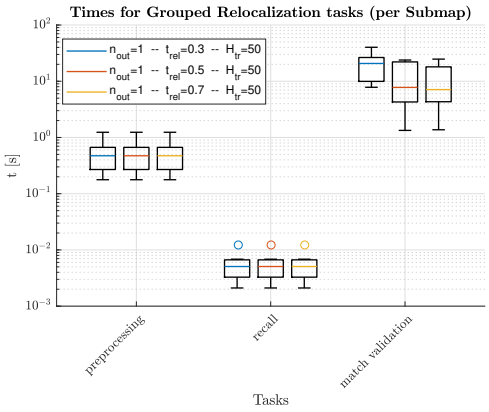
(a) IN_RUN_1



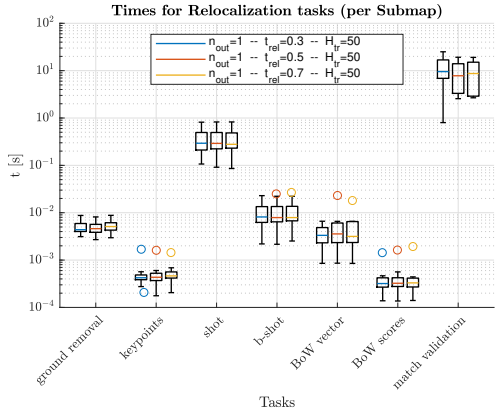
(b) IN_RUN_1



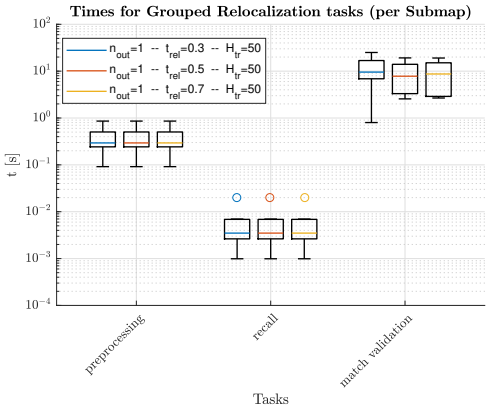
(c) IN_RUN_2



(d) IN_RUN_2

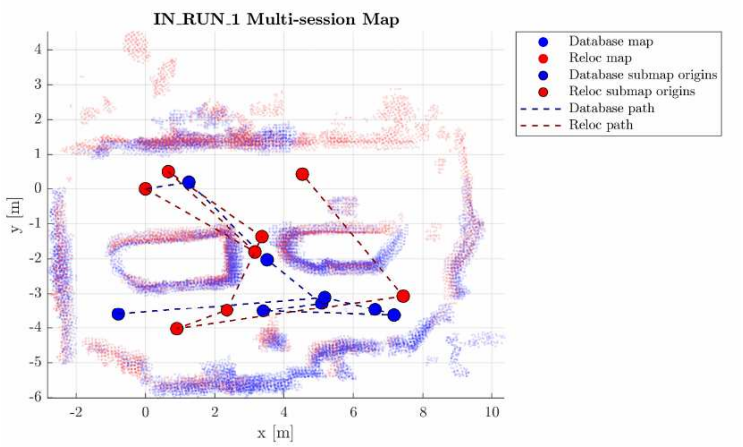


(e) IN_RUN_3

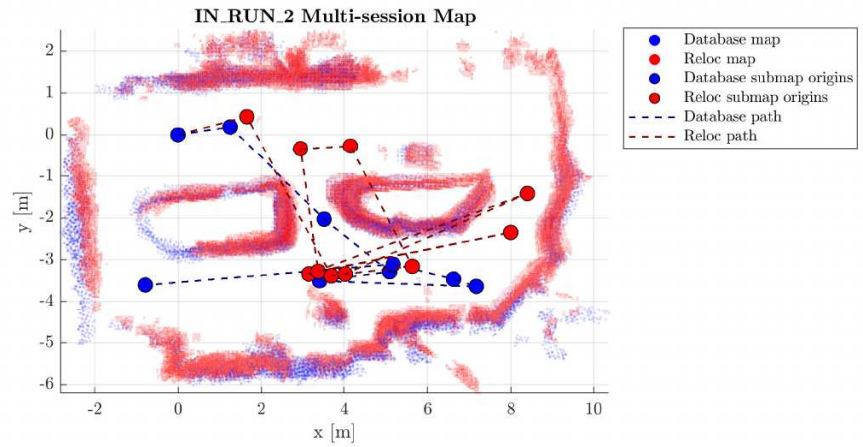


(f) IN_RUN_3

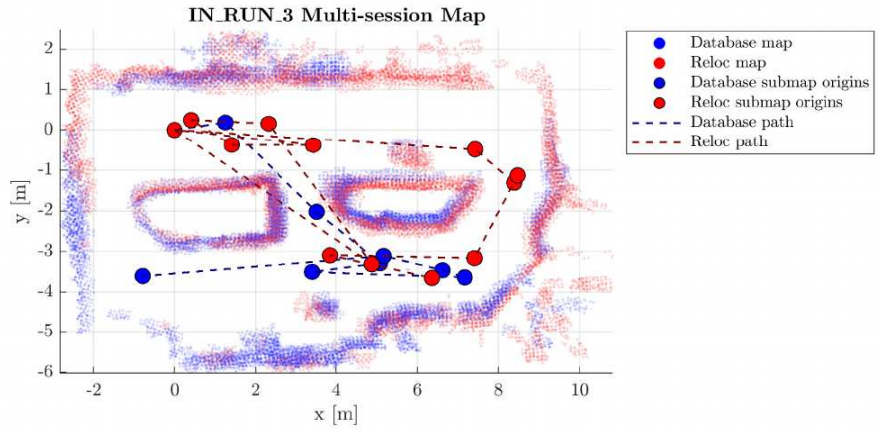
Figure 3.47: Timings for each step involved in the full relocalization pipeline. Each tasks show the different timings also for three different parameters set for the pipeline. H_{thresh} is fixed to 50 while t_{rel} spans the values $\{0.3, 0.5, 0.7\}$ as in figures 3.44, 3.45 and 3.46. Times are in logarithmic scales for better readability



(a) IN_RUN_1



(b) IN_RUN_2



(c) IN_RUN_3

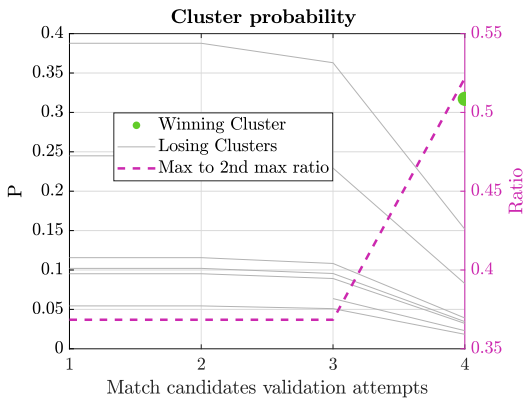
Figure 3.48: Visualization of the relocalization results for the multi-session experiment. The three figures relate to the runs IN_RUN_1, IN_RUN_2 and IN_RUN_3. All three figures show a top view of the obstacle point clouds as well as the submap origins. The red maps are aligned over the database blue map of IN_RUN_DB using the estimated transformation.

the same environment such that relocalization is possible. For this evaluation and for the next one (section 3.5.8) as well, the full B-Tex-SHOT is employed instead of the structure-only B-SHOT. The parameters for the relocalization pipeline have been set accordingly to the recall performances observed in section 3.5.5, therefore $H_{thresh} = 80$ and $t_{rel} = 0.75$ to reduce the computational effort required for match validation. Regarding the Tf clustering scheme (section 3.4.2), the spatial threshold for including candidate transformations in each cluster was set to 1 meter for x, y and z and 10 degrees for the yaw component. The results of the Tf clustering scheme for extracting good transformations between the individual session origins are shown in figure 3.49. Plots in the left column are related to the IN_CORRIDOR_RUN while plots in the right column are related to IN_OUT_RUN. The top plots show how probabilities associated to each Tf hypothesis, which are function of the votes for that cluster (see section 18), vary while multiple submap matches are selected as candidates. Indeed, under the hypothesis that wrong keypoint matches vote for random transformations, if some correct keypoint correspondences are present across one or multiple submap matches, the respective transformation should be voted for multiple times. Regarding figure 3.49a, the cluster voting for the correct transformation appears only during the 4th submap match attempt and the ratio between its associated probability over the second best exceeds 0.5, which is the trigger threshold. It can be observed how all the probabilities associated to the other clusters suddenly drop as the correct one appears. The same behavior emerges also from the IN_OUT_RUN, where the correct transformation is voted from the 3rd submap match attempt and the ratio metric suddenly increases to over 0.8 triggering the relocalization. It can be observed that during the 2nd match attempt the ratio was higher than 0.5 corresponding to a wrong cluster. This happens while too little clusters are voted for and for this reason we wait to evaluate at least 3 match attempt before selecting a winner.

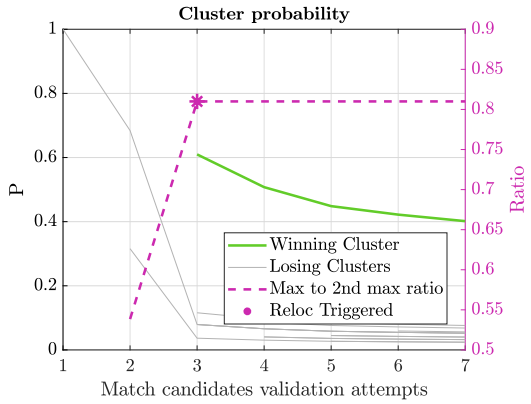
3.5.8 Relocalization on a Planetary Analogous Environment

In this section, the full relocalization pipeline is tested on two datasets captured on Mount Etna, designated as a planetary analogous environment. The two datasets are denominated *Etna_easy* and *Etna_hard*. In the first, the LRU rover drives autonomously across remotely designated waypoints. The rover travels around some rocky areas in two mapping sessions, separated by a brief pause. In this dataset, the environment is observed from roughly similar viewpoints across the two sessions and contains significant 3D information. It is therefore easier to candidate and validate submap matches. In both Etna datasets, the keypoint selection is based on high curvature regions (section 3.2.4). Figure 3.52 shows the complete map build after relocalization. Figure 3.52b shows how the probability associated to Tf clusters spikes when the 4th submap pair is pushed to the validator. Many correctly matching keypoints are here grouped in Hough3D groups which collect a large number of votes. The associated ratio metric is in fact around 0.9, much higher than the threshold 0.5 and close to the perfect score 1.

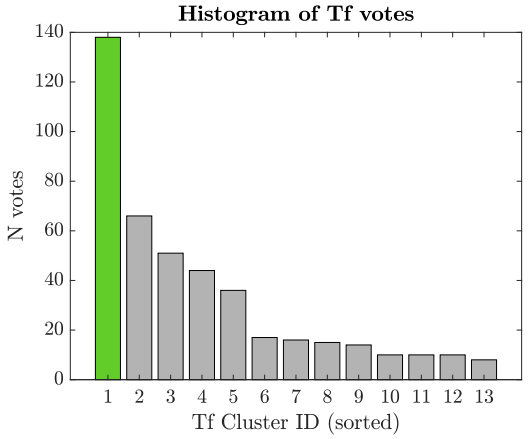
In the *Etna_hard* session, both the LRU and LRU2 rovers explore autonomously the environment without any remote intervention. The 3D map that they produce overlap for some submaps, however little 3D features are present. Meaningful information is contained



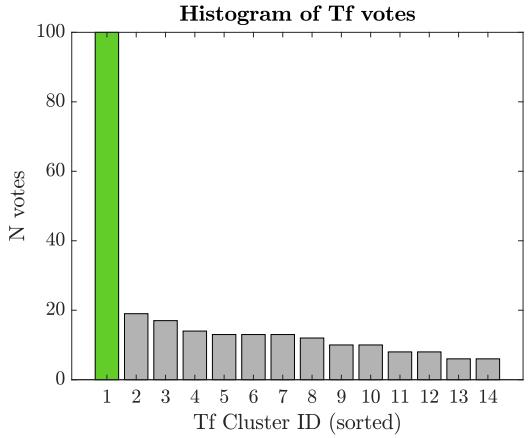
(a) Cluster probabilities



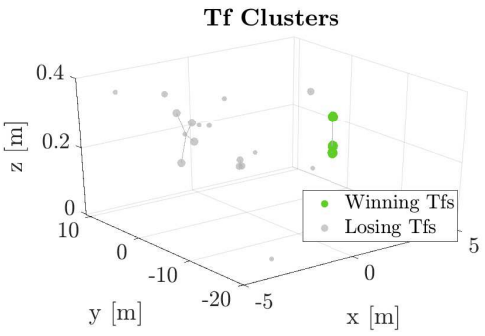
(b) Cluster probabilities



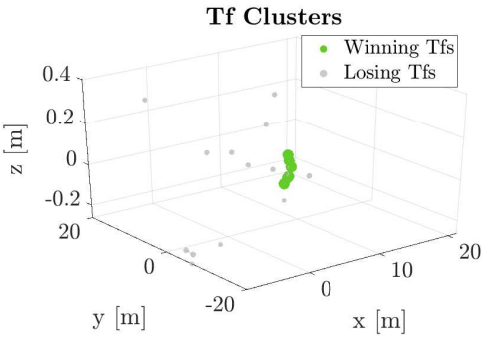
(c) Votes Histogram



(d) Votes Histogram

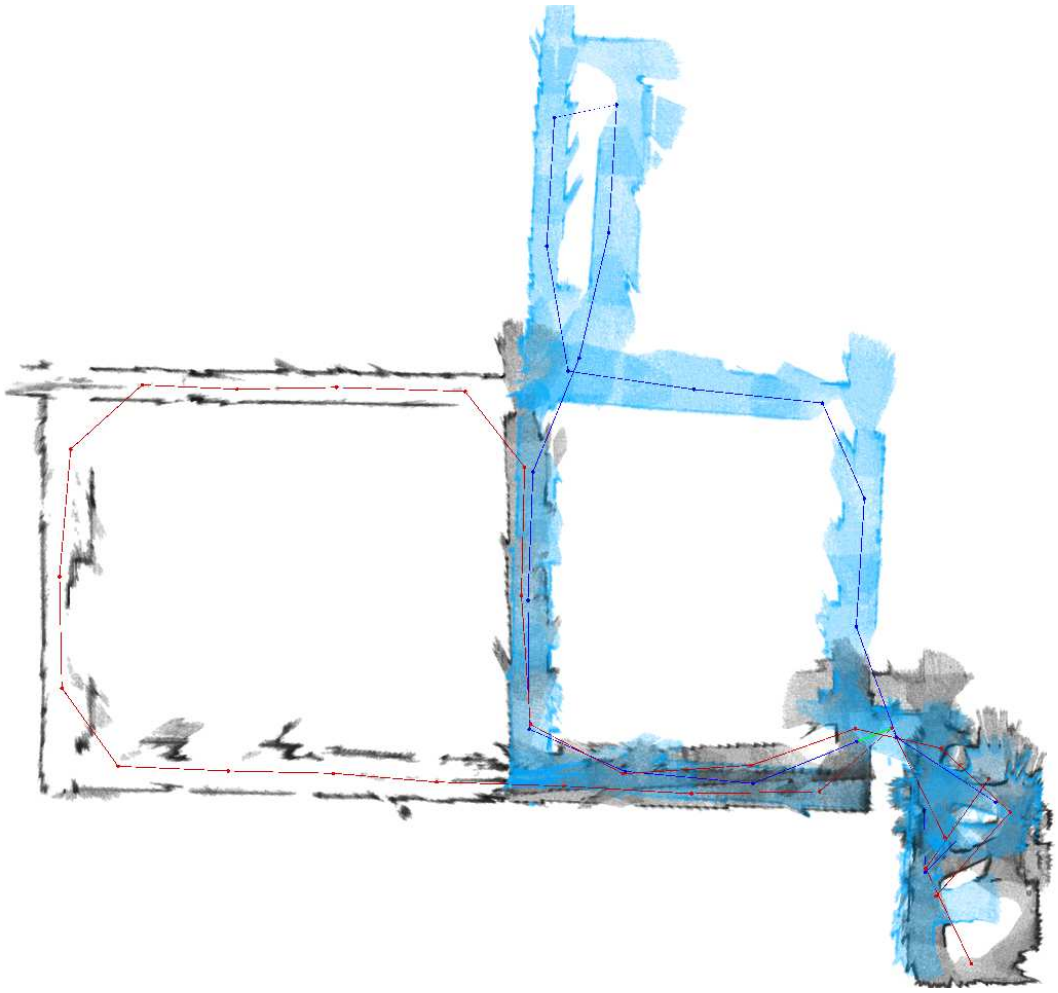


(e) 3D Tf Clusters Representation

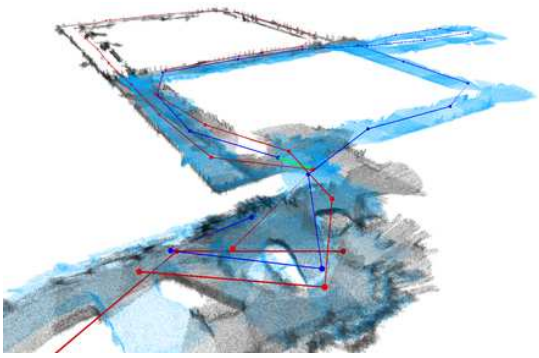


(f) 3D Tf Clusters Representation

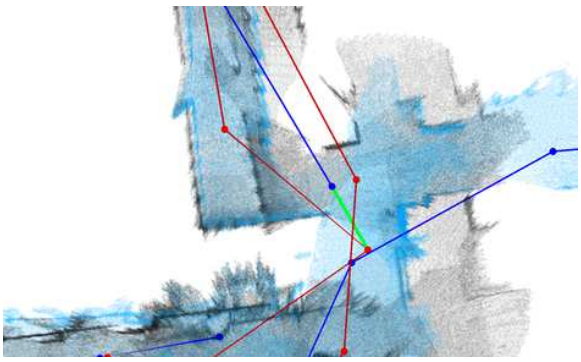
Figure 3.49: Results of the Tf clustering step after successfull relocalization for the IN_CORRIDOR_RUN (left column) and IN_OUT_RUN (right column). (a-b) evolution of the probability associated to each Tf cluster proportional to the number of votes for subsequent relocalization attempts, or pushed candidate matches. (c-d) ordered histograms of votes associated to each Tf cluster. (e-f) 3D representation of the Tf clusters considering only the $\{x, y, z\}$ coordinates and neglecting yaw.



(a) Full Map (Top View)

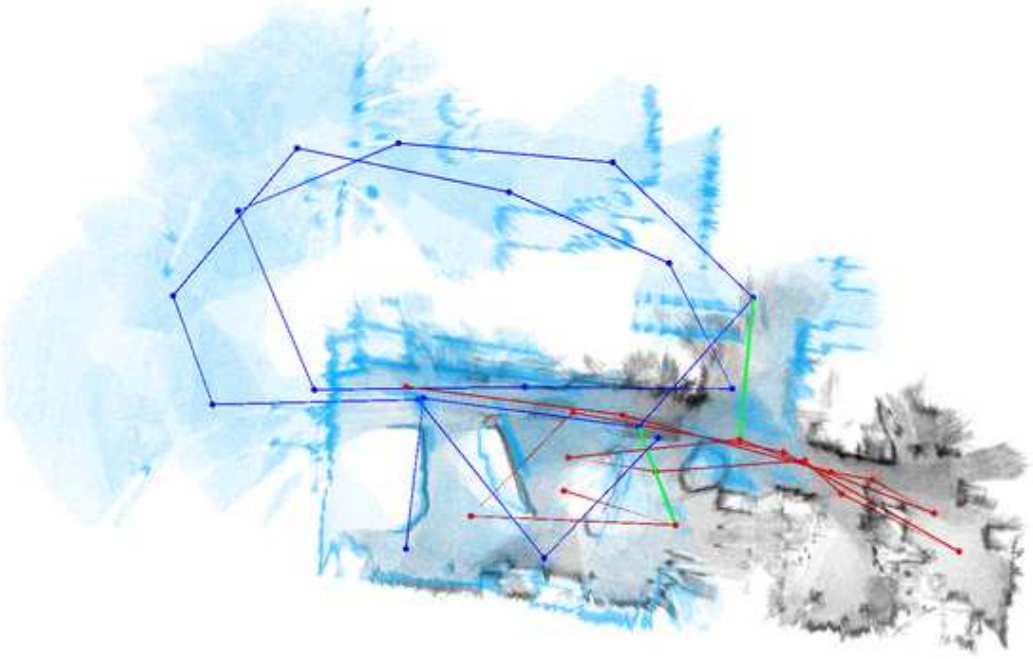


(b) 3D View

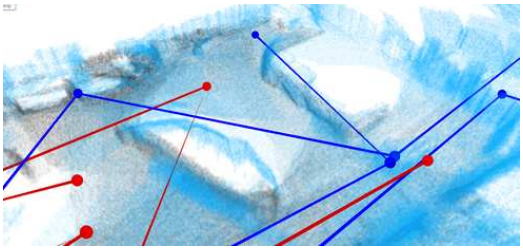


(c) Detail

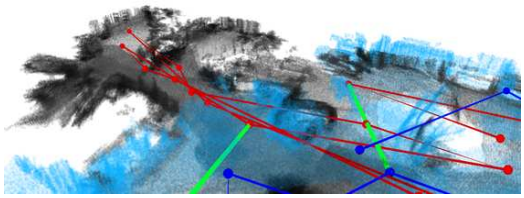
Figure 3.50: Views of aligned maps for IN_CORRIDOR_RUN after successfull relocalization. Maps and submap trajectories are related to LRU2 while the grey map and red submap trajectory are related to LRU. The visible green line connects the origins of submaps validated by the pipeline. The transformation between the connected local reference frames and propagated to the respective session origins defines the global transformation that alignes the LRU2 map to the LRU one.



(a) Full Map (Top View)

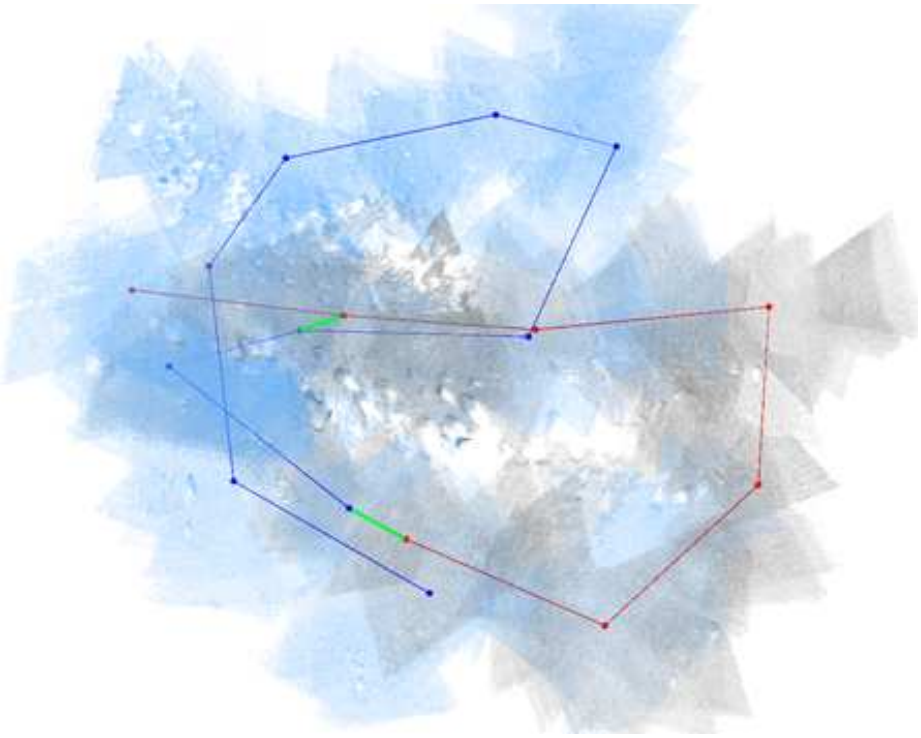


(b) Detail 1

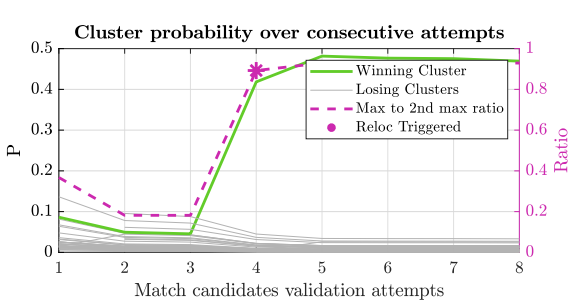


(c) Detail 2

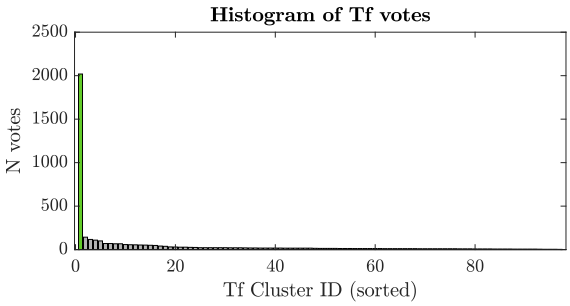
Figure 3.51: Views of aligned maps for IN_OUT_RUN after successfull relocalization. Maps and submap trajectories are related to LRU2 while the grey map and red submap trajectory are related to LRU. The visible green line connects the origins of submaps validated by the pipeline. The transformation between the connected local reference frames and propagated to the respective session origins defines the global transformation that alignes the LRU2 map to the LRU one. Detail views show how mapped environment features from the individual SLAM sessions are aligned after relocalization.



(a) Full Map *Etna Hard*

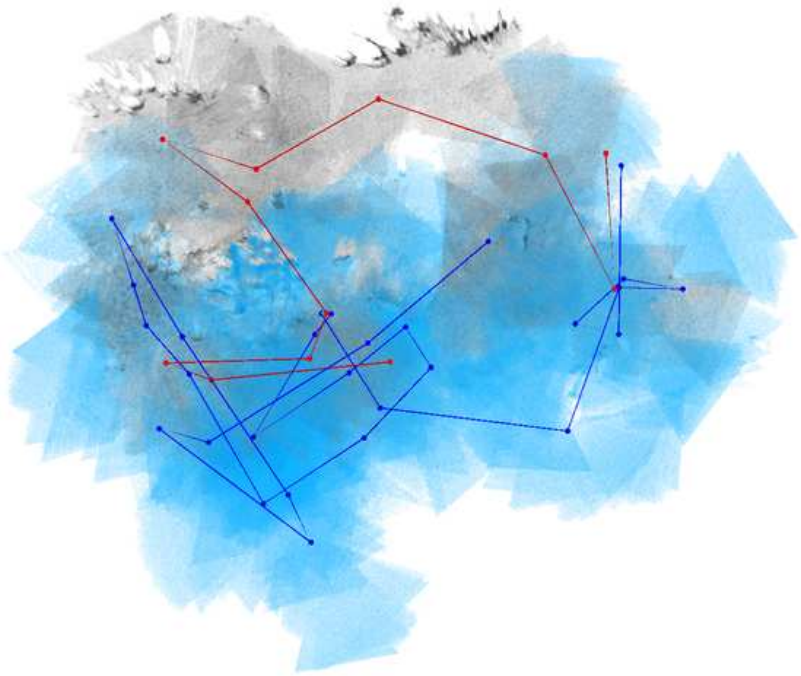


(b) Cluster Probability

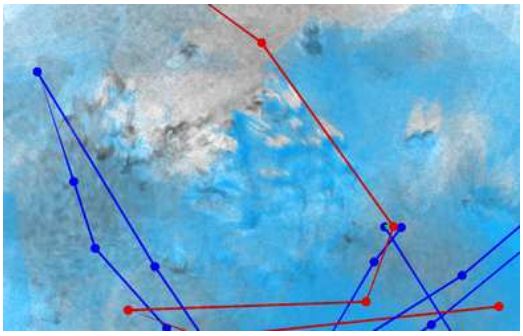


(c) Histogram

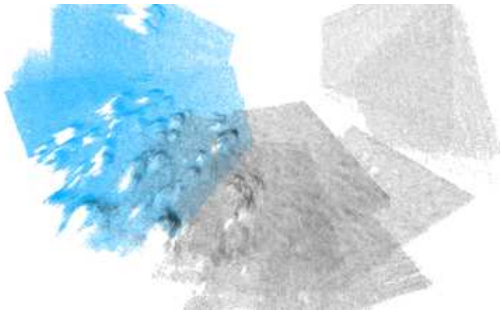
Figure 3.52: Full map and details for the *Etna easy* sequence. (b)-(c) are probability values associated to the Tf clusters after Hough3D groups. Relocalization is triggered after pushing the 4th candidate submap pair



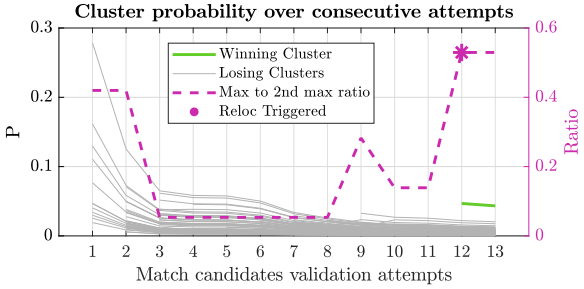
(a) Full Map *Etna Hard*



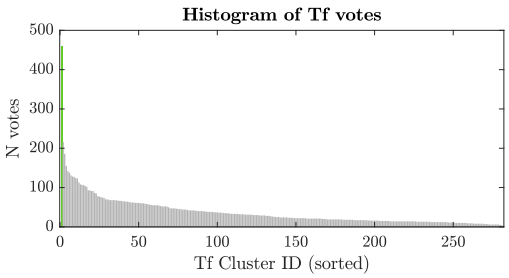
(b) Detail 1



(c) Matching submaps



(d) Cluster Probability



(e) Histogram

Figure 3.53: Full map and details for the *Etna hard* sequence. Details show the portion of the map where the only matching submap pair is located. Relocalization is triggered only near the end of the sequence

only in a few submaps concentrated on the center-left region of the map. In this case only one submap match is validated and is pushed just before the end of the session. Figure 3.53 shows the full merged maps as well as some details in the regions where relocalization occurs. Figure 3.53c in particular shows only the two matching submaps aligned from the winning transformation. The relocalization quality can be appreciated by observing how the rock features overlap with good visual accuracy.

Chapter 4

RGB-D Monocular Visual Odometry With a Low Resolution Time of Flight Camera

This chapter presents a monocular Visual Odometry algorithm where the scale ambiguity problem is solved by integrating depth measurements from low resolution range sensors. The algorithm is intended as a feasibility test for a following real-time implementation (chapter 5) and to compare the approach with stereo Visual SLAM and multiple range sensor configurations. This work extends our paper "Monocular Visual Odometry Aided by a Low Resolution Time of Flight Camera" presented at the 2017 IEEE International Workshop on Metrology for Aerospace [24].

4.1 Related Works

In this section it is given an overview of algorithms and methods for embedding scale information in monocular Visual SLAM or Visual Odometry systems. This evaluation, which relates not only to this chapter but also to chapter 5, features a brief discussion of peculiarities and limitations of the reported works focusing on why they might not be optimal for integration in small sized and resource constrained vehicles. As using only monocular observations the scale of the reconstruction is not constrained, additional sensors must be involved to give metric references. Amongst all solutions reported in the literature to solve this problem, three categories can be identified regarding the type of additional sensors used. The first group comprise all those methods which embed metric information from *depth sensors* such as RGB-D cameras or LiDARs. The second group gather methods who use cheap *altitude sensors* which can be found onboard many commercial UAVs and the third group bases scale recovery on *deep learning*.

4.1.1 Depth Enhanced Methods

This category of methods involves all those algorithms who actively use range information from RGB-D cameras, LiDAR sensors or Time of Flight cameras. RGB-D cameras comprise

both an RGB sensor for the visible spectrum and a depth sensor which can either use the time-of-flight or structured infrared light principle. RGB-D cameras are most suited for indoor usage such as augmented reality [21] and 3D reconstruction [151]. Many Visual SLAM system have however been published exploiting the depth measurements of RGB-D cameras, starting from the well known ORB-SLAM2 [98] which in addition to monocular and stereo also support RGB-D cameras. Amongst the SLAM system dedicated to only RGB-D can be cited RGBD-SLAM [36], KineticFusion [100] and more recently RGBDTAM [28]. The authors of [13] use an RGB-D camera exploiting only depth information to classify the perceived environment structure into planar regions, distinguishing obstacles and walls in alternative to a more expensive LiDAR sensor. In [67] a light RGB-D camera is mounted on an UAV to perform visual SLAM and plan paths in the non occupied places in the environment. Although the usage of an RGB-D camera might be attractive for the low cost of this type of sensors, their metrological performances degrade quickly in sunlight therefore outdoor usage is limited. In addition, while many small and lightweight RGB-D cameras can be found on the marker nowadays, their capability of measuring accurate ranges is limited to 1 or 2 meters. More performing sensor, as the Microsoft Kinect v2, require a dedicated power source and have larger dimensions and weight.

LiDARs (2D or 3D) are particularly valid methods for autonomous driving applications since they provide range information with high accuracy but at a much higher price than RGB-D sensors. V-LOAM [155] is a method for combining visual and LiDAR odometry [154] where motion estimates obtained by visual odometry are refined at a lower frequency ($1Hz$) by matching LiDAR scans. The benefit of this approach is a consequence of the complementary nature of visual and LiDAR odometry: the former allows to estimate rapid motions at a high frame rate while the latter provides accurate and relatively dense range information in order to eliminate drift and scale inconsistencies. However, the measurement setup used to evaluate V-LOAM comprises a 2D LiDAR in a sweeping configuration in order to obtain 3D information. Each LiDAR *sweep* must be accurately undistorted using a motion hypotesis. Additionally, the bulk of the sweeping mechanism and LiDAR limits the implementation on small sized vehicles. LIMO [53] is a recently published visual odometry pipeline which embeds ranges from a 3D LiDAR sensor in a more tight approach with respect to V-LOAM. While in the latter, visual estimations are used to refine already precise poses from LiDAR odometry, LIMO uses range information to establish scale constraints between camera poses and triangulated landmarks. This approach, published later than our original work [24] is similar to what we propose in this chapter. However, LIMO relies on using full 3D LiDARs, which provide a complete covering of the surrounding, and establish scale constraints by extracting planes from neighboring LiDAR points and observing the point-to-plane distance with associated visual landmarks. This approach is therefore not generalizable to 2D LiDARs which only return planar scans or lower resolution and noisier range sensor types.

Time of Flight cameras are active depth sensor which measure the range of the observed environment by computing the time of flight of modulated near infra-red (NIR) light produced by arrays of LEDs and reflected by objects. Compared to LiDAR sensors they return dense per-pixel range information at a high rate and have usually compact sizes. In addi-

tion, the intensity image of the return signals can be used for visual estimations such that the sensors resemble an RGB-D camera, however to do so it is required to have a relatively high sensor resolution. The authors of [91] investigate the usage of a SwissRanger SR-3k time of flight camera for 3D mapping of an indoor scene. The authors compare multiple approaches for evaluating the camera motion, firstly by aligning consecutive point clouds through an ICP algorithm, then by exploiting optical flow on range images and finally from SIFT feature matching. They show the superiority of aligning via ICP consecutive clouds although more expensive from the computational perspective. A time of flight camera of a smaller resolution of 64x48 pixels is used in conjunction with a spherical camera in [108]. The sensor setup is rotated to acquire "depth panoramas" from low resolution range images in order to build an obstacle map for obstacle avoidance and 3D map building.

4.1.2 Altimeter Based Methods

Accurate knowledge of a metric scale is necessary for autonomous control of UAV where monocular vision is employed to perform ego motion estimation. In [150] the visual scale is an explicit component of the vehicle state refined in an EKF framework together with gyro and acceleration biases. In [45] and [83] the metric scale is retrieved comparing the length of the travels in the z direction estimated by the monocular visual odometry and by an ultrasound altimeter. In [37] measurements from a barometric or ultrasonic altimeter are used to consistently estimate the metric scale in a statistical formulation. The authors assume that each measure is affected by Gaussian noise and the scale is then obtained using a Maximum Likelihood estimator. None of these approaches integrates directly scale estimates in the visual pipeline preferring to embded monocular visual odometry in a "black box" and relying instead on information fusion through EKFs.
A different approach is followed in [3] and [4] where range measures from a radar altimeter are used inside the visual pipeline to scale the triangulation of 3D landmarks. However, contrarily to our approach, range information is neglected in the optimization *backend*.

4.1.3 Learning Based Methods

Recent advances in deep learning techniques using convolutional neural network (CNN) show that depth can be learned in a supervised manner [117] or unsupervised [146]. In the first work, ground-truth depthmaps are used for training so that depth estimation is performed to an absolute scale. However, the necessity of supervision renders this family of approaches unsuitable for autonomous exploration. The authors of the second work propose to learn depth in an unsupervised manner including a direct visual odometry algorithm as a pose predictor for an image stream. Dense depth maps computed from the VO are used to build a cost function to be minimized during training. Aside from the computational cost, the authors show that failure to predict depth can occur in dynamic scenes or regions with low texture information. Deep learning is also applied to end to end learning of Visual Odometry. The authors of [148] train a network where couple of subsequent frames are fed to a CNN whose output is the input to a Recurrent Neural Network. The network is trained on the KITTI [48] dataset for autonomous driving and gains scale awareness from

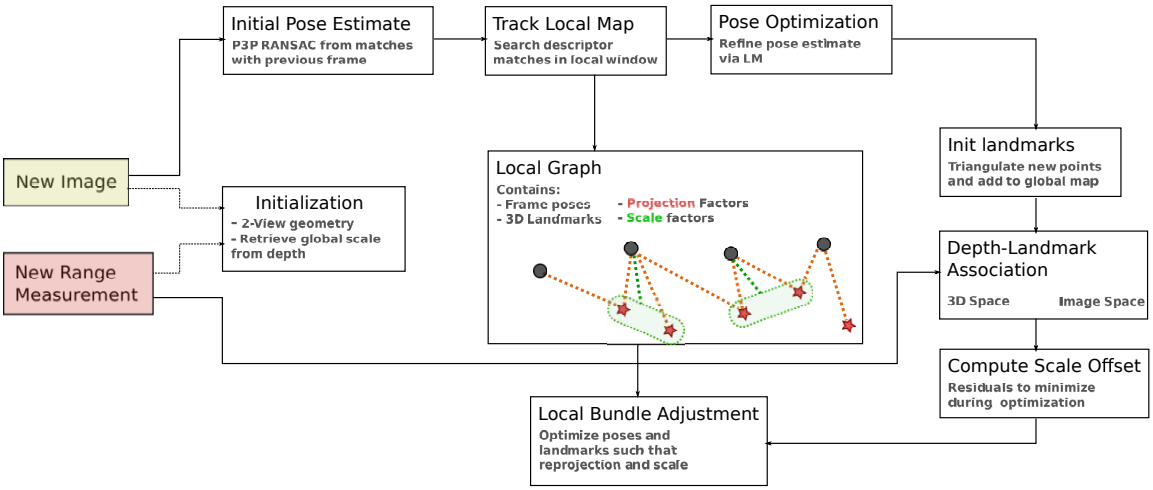


Figure 4.1: Functional Scheme of the RGB-D Visual Odometry pipeline described in this chapter. For each new incoming frame from the camera, the visual *front-end* searches the map of landmarks for 3D-2D correspondences and computes the instantaneous pose of the camera. Each ToF depth measurement is used to refine the global scale by either direct association to the 3D map or to selected images (see following section 4.4 and 4.4.2). Note that this algorithm does not process images in *real-time* (up to the camera frame rate) as it is meant as a feasibility test for the one developed in the next chapter.

the training poses, which are not generalizable to different environments. Depth prediction is used in [136] simulating the usage of an RGB-D camera where depth is learned and not measured. The authors show the superiority of this method over pure dense monocular approaches and even RGB-D SLAM systems in outdoor environment, when the sunlight degrades depth estimates. A scale-aware direct sparse odometry system is presented also in [152] where a deep neural network predicts disparity images from a virtual stereo setup. Training is performed in an unsupervised manner by employing a stereo SLAM system [147].

4.2 Algorithm Overview

This section presents a general overview of the developed scale aware monocular Visual Odometry algorithm. This pipeline comprises a visual *front end* whose aim is to track the camera position with respect to a map of 3D landmarks. Both the camera poses as well as the map are optimized by a *back end* utilizing the depth information provided by generic range sensors. Figure 4.1 depicts a scheme that comprises all the functional blocks of the pipeline. For each new incoming image, 2D SURF features are extracted and matched with the previous images in a local window. Given a set of correlations between initialized 3D landmarks (for which the 3D coordinates are fully determined) and 2D image features, the camera pose is computed solving a PnP problem. Matched features which correspond to non initialized frames are used to triangulate new 3D landmarks and add them to the map so that they can provide new information to localize the camera for the following frames. All the camera poses in a fixed-size local window as well as the observed 3D landmarks are optimized in the *back-end*, where depth measurements from the time of flight camera

are used to enforce *scale* constraints. Two different approaches to associate ToF depth measurements to the monocular systems are evaluated: the first correlates directly the 3D ToF measurements to visual information by projecting the ToF point cloud in the respective frame, the second approach correlates ToF measurements to landmarks searching for neighbors in the 3D space. Scale information is used both in the optimization *back end* as well as during *initialization* of the monocular pipeline. We developed this pipeline in order to use low resolution depth sensors such as an ESPROS epc610 Time of Flight camera with resolution 8x8 pixels and field of view 8.7x8.7 degrees. The low resolution combined with a narrow field of view makes it impossible to directly apply point cloud registration algorithms such as ICP [113] (Iterative Closest Point) or NDT [10] (Normal Distribution Transform). The clouds are in fact not informative enough to be exploited on their own and, in case of a low-power depth sensor such as the aforementioned one, too noisy to capture fine geometric details. Our approach however enforces depth constraints between the camera measurement, or the 3D point cloud generated by triangulating landmarks, and the depth sensor measurements resulting in an optimal fusion between the two. Furthermore, as the algorithms handles 3D measurements without assumptions on the sensor type, it is generalizable to any kind of depth sensor. This algorithm have no real time requirements and is intended as a feasibility study for the development of a high performance and computationally light pipeline which is the subject of the following chapter of this thesis. The focus of this chapter is to explore the possibility of using very low resolution range sensors for visual systems with constrained resources both in terms of mass as well as power requirements. This type of sensors not only is small, light and requires little power but also its limited amount of 3D data for the system to process reduces the computational effort significantly with respect to LiDAR SLAM systems.

4.3 Front End

With the term *front end* is referred here the part of the developed visual odometry algorithm in charge of computing the camera pose for each new frame received. The camera is calibrated using the Zhang method [157] in order to determine its intrinsic parameters such as the camera matrix K and the distortion parameters d. Each frame is firstly corrected for lens distortion, then SURF (Speeded Up Robust Features) features [8] are extracted. The

Table 4.1: Brief comparison of depth sensors using the LiDAR/ToF measurement principle

Sensor	**Type**	**Range** [m]	**FOV** [deg]	**Acc** [mm]	**Power** [W]	**Weight** [g]	**Price** [g]
Espros epc610	3D ToF	3	8.7x8.7	±4cm	2*	10	1k$
Espros epc611	3D ToF	3	8.7x8.7	±4cm	2*	5	500$
Mesa SR4000	3D ToF	8	43x34	±1cm	10	500	10k$
Kinect V2	3D RGB-D	4.5	70x60		20	1400	200$
Asus XTion	3D RGB-D	3.5	58x45		5	600	300$
Sick Tim572	2D LiDAR	25	270	±3cm	5	250	2k$
Sick PLS 312	2D LiDAR	50	180	±5cm	20	5000	5k$
Velodyne Puck	3D LiDAR	100	360x30	±3cm	8	830	10k$
HDL-64E	3D LiDAR	120	360x27	±1cm	60	12700	100k$

* Including evaluation kit for development

choice of selecting this descriptor is driven by the fact that no real time requirements are imposed for this algorithm, therefore it can benefit from the accuracy and repeatability of the SURF descriptor regardless of its relatively high computational time ($\sim 500ms/frame$). The SURF *detector* is particularly suited for natural environments as it behaves as a detector of blobs and the *descriptor* is invariant to both scale, rotations and illumination. The *front end* of our algorithm considers a window of 5 images, $\mathbf{I} = \{I_i\},\ i = n : n-5$. Features $x_j = \{u, v\}_j$ extracted from the current image I_i are matched across the local window using L2 (Euclidean) distances of their corresponding descriptors to find projections of the same 3D landmarks X_j. Such correspondences between features can be used to determine the 3D landmark coordinates using the DLT (Direct Linear Transformation) [62] algorithm.

4.3.1 Initialization

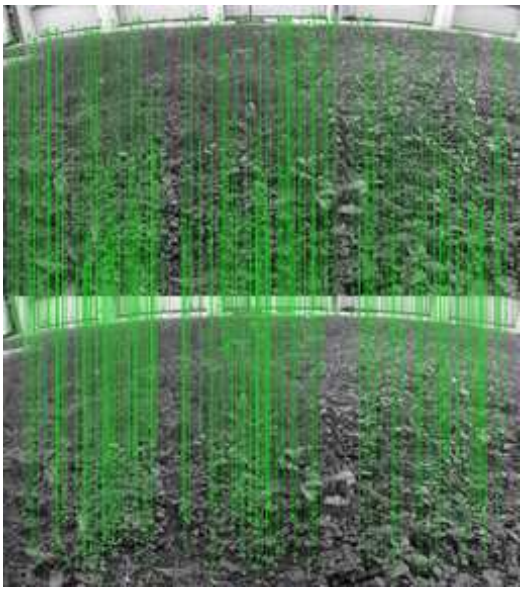
Differently from stereo or RGB-D visual pipelines, where a full 3D representation of the environment is available from the first frame, monocular visual pipelines need to initialize a tentative *structure* for the perceived environment in order to localize with respect to it for all the next frames. The same approach is followed here as in the real time visual odometry discussed in the next chapter, which is estimating a geometry of camera poses and landmark coordinates from two views. Given the first two images I_i and I_{i+1} a set of local SURF features $F_i = \{f_i,\ i = 1 : n\}$ and $F_j = \{f_j,\ j = 1 : m\}$ are extracted and matched minimizing L2 distances:

$$f_i \text{ and } f_j \text{ match} \leftrightarrow f_j = \arg\min_j ||f_i - f_j||_{L2} \quad \text{and} \quad ||f_i - f_j||_{L2} < t \tag{4.1}$$

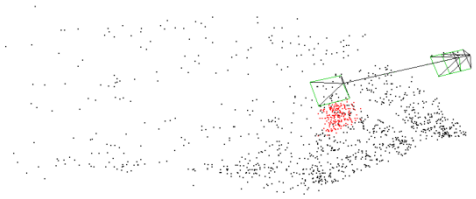
where t is a threshold for the descriptor distance such that the correspondences between the closest descriptors where the distance is too high are rejected. Knowing the camera matrix K after intrinsic calibration, the obtained correspondences can be used as input to a 5 point algorithm [101] of a normalized Direct Linear Transformation [62] to determine the *essential matrix* or a *homography matrix* for the two views respectively if the scene is non-planar or planar dominant. Further details about the meaning of these two matrices and a brief mathematical formulation are referred to the introductory chapter in section 2.3.2 and 2.3.1. Knowledge of the 6 DoF transformation matrix between the two views $\tilde{\mathbf{T}}_{P_0}^{P_1} = [R|\tilde{t}]_{P_0}^{P_1}$ allows to triangulate the 3D points related to the matched features and build an *unscaled* map. This is due to the fact that both the aforementioned algorithms determine translations up to a scale factor. As the last step during initialization, a Global Bundle Adjustment is performed to optimize both the 3D point coordinates as well as the camera poses by minimizing the *reprojection error* of all landmarks:

$$\varepsilon_{i,j} = |\mathbf{x}_{i,j} - \pi(\mathbf{X}_j, \mathbf{T}_{P1})| \tag{4.2}$$

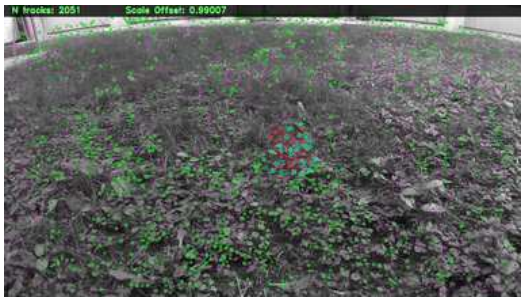
where $\mathbf{T}_{P1}$ is the second camera pose in the two view geometry and $\mathbf{T}_{P0}$ is implicitly the identity $[I|\mathbf{0}]$. $\mathbf{X}_j$ are the 3D coordinates of the triangulated landmarks and π denotes the projection function from a pinhole camera model (section 2.1). The next step for initializing the monocular visual odometry is to compute a correct scale of the reconstruction, for which



(a) Matches between two views

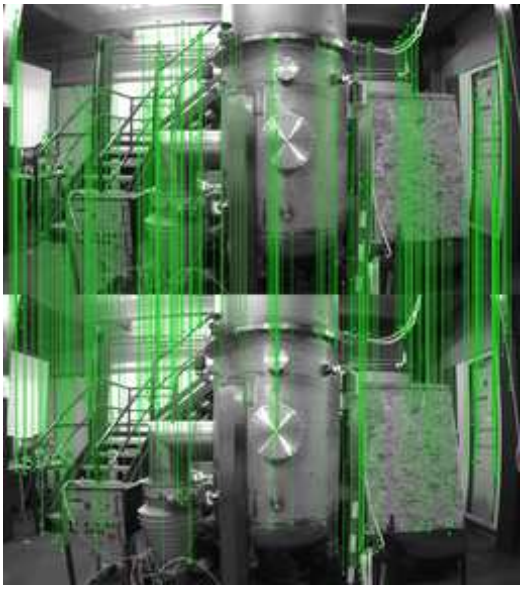


(b) 3D view

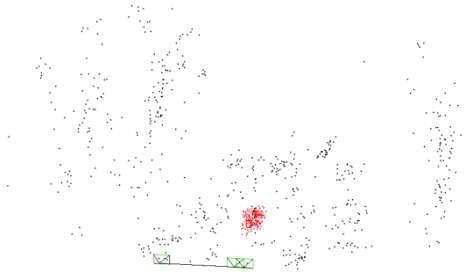


(c) Algorithm view

Figure 4.2: Example of initialization procedure. (a) Two consequent images are used to extract SURF features. After matching them and computing an homography for this scene is planar dominant, the full 3D geometry of the environment is reconstructed as well as the camera poses, visible in (b). The algorithm proceeds then to track the initialized features (c) and compute camera poses. (b) show reports both the *scaled* and *unscaled* second camera pose. Black dots are 3D landmarks, red points are time-of-flight camera ranges



(a) Matches between two views



(b) 3D view

(c) Algorithm view

Figure 4.3: Example of initialization procedure for a non-planar dominant scene

depth measurements from the range sensor are associated as described in section 4.4.

4.4 Association of depth measurements

In this section it is described how depth measurements from the employed range sensor are associated to visual information coming from the RGB camera. In order to estimate a correct scale of the environment, depth constraints are enforced from the available ranges. Contrarily to other approaches in the literature where LiDAR scans are used in the first place to perform odometry, our approach is camera-centric: the perceived depth is used to correct the scale of triangulated landmarks as the visual odometry system builds a map during exploration. Monocular setups in fact do not perceive scale and their pose estimates can be parametrized as:

$$\mathbf{T} = \left[\begin{array}{c|c} sR & t \\ \hline 0 & 1 \end{array}\right] \quad s \in \mathbb{R} \tag{4.3}$$

(although in this work transformations are not parametrized with the explicit scale factor s) where R is a rotation matrix, t is a translation vector and s is a scale factor. Equation 4.3 highlights that errors accumulated by visual estimations do not propagate only on the camera orientation and position, but also on the scale factor. If the scale s is not optimized at run time, it will eventually drift, especially during sharp turns where 3D landmarks that define local scales are observed for little time [98]. Associating 3D range measurements to a map of landmarks is not a trivial task as this map, deriving from monocular visual estimations, can be very sparse and noisy. We investigate two different approaches for this task. The first of them involves computing scale offsets based on the Euclidean distance of landmarks and range points in the 3D space. The second approach relies instead entirely on visual information, tracking range projections in the following frame using optical flow.

4.4.1 Depth association on the image space

Association of depth measurements with visual information on the image space is performed by firstly projecting the time of flight point cloud in the RGB image. Let the time of flight point cloud be denoted by $\mathcal{P}_i^{\text{ToF}} = \{\mathbf{X}_l^{\text{ToF}}\}$ for $l = 1 : n_{\text{ToF}}$ with $\mathbf{X}_l^{\text{ToF}}$ being the individual 3D points. By knowledge from extrinsic calibration (section 4.6.1) of the rigid transformation between the time of flight and RGB cameras $\mathbf{T}_{\text{ToF}}^{\text{cam}}$, it is possible to project $\mathcal{P}_i^{\text{ToF}}$ in the image frame, finding the feature locations in pixel coordinates:

$$\mathbf{x}_{i,l} = \pi(\mathbf{T}_{\text{ToF}}^{\text{cam}}, \mathbf{X}_l^{\text{ToF}}) \tag{4.4}$$

where $\pi()$ denotes the projection function. Projections of ToF points $\mathbf{x}_{i,l}$ are tracked from frame I_i to frame I_{i+1} by means of a Pyramidal Lukas-Kanade Tracker using a local window of size 31x31 pixels. The result of this search is a set of pairs of 2D correspondences for the time-of-flight camera measurements, which are used to triangulate the 3D points $\tilde{\mathbf{X}}_{ToF,l}$ with respect to the current scale (figure 4.4a depicts this process). Differences between the true global scale and the current scale of the monocular reconstruction produce a different depth of each triangulated point with respect to the measured $\mathbf{X}_{ToF,l}$. A scale factor is

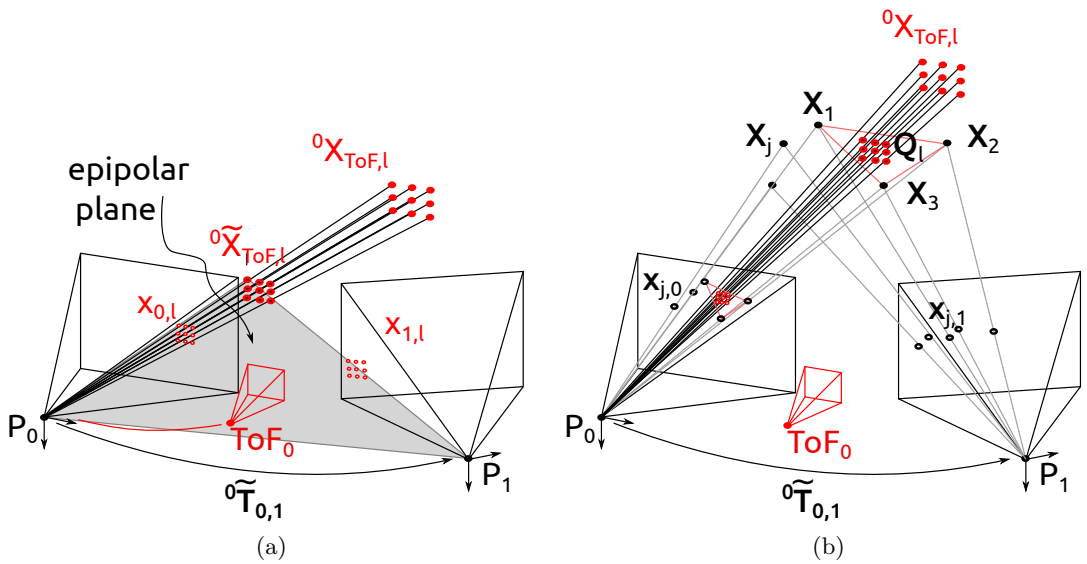


Figure 4.4: (a) Camera Features Association to Depth Measurement on Images Space. The geometric consistency between ToF 3D points projections on P_0 $\mathbf{x}_{0,l}$ and the projections on P_1 $\mathbf{x}_{1,l}$ are verified by researching them along the epipolar lines. Unscaled ToF points $\tilde{\mathbf{X}}_{ToF,l}$ are compared with the ToF depth measurement $\mathbf{X}_{ToF,l}$ to retrieve the map absolute scale r. (b) Camera Features Association to Depth Measurement on 3D map. Three neighbor 2D points are selected so that the plane defined by the respective 3D points should contain the ToF measure.

computed finally as:

$$s_{corr} = \frac{1}{n_{ToF,i}} \sum_{l} \frac{\|\mathbf{X}_{ToF,l}\|}{\|\tilde{\mathbf{X}}_{ToF,l}\|} \tag{4.5}$$

where l spans across all the ToF projection tracks in I_{i+1}. During initialization, the scale factor s_{corr} is multiplied to the second camera pose as well as the 3D coordinates of landmarks while during each Bundle Adjustment, the individual scale offsets for each l are added as factors to minimize during optimization. Figures 4.3b and 4.2b report a 3D view of the initialization stage, showing both the scaled and un-scaled second camera poses.

4.4.2 Depth association on the 3D space

The process of associating ToF measurement across frames using optical flow of ranges projections is strictly dependent to the texture quality of the image patches. If the search area of the image I_{i+1} contains very repetitive or too uniform texture patterns, the search can converge to wrong 2D coordinates ending in wrong triangulation of ${}^{Pi}\tilde{\mathbf{X}}_{ToF,l}$, therefore computing wrong scale factors. For this reason another approach is tested which should favor robustness of scale factor estimations rather than accuracy. In this approach, ToF measurements are associated directly to triangulated landmarks from the visual *front end* in order to evaluate the difference in their depths. It is recalled here the two-view geometry reported in figure 4.4b, where a ToF cloud is captured at the same time of I_i in position P_i. The set of 3D points $\mathcal{P}^{ToF}$ captured by the Time of Flight camera is projected in the image frame of the respective RGB frame (Eq. 4.4). For each one of the projected points, three

neighbour SURF features are selected so that the plane that is defined by the respective 3D points should contain the ToF measure. This approximation is better the more dense and uniformily distributed is the set of matched SURF features in the images. The nearest neighbor search consists in finding $\mathbf{x}_j,\ j = 1:3$ such that the L2 distance from the ToF point projection is less than a threshold (generally 20 pixels in our tests):

$$\{\mathbf{x}_j, \mathbf{x}_{i,l}\} \text{ associated } \iff ||\mathbf{x}_j - \mathbf{x}_{i,l}||_{L2} < thresh \tag{4.6}$$

where $\mathbf{x}_{i,l}$ is the l-th ToF projection in I_i. Let be $\mathcal{X}_l = \{\mathbf{X}_{1,l}, \mathbf{X}_{2,l}, \mathbf{X}_{3,l}\}$ the set of points describing the plane where $\mathbf{X}_{ToF,l}$ should lie if the scale factor was correct. Let also be $\mathbf{Q}_l$ the intersection between the measurement semi-line and the plane defined by $\mathcal{X}_l$. An estimate of the scale correction factor is:

$$s_l = \frac{\|\mathbf{X}_{ToF,l}\|}{\|\mathbf{Q}_l\|} \tag{4.7}$$

A factor s_l is computed for each point in the ToF measured array. The final estimate of the scale factor s is obtained averaging each s_l. As for the previous association mode, at initialization time the full set of triangulated 3D points $\mathcal{X}$ and the vector $\mathbf{t}_{P_{i+1},P_i}$ are multiplied by s. At Bundle Adjustment time the scale factors are added to the graph representing the range cost functions to minimize.

4.4.3 Pose estimation

As the monocular pipeline is initialized, a map of landmark with determined 3D coordinates is available to localize the camera for the following frames. Given a set of 2D landmark projections in pixel units and the corresponding 3D spatial coordinates, the camera pose is estimated in a 3D-to-2D approach using the EPnP algorithm [82] embedded in RANSAC scheme for robust outlier rejection. The robustness deriving from RANSAC comes with the price of computational effort, approximately linear in the number of iterations. However as mentioned before, the aim of this algorithm is to demonstrate the robustness of our scale recovery approach and real-time constraints will be enforced only for the algorithm described in the following chapter. For each new frame I_i, firstly feature correspondences are searched in the previous image I_{i-1} by matching SURF descriptors, and a first pose estimate for P_i is determined from EPnP. Using this first guess for the camera location, triangulated points belonging to a frame window (of 5 frames in our tests) are projected on image I_i and new matches are searched between their descriptors in the past frames and the current one. These new feature associations are used to refine the EPnP estimate by a non-linear optimization stage using Levenberg-Marquardt algorithm. As the camera moves and initialized landmarks (for which the 3D location is known) exit from the field of view, feature matches detected previously which belongs to non initialized landmarks are used to triangulate them therefore augmenting the map for localization in the next poses.

4.5 Back End

As the camera moves and the map is augmented, errors in pose estimation and landmark triangulation cause each pose P_i to accumulate *drift* both in rotation, translation and scale. At fixed time intervals, it is performed a windowed Bundle Adjustment where a fixed window of frames (15 in our tests) is optimized for the lowest reprojection errors and scale offsets. The goal of this step is to find the camera poses and landmark locations which are most consistent with the camera observations (detected features) and range measurements. Due to the non linear nature of the Bundle Adjustment, a correct solution to the problem can be expected only if the initial solution is close to the optimum, for which the attention towards robustness to outliers in pose estimation is beneficial. The term Local Bundle Adjustment refers here to the fact that only a fixed window of frames is optimized instead of the global history, and is meant to be performed in real time or near real time. The problem involves minimizing the following cost:

$$C(\mathbf{x}, \mathcal{X}, \mathcal{T}, \mathcal{P}^{ToF}) = F(\mathbf{x}, \mathcal{X}, \mathcal{T}) + G(\mathbf{x}, \mathcal{X}, \mathcal{T}, \mathcal{P}^{ToF}) + S(\mathcal{T}) \tag{4.8}$$

F denotes a contribution to the cost computed over the landmark reprojection errors over the camera frames where they are observed:

$$F(\mathbf{x}, \mathcal{X}, \mathcal{T}) = \sum_{i=n_{pos}-m}^{n_{pos}} \sum_{j=1}^{n_{obs}} f_{i,j}(\mathbf{x}_{i,j}, \mathbf{X}_j, \mathbf{T}_i) \tag{4.9}$$

where $\mathcal{X}$ denotes the set of observed landmarks $\mathbf{X}_j$ and $\mathcal{T}$ denotes the set of camera poses $\mathbf{T}_i$ contained in the fixed window. $f_{i,j}$ is the reprojection error of the j-th landmark in the i-th frame:

$$f_{i,j} = |\mathbf{x}_{i,j} - \pi(\mathbf{X}_j, \mathbf{T}_i)| \tag{4.10}$$

G denotes instead a cost contribution computed from the scale offset between the triangulated landmarks and the respective time-of-flight camera measurements. If the measured ranges are associated to the map in the 3D space (as explained in section 4.4.2), then:

$$G(\mathbf{x}, \mathcal{X}, \mathcal{T}, \mathcal{P}^{ToF}) = \sum_{i=n_{pos}-m}^{n_{pos}} dist(\mathcal{P}_i^{ToF}, \mathcal{X}_i) \tag{4.11}$$

where $dist()$ computes the distance between landmarks and ToF points as in eq. 4.7. However, if the algorithm is associating ToF measures to the image space, the quantity which defines a cost term for the optimization is defined as in section 4.4.1.

J is a regularization term for the scale during optimization of the local window. Poorly constrained landmarks in fact can lead to divergence of the scale factor as the algorithm is purely monocular. A prior scale $\tilde{s}$ is computed before starting the optimization process as the $L2$ distance between the origins of the first two poses in the local window. Being $k = n_{\text{pos}} - m$ with n_{pos} frame index and m usually 15, the scale prior is:

$$\tilde{s} = ||\mathbf{t}_{k+1}^W - \mathbf{t}_k^W||_{L2} \tag{4.12}$$

(a) Sensor Setup (b) Morpheus Rover

Figure 4.5: Experimental setup used for the evaluation of the proposed Visual Odometry pipeline. (a) Sensor setup comprising a StereoLabs ZED camera, an Espros epc610 Time-of-Flight camera and a SICK PLS312 LiDAR. (b) The Morpheus Rover [25] carrying the sensor setup angled towards the ground to allow the range sensors to detect reasonably close objects. A Swift Piksi Multi differential GPS is mounted in the back of the rover to obtain a ground truth with centimenter-level accuracy

and the regularization term is consequently:

$$J(\mathcal{T}) = \tilde{s} - ||\mathbf{t}_{k+1}^{W} - \mathbf{t}_{k}^{W}||_{L2} \tag{4.13}$$

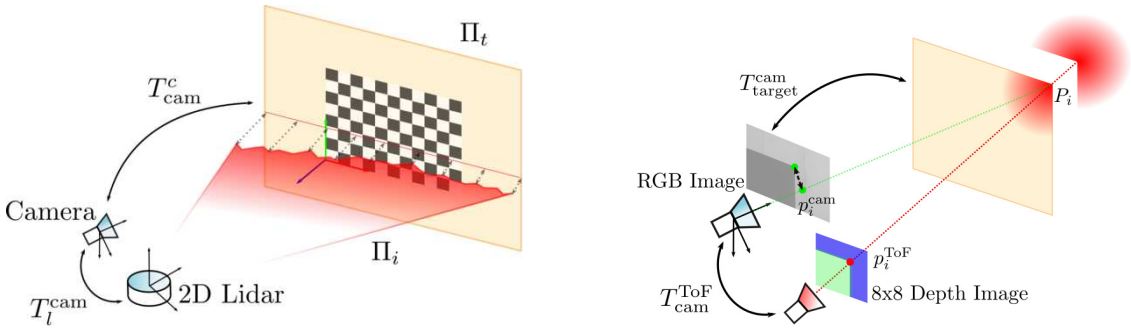
where the second term changes during optimization.

4.6 Experimental Setup

To test the performances of the proposed range-aided Visual Odometry algorithm, an experimental platform was set up comprising a stereo camera, a low resolution Time-of-Flight camera and a 2D lidar. While the Visual Odometry is purely monocular and runs using only the left camera frames, full stereo frames are recorded for the sake of performance comparison with scale-aware state of the art stereo visual SLAM algorithms. Figure 4.5a shows the assembled sensor setup, with the Time-of-Flight camera on top (only frontal printed circuit board and lens), the stereo camera and the plane scan LiDAR on the bottom. Figure 4.5b shows the "Morpheus" mobile robotic platform carrying the sensors in the front slightly angled towards the ground such that both the LiDAR and ToF camera could return range measures in the 1-2 meters range. The rover was controlled remotely for the experiment and sensor data as well as D-GPS relative poses were recorded on a laptop in *bagfile* form for later evaluation.

4.6.1 Camera and range sensors extrinsic calibration

Most of the solutions present in literature to calibrate cameras and range sensors setups are related to the usage of 3D or 2D LiDARs. The authors of [78] use a target composed



(a) Calibration LiDAR-camera (b) Calibration ToF-camera

Figure 4.6: 4.6a shows the extrinsic setup involving the color camera and the 2D LiDAR: Π_t is a calibration target, which is a plane with a checkerboard pattern allowing knowledge of T^c_{cam}. The lidar scan plane is Π_l and intersects Π_t in a line. 4.6b shows the extrinsic setup between the RGB and ToF cameras: the sensor setup is moved with respect to a square target in the 3 translation axis knowing with high accuracy the relative pose. Ranges from the low resolution ToF camera allows to detect corners, which are correlated with Harris corners in the RGB image minimizing a reprojection error.

of two inclined planes and associate manually extracted lines in the images with corners in the lidar scans. The transformation between the camera and LiDAR reference systems is estimated by aligning the found correspondences for a set of camera and range measures. In [99] it is proposed a closed form solution for the camera-LiDAR calibration given a simple target. In [47] it is described a calibration technique for 3D lidars and cameras based on the alignment between planes coinciding with multiple checkerboards with planes detected in the range scans. [2] propose a calibration framework for cameras and 2d lidars in sweeping configuration and [52] propose a calibration method for 3D lidars and cameras involving trihedral targets. A few works exist also on the calibration between time-of-flight and RGB cameras. The authors of [59] present a calibration method tested with a Mesa Imaging SR4000 time of flight camera (176x144 pixel resolution), which in addition to depth provides also monochrome intensity images. This allows to match checkerboard corners between images and estimate the extrinsic parameters by projecting the measured 3D points to the RGB frames. As depth noise can easily arise from differences in surface reflectivity and spurious light, the checkerboard is segmented from the intensity images and a plane is fitted on the respective depths. Depths of the matched features in the intensity images of the RGB and ToF camera are extracted from the fitted plane. Image-based alignment is performed also in the work of [70], where a 2D planar target with circular markers placed in an irregular pattern is observed by both an RGB and time of flight camera. 2D correspondences and depth measurements provide a first relative pose estimation, which is refined by estimating depth biases. Amplitude images of the reflected signal from the time of flight camera are used to extract and match features with other imaging sensors in the Astrobee robot [27].

ToF-Camera calibration

The time of flight camera that we used for this work has a much smaller resolution of 8x8 pixels and does not measure intensities in the visible spectrum, therefore methods based on image alignment are not applicable. The very minimal resolution also prevents to use the amplitude images to distinguish features from different reflectivity of printed targets. The calibration method that we developed for this challenging setup [105] is based on the detection of meaningful points from the time of flight images and minimization of the reprojection error in the image frame. We employed a very simple calibration target comprising a flat square plate such that both the RGB and ToF cameras could precisely detect one of its corners (see figure 4.6b for a graphical representation of the calibration setup). The calibration target was moved with respect to the camera setup both in the lateral and longitudinal direction, on a distance in the z camera direction ranging from 0.5 to 1.5 meters. The lateral movement was instead constrained by the narrow field of view of the time of flight camera, as the requirement was to observe the same plate corner. Let be P_i the 3D coordinates of the plate corner detected by the time of flight camera and z_i^{cam} the detection of P_i in the RGB camera frame. The problem to solve is essentially:

$$\arg\min_{\mathbf{T}_{\text{cam}}^{\text{ToF}}} \sum_{i=0}^{n} ||z_i^{\text{cam}} - \pi(K \ \mathbf{T}_{\text{ToF}}^{\text{cam}} \ P_i^{ToF})|| \tag{4.14}$$

where n is the number of corner detections, π is the projection function on the RGB camera frame through its camera matrix K and $\mathbf{T}_{\text{ToF}}^{\text{cam}}$ is the transformation matrix between the RGB and ToF camera reference frames. Eq. 4.14 represents a simple 3D-to-2D camera resectioning problem. A dark color is applied to the target plate in an uniform way such that the image gradients with respect to a bright background are maximized. Detection of the plate corner (figure 4.7) is then performed by extracting Harris corners [61] on the whole image and selecting the one with the highest metric. Regarding the corner detection on the time of flight camera measures, we choose to exploit the range images of 8x8 pixels instead of 3D data which is noisy and suffer from artifacts in the proximity of edges. Each low resolution depth image is interpolated in the two direction using splines to achieve

(a) Cropped (b) Full Image

Figure 4.7: Example of plate corner detection on a camera frame. (a) cropped to the interest region, (b) full image undistorted. The green cross identifies the highest scoring Harris corner.

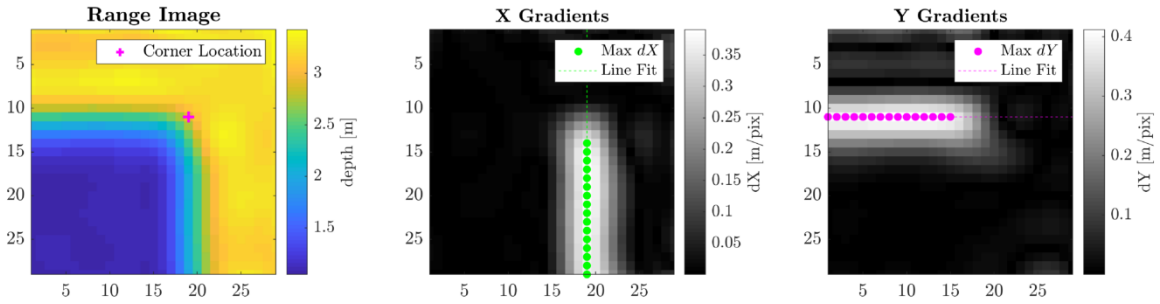
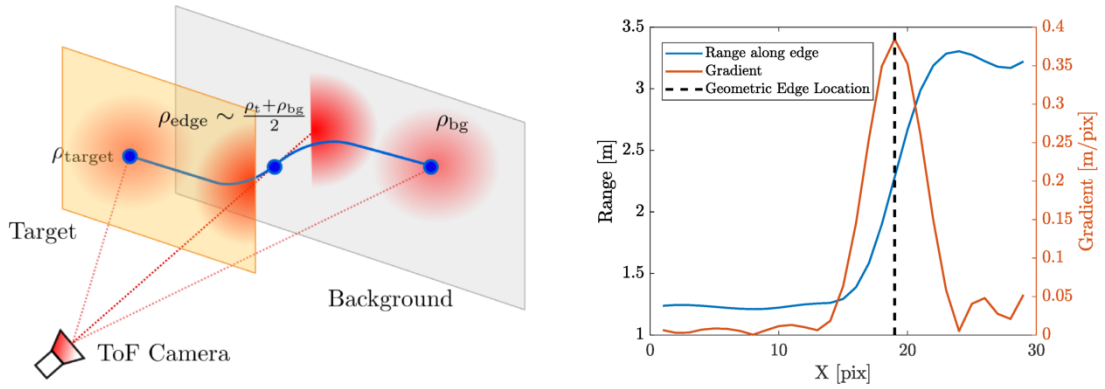


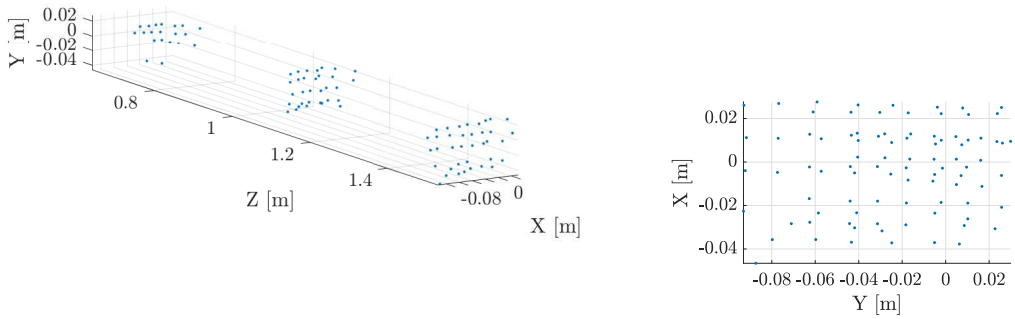
Figure 4.8: Detection of the plate corner on the ToF range images. The left image depicts ToF ranges interpolated using splines. The distinction between foreground and background is here evident. A magenta cross pinpoints the bi-dimensional location of the corner detection as the intersection between the edge lines computed from maximum gradients in the X and Y directions.



(a) Sample view of edge scenario

(b) Ranges and Gradient

Figure 4.9: Depiction of the *veiling* effect during range measurements coincident with separation zones from foreground and background. As each range measurement is performed observing a narrow field of view, on the edges both the foreground and background contribute to the return signal, leading to a depth value which is close to the mean of target and background. (b) shows the actual ranges and gradients observed along one of the rows of the range image shown in figure 4.8. From this considerations, the maximum gradient regions coincide with the geometric edge of the calibration target.



(a) (b)

Figure 4.10: 3D visualization of the plate corners detection from the time-of-flight camera ranges. Corner positions span an interval of around 1 meter in z and 10 centimeters in x and y

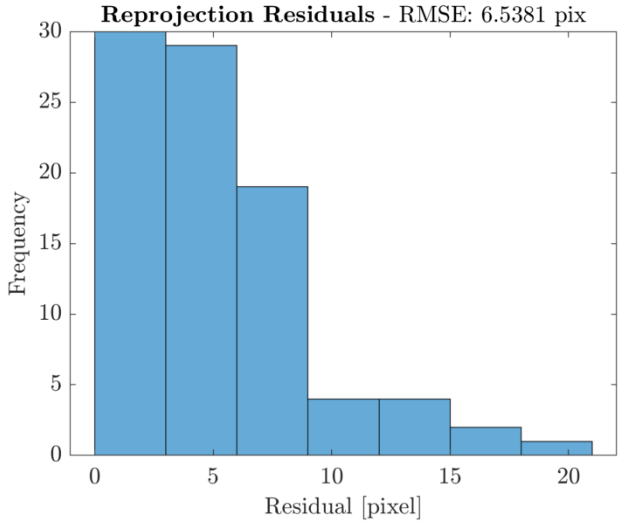


Figure 4.11: Histogram of the reprojection errors between the plate corner point and the respective Harris corners in the RGB frame. The RMSE (Root Mean Square Error) in this case is about 6.5 pixels, which is acceptable considering the high noise in the corner depth estimates

up-sampled frames of 32x32 pixels. A second step is to compute the gradients of the depth ρ along the x and y directions:

$$\nabla I_\rho = \left[\frac{\partial I_\rho}{\partial x}, \frac{\partial I_\rho}{\partial y} \right] \tag{4.15}$$

For each row of the range image I_ρ we search for the maximum gradient in the y direction and store the pixel location where it happens. Then we search and store the pixel values where the maximum gradients in x occur. The objective here is to extract the two edges belonging to the target plate such that their intersection leads to the position of the corner.

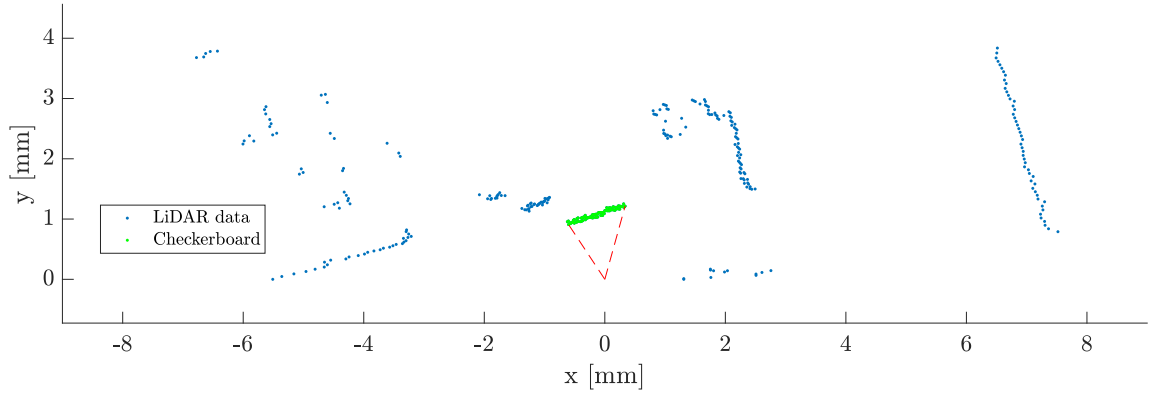
As the range images are captured such that the plate edges are mostly parallel to the x and y axes, we are confident that the highest gradients only appear along the edges directions. Furthermore, we can say that the maxima should be coincident with the geometric edge

location. As the ToF camera sweeps its field of view to generate range measurements, when the observed position coincides with an edge, the return signal will be partly affected by the foreground and partly by the background in similar proportions. The depth returned would be approximately the mean of foreground (target in this case) and background, and the spatial gradient would be maximum as it is shown in figure 4.9. The resulting set of points, which can be visualized in figure 4.10, identify the two plate edges and the intersection of the fitted lines denote the bi-dimensional corner point location. Taking the depth of this identified corner however would return a wrong value because of veiling effect as the field of view of the camera in that point comprise both the foreground and background. For this reason we choose to associate to the corner the mean depth of the foreground. Having stored a significant set of plate corner detections, a first estimate of the extrinsic configuration between the RGB and ToF cameras is obtained by solving a *Perspective-n-Points* problem, where the RGB camera is aligned with respect to the Time of Flight camera such that the detections of the corner with Harris features are consistent with the measured 3D positions. We use a P3P [74] algorithm embedded in a RANSAC scheme to discard outliers. Finally, the non-linear problem expressed in equation 4.14 is solved using Levenberg-Marquardt starting from the first given estimate. Figure 4.11 reports the reprojection residuals after the optimization step showing that the majority of projection factors were solved with a residual error close to or lower than 5 pixels. Some high residuals are still present from corner detections which were affected by high range errors.

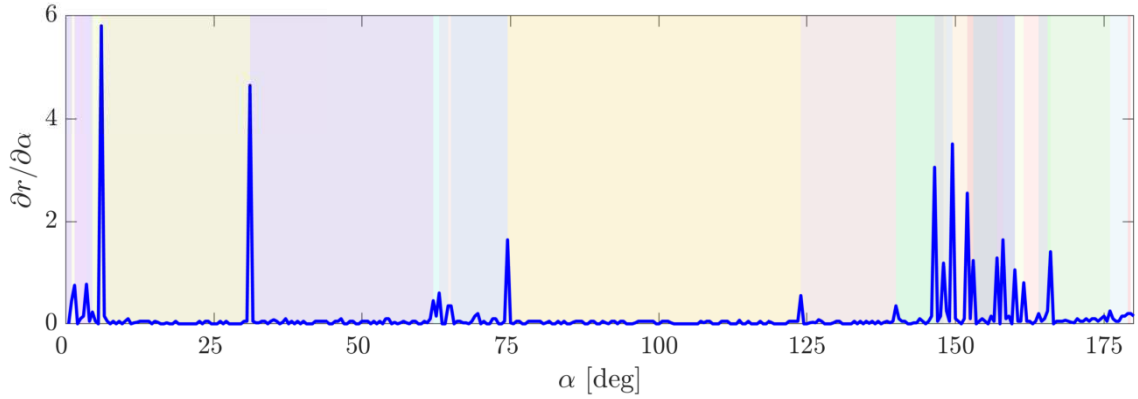
2D LiDAR-Camera calibration

Extrinsic calibration of the 2D LiDAR and stereo camera is performed implementing the method of Vasconcelos et al. [141]. Figure 4.6a shows a scheme of the sensor setup as well as the unknowns to estimate. The calibration target in this case is a planar surface Π_t whose position in the 3D space is recognized by the camera through the use of a known checkerboard pattern. The LiDAR measures ranges by scanning the plane Π_i, which intersects the target plane Π_t in a line L_i. The target of the calibration procedure is to find the transformation T_l^{cam} such that the measured ranges coincide with the line L_i.

The first step is to capture a sequence of synchronized LiDAR scans and images (from the left camera frame). As the calibration method [141] relies on fitting lines in the part of the scan representing the planar target, an automatic segmentation procedure is implemented to exclude all the environment points from the scans. Let be α the scan angle, which for the SICK PLS 312 LiDAR is comprised in the interval $[0; 180]$. For each couple of subsequent scan points, the angular derivative of the range measurements is used to determine if the two points lie on the same smooth surface or not. Therefore if the range variation $||(r(\alpha_{i+1}) - r(\alpha_i))/\Delta\alpha||$ is higher than a threshold, a boundary between two smooth surfaces is detected. The derivative is approximated to simply $||r(\alpha_{i+1}) - r(\alpha_i)||$ and the threshold is set to 0.5 meters. All the angle values for which this condition is met represent offsets between continuous surfaces, which are segmented from the full scan. The segment whose length is closer to the checkerboard size is selected as the target. To avoid any wrong decision, the procedure is supervised by checking the automatic selection of the target plane. Figure 4.12a shows an example of segmented 2D scan where the environment



(a) Segmented 2D Scan

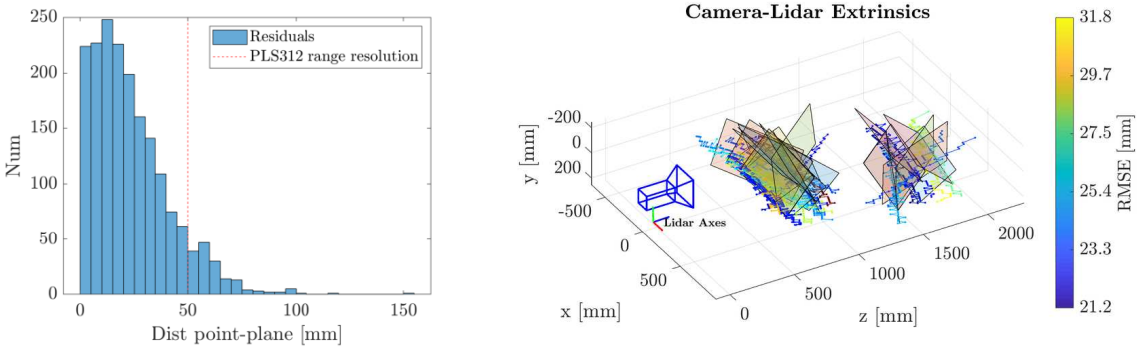


(b) Range Derivatives

Figure 4.12: (a) Segmentation of the target plane from the full scan cloud. The LiDAR sensor lies on the axes origin, dashed red lines denote the margins of the target points which are highlighted in green. (b) Derivatives of the range measurements over the scan angle. High values represent offsets between the surfaces observed from the environment. Between scan angles 150 and 250 lies the checkerboard of (a).

points are plotted in blue while the detected target plane is highlighed in green. The continuous surfaces from which the plane is selected are highlighted in figure 4.12b as colored patch. These are delimited by peaks in the range derivative curve. For this example, the patch between approximately 150 and 250 degrees denotes the target plane, as the length of the segment is close to the width of the target.

As enough pairs of images and segmented scans are acquired, the calibration method, for which the details can be found in the author's paper [141], is invoked to find a fitting transformation T_l^{cam} between the LiDAR sensor and the camera. In order to evaluate the outcomes of the calibration, we observe the distribution of the residuals, which are point-to-plane distances between the checkerboard scan points and the plane measured from the checkerboard pattern. Figure 4.13a shows an histogram of residuals after calibration. The distribution is denser close to the value of zero and decreases rapidly towards the 50 mm value, which is also the resolution of the depth measurements returned by the SICK PLS 312 LiDAR. Figure 4.13b shows instead all the target calibration planes, oriented in a way

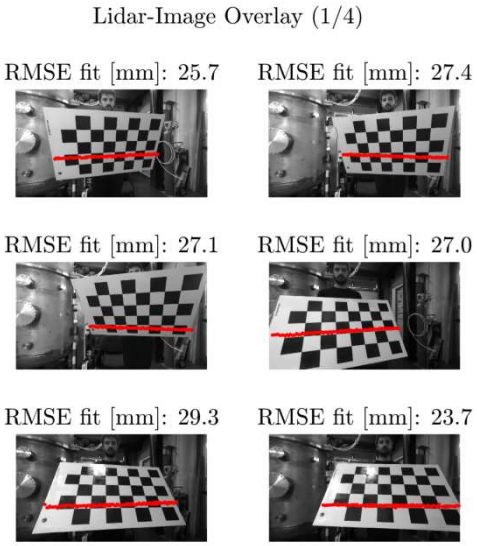


(a) Histogram of Residuals

(b) Extrinsics Visualization

Figure 4.13: Figure 4.13a displays a frequency histogram of the point-plane distances between scans and target planes after extrinsic calibration. A vertical red dashed line highlights the PLS lidar range resolution. Figure 4.13b is a 3D view of the camera-lidar configuration after calibration, reporting also the position of the target planes and the range measurements, which are colormapped according to their RMSE error.

that all the involved degrees of freedom of the alignment problem are well constrained. Segmented ranges for the calibration target are also displayed colormapped by their RMSE error, which results from the root mean square of the residuals themselves. The highest values are related to the most inclined planes, however always being acceptable in comparison with the range resolution of the sensor.



(a) Scan projection

Figure 4.14: Projection of LiDAR scans (segmented target only) to the checkerboard planes after calibration. Correct correspondences between the target and scan borders suggest a good outcome for the extrinsic calibration.

Name	Range sensor	Association method
2D-ToF	EPC610 ToF camera	Image space
3D-ToF	EPC610 ToF camera	3D space
2D-LiDAR	SICK PLS-312	Image space
3D-LiDAR	SICK PLS-312	3D space

Table 4.2: VO configuration tested

4.7 Results and Discussion

In this section the proposed algorithm is evaluated for its accuracy in reconstructing the camera trajectory. Because this method is intended as a feasibility test for the algorithm proposed in the next chapter, no particular attention is put into real-time performances. The choice of using SURF features in fact slows down the runtime to about 1 Hz in average, therefore no performance figures are presented here. We performed two different experiments, in the first one the sensor setup was moved manually using a linear slide in order to have millimeter-level accuracy in the ground truth for translations. As a second test scenario, the sensor setup was mounted on our mobile rover platform (visible in figure 4.5b) which was manually driven in a natural environment. Although our algorithm is supposed to enable small aerial vehicles to perform VO up to a correct scale, the sensors that we employed were impossible to integrate in an UAV for their size and weight.

We use a variety of error metrics to evaluate performances with a focus on translation drift which exposes scale inaccuracies. Let x_i^* denote ground truth poses at frame i and let x_i denote the ones estimated by the algorithm. The first error metric that is evaluate is the Absolute Trajectory Error, simply defined as the L2 norm of the difference between the true and estimated poses:

$$ATE_i = ||x_i - x_i^*|| \tag{4.16}$$

which tells how much the trajectory is deviating from the true path. However, there are multiple sources of absolute trajectory errors such as rotation errors, estimation biases [39], mis-calibrations and scale drift in case of monocular pipelines. The ATE is then complemented by a scale drift metric which is defined as the difference in length between consequent Visual Odometry steps. Let be $\delta_i = ||x_{i+1} - x_i||$ the trajectory length between two subsequent estimated poses and $\delta_i^* = ||x_{i+1}^* - x_i^*||$ respectively for the ground truth. The instantaneous scale is then defined as:

$$\text{Scale}_i = \frac{\delta_i}{\delta_i^*} \tag{4.17}$$

Values over 1 indicate that the algorithm is overestimating the step length therefore the metric scale. Values under 1 indicate the opposite. It is also evaluated to what extent the scale drifts over time resulting in a different trajectory length:

$$SD_i = \sum_{i=0}^{n} ||\delta_i - \delta_i^*|| \tag{4.18}$$

In addition to these error metrics it is also plotted the scale offset perceived by the algorithm

at runtime. This value is the mean of all point-point distances which are minimized by the optimization back-end in order to maintain a correct scale. To some extent these values shows the capabilities of the algorithm to adapt to scale changes and drifts along the trajectory, and confirm that a global scale is actually being optimized. When the algorithm is behaving correctly the scale offset value should stabilize to 1. To compare the accuracy of our algorithm to widely known and available Visual SLAM solutions, ORB-SLAM2[1] [98] is run on our datasets in stereo mode. Although ORB-SLAM2 is not modified to use range measurements in any way, by using a stereo camera landmark depths are initialized with a correct global scale.

4.7.1 Small scale motion evaluation

In the first test scenario, the sensor setup is mounted on a graduated linear slide (partially visible in figure 4.5a) and moved in the transversal direction in steps of 5 cm over a total length of 135 cm. This evaluation highlights the motion estimation accuracy on small scales and the precise ground truth allows to compare the algorithm behavior in its different configurations. The algorithm is in fact tested using both ways of associating ranges to visual landmarks, on the 3D and image space, and using the low resolution time of flight camera and the 2D LiDAR as well (see table 4.2 for the nomenclature used in the following paragraphs). Two sequences are captured in different environments: the first sequence, labeled *INDOOR*, takes place in an indoor laboratory environment. The sensors observe artificial landmarks at different depths and characterized mostly by reflective surfaces which can induce noise in range sensing especially for the time of flight camera. The second sequence, labeled *OUTDOOR* takes place in a grassy outdoor scenario. The sensors are slightly tilted towards the ground in order to measure distances in the $[1-2]$ meters range. In this case, the environment is very feature rich and visual tracking is facilitated.

INDOOR sequence

Figure 4.16 shows two example camera views of the test environment including the detected SURF features (in magenta), the one corresponding to initialized landmarks (in green) and the projection of range measurements in the image frame for both the time of flight camera (figure 4.16a) and the 2D LiDAR (figure 4.16b). The camera trajectory, depicted in green, starts at the axes origin and proceeds to the left. Figures 4.16c and 4.16d shows the point cloud of triangulated visual landmarks as well as the 3D points corresponding to range measurements using the time of flight camera. The range measurement noise can be easily visualized in the top view as the ToF clouds, positioned in the environment respectively to the camera pose estimate, appear as an noisy scatter of points. This environment appears to be particularly challenging in fact for the algorithm when using the low resolution time of flight camera. The pose errors along the trajectory is particularly high in fact by looking at figure 4.15a. However, by constraining the scale using the 2D approach, the drift is significantly lower across the whole path if compared to the 3D method. Using the 3D approach, in order to establish scale factors, initialized landmarks must be found in the

[1] https://github.com/raulmur/ORB_SLAM2

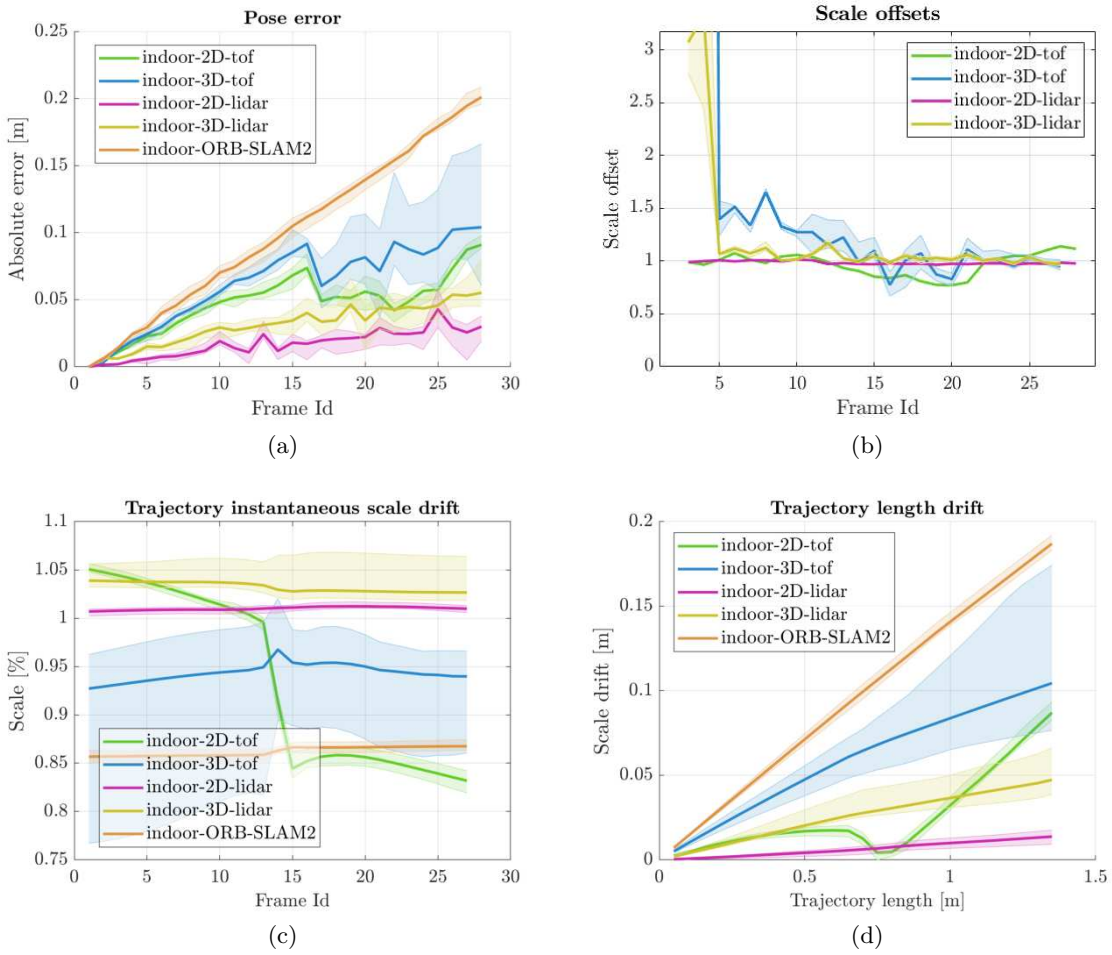
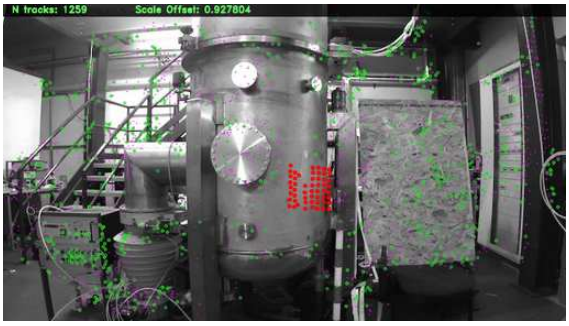
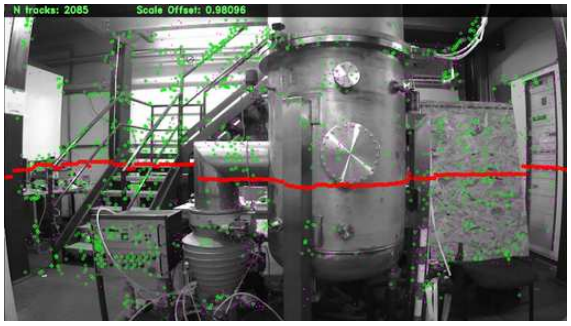


Figure 4.15: Evaluation results in the INDOOR sequence. The algorithm is tested in a variety of configuration, using both the LiDAR and ToF range measurements and switching between the 3D and 2D approaches. In addition, ORB-SLAM2 is tested in stereo mode using the full output from the ZED stereocamera. The "Scale offsets" plot does not contain information about ORB-SLAM2 as it considers the scale residuals using range sensors. These results are related to 20 independent runs, solid line is the median value, bands are first and third quartiles.

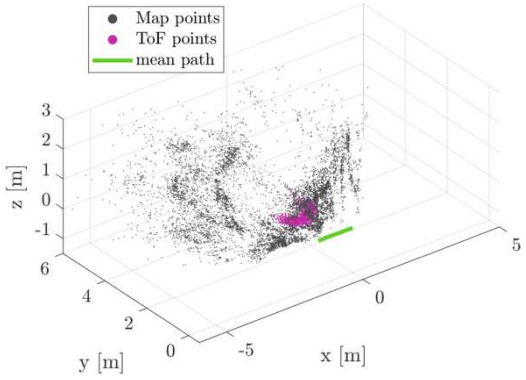
proximity of range measurements projections. For the majority of the views however, range projections lie on almost uniformly textured surfaces where little to none SURF features are detected and triangulated. In this scenario, using the 2D LiDAR as range sensor allows to obtain much more accurate poses and more stable scale estimates along the trajectory. As the projected LiDAR points are in higher number and cover a wider region in the image, it is more likely for the algorithm to establish scale constraints for both the 3D and 2D approaches. However, also in this case the 2D method is more accurate and allows to reconstruct almost a perfect scale factor by looking at figure 4.15c. Figure 4.15b shows how the scale offset perceived by the algorithm converges to the value of 1 as more frames are covered. This means that in all the 4 configurations, the VO algorithm converged to a consistent scale showing a well conditioned behavior. Notably ORB-SLAM2 performed worse than all configurations, initializing a wrong reconstruction scale at the beginning of



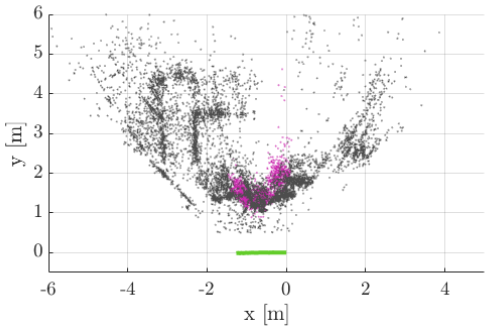
(a)



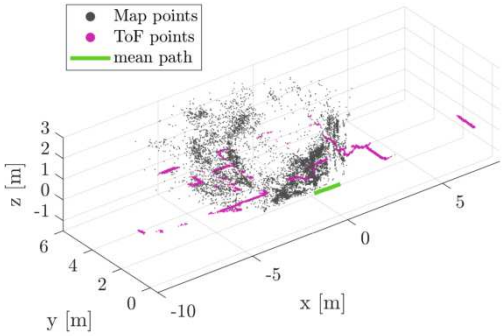
(b)



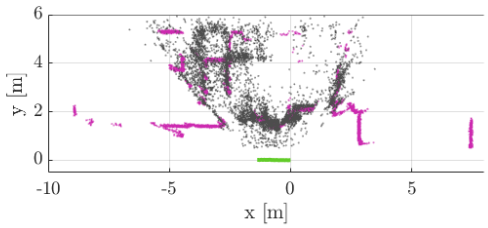
(c)



(d)



(e)



(f)

Figure 4.16: 3D reconstructions from the INDOOR sequence. (c) and (d) are 3D and top views of the reconstructed environment using the time-of-flight camera as the range sensors. Points in magenta are 3D locations of range measurements positioned in the environment from the estimated camera transformations. (e) and (f) are the reconstructions performed using the SICK PLS 312 LiDAR as range sensor. The accuracy in retrieving the correct environment scale is visibile in (d) from the good overlap of range scans between themselves and on the environment (black dots). (a) and (b) are example views of the dataset superimposing ToF and LiDAR ranges respectively. Magenta dots are SURF features, green circles denoted initialized features

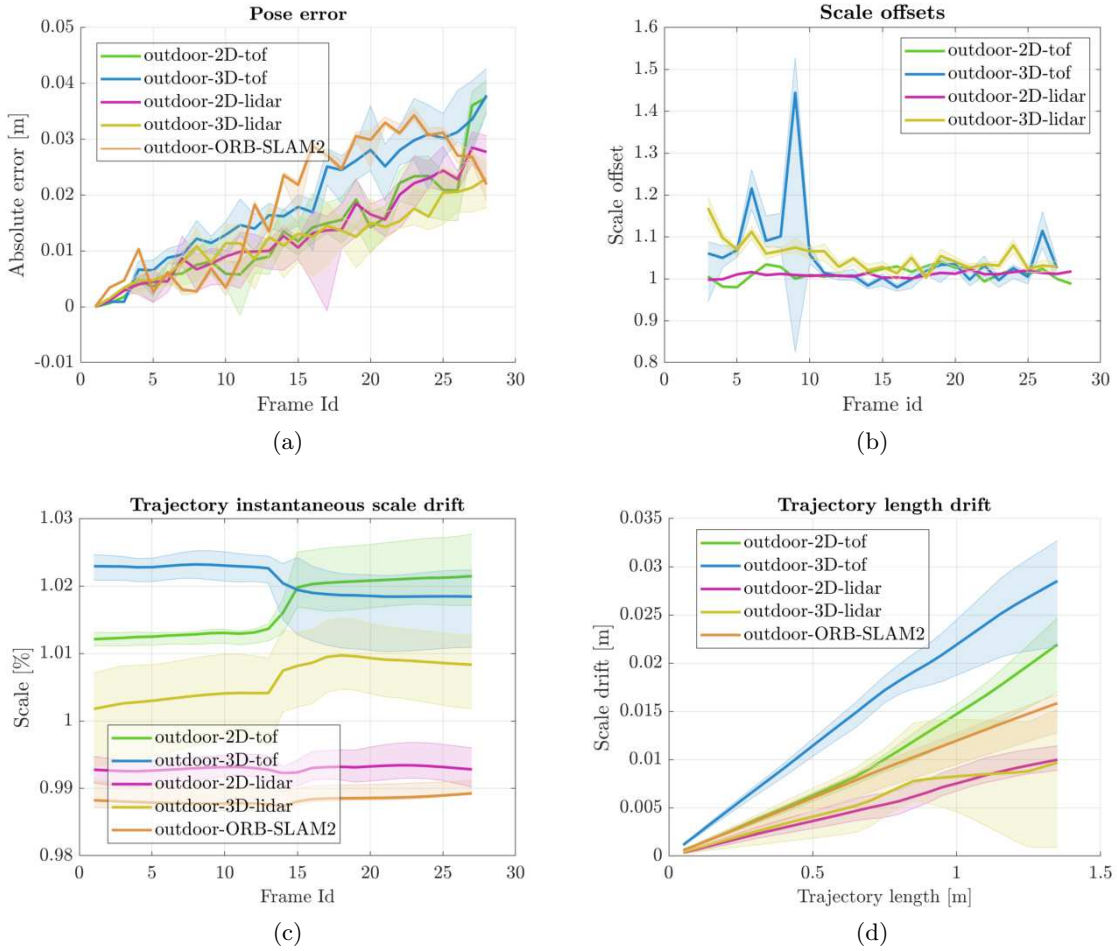
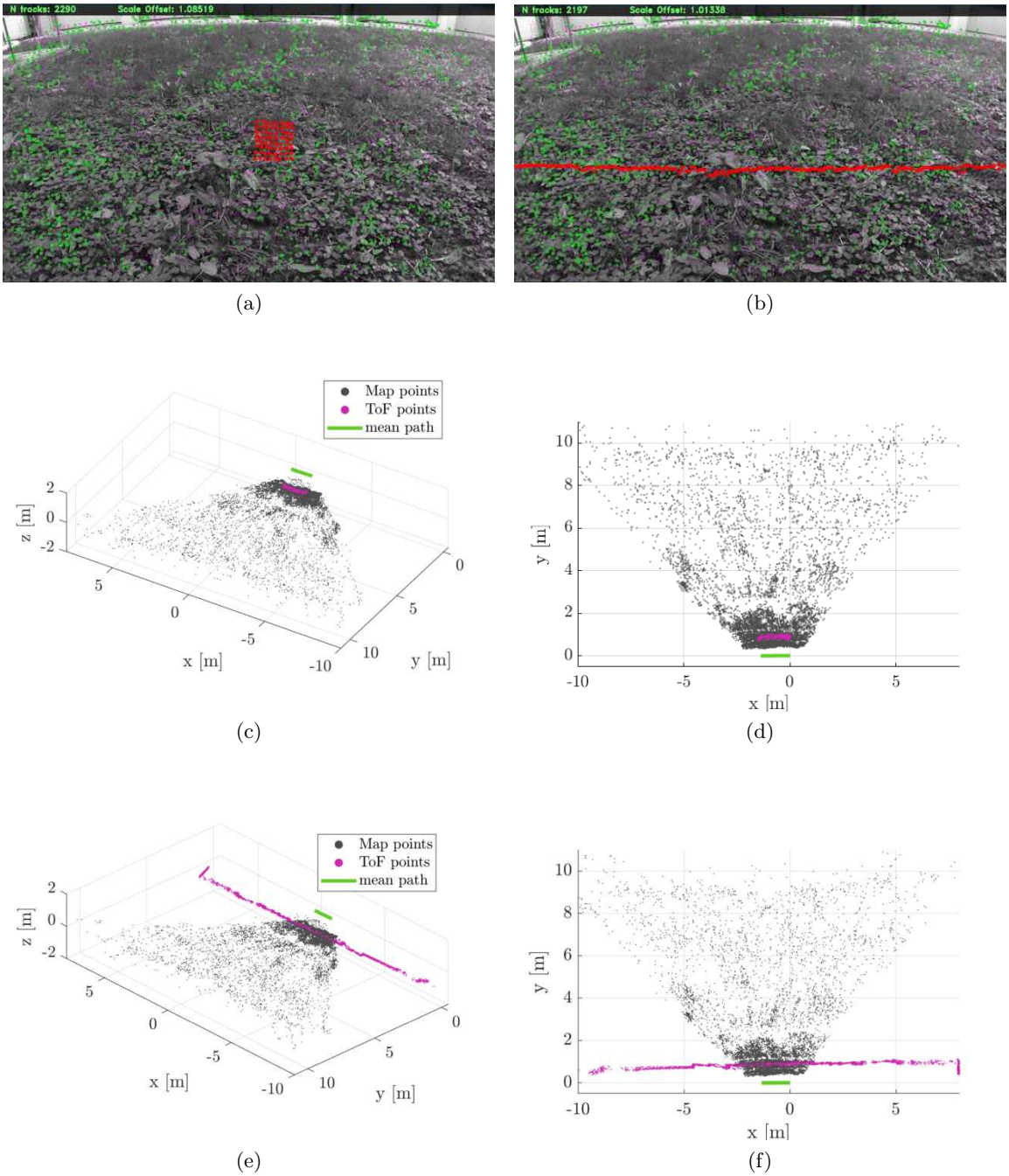


Figure 4.17: Evaluation results in the OUTDOOR sequence. The algorithm is tested in a variety of configuration, using both the LiDAR and ToF range measurements and switching between the 3D and 2D approaches. In addition, ORB-SLAM2 is tested in stereo mode using the full output from the ZED stereocamera. The "Scale offsets" plot does not contain information about ORB-SLAM2 as it considers the scale residuals using range sensors. These results are related to 20 independent runs, solid line is the median value, bands are first and third quartiles.

the trajectory and not being able to correct it from further optimizations.

OUTDOOR sequence

The outdoor sequence consists in a translational motion over a feature rich grassy scenario. The environment is approximately flat and easy for tracking and initialization. Figures 4.18a and 4.18b are example camera views with reprojected range measurements. Compared to INDOOR, in this sequences all configurations of the algorithm delivered good pose estimates with similar accuracies. Specifically, the 2D-tof configuration performed as well as the LiDAR one, exhibiting always similar Absolute Trajectory Errors along the path. The 3D-tof configuration performed slightly worse but never significantly, achieving just around a centimeter higher errors at the last pose compared to the other methods. A reason



(a) (b) (c) (d) (e) (f)

Figure 4.18: 3D reconstructions from the OUTDOOR sequence. (c) and (d) are 3D and top views of the reconstructed environment using the time-of-flight camera as the range sensors. Points in magenta are 3D locations of range measurements positioned in the environment from the estimated camera transformations. (e) and (f) are the reconstructions performed using the SICK PLS 312 LiDAR as range sensor. (a) and (b) are example views of the dataset superimposing ToF and LiDAR ranges respectively. Green circles denote initialized features

Sequence		2D-ToF		3D-ToF		2D-LiDAR		3D-LiDAR		ORB-SLAM2	
		RMSE	fATE	RMSE	fATE	RMSE	fATE	RMSE	fATE	RMSE	fATE
INDOOR	[m]	0.053	0.088	0.066	0.094	**0.023**	**0.034**	**0.035**	**0.051**	0.117	0.200
	[%]	3.94	6.50	4.89	6.94	1.71	2.49	2.61	3.80	8.68	14.79
OUTDOOR	[m]	0.017	0.037	0.021	0.035	**0.016**	0.029	**0.014**	**0.024**	0.021	**0.022**
	[%]	1.27	2.72	1.59	2.61	1.21	2.15	1.05	1.78	1.59	1.65

Table 4.3: Performance summary for all the VO configurations in the INDOOR and OUTDOOR short sequences. Results are reported in terms of RMSE (Root Mean Square Errors) as well as percentual over the total trajectory length of 1.35 meters.

for this is that computing the scale offset as the difference between the range measurement and the average depth of neighboring landmarks might be inaccurate if the camera is not looking perpendicular to the ground plane, which is the case of this evaluation as the camera was inclined about 30 degrees from the horizontal plane. Also in this case, the scale offset plot (figure 4.17b) shows a good behavior of the algorithm in all configuration as the scale residuals were always converging to the value of 1 even if case of sudden scale errors. In particular, the VO in configuration 3D-ToF struggles at the beginning of the trajectory losing scale consistency but recovers quickly in the following poses.

A summary of the performances in the INDOOR and OUTDOOR short sequences in given in table 4.3. Bold characters highlight the two highest performing configurations for each sequence. Unsurprisingly, the VO in conjunction with the 2D LiDAR performs always better than the time of flight camera. However, the penalty in pose estimation accuracy using such a low resolution sensor is not always significant. While in the more challenging INDOOR sequence using the time of flight camera almost doubles the error with respect to the LiDAR, in the OUTDOOR scenario the figures are absolutely similar to the point that the performance gap is irrelevant. Considering also the difference in power requirements, weight and cost (refer to table 4.1), the choice of using such sensors is intriguing, especially for resource-constrained platforms such as UAVs or small exploration vehicles.

4.7.2 Long range experiments

To test our VO algorithm on larger scale motions, the sensor setup was mounted on our mobile robotic platform (shown in figure 4.5b) along with a differential GPS for a precise ground truth with centimeter-level accuracy. The rover was manually driven while recording a stream of stereo images, ToF and LiDAR scans and D-GPS data. All sensor data was recorded at each individual refresh rate and synchronized offline at 4Hz, which is the scan frequency of the SICK PLS 312 LiDAR. Full stereo frames were recorded in order to test ORB-SLAM2 as in the previous sequences, however our monocular VO used the left camera frames. A sequence was captured at the Europa park in Padova and features a natural scenario, mostly comprising an almost flat grassy field. As the sensor setup was tilted downward by approximately 30 degrees, both the time of flight camera and the LiDAR were always able to detect the ground at a reasonable distance of around 1.5 meters. The rover was driven on a curvilinear trajectory of around 60 meters length which can be visualized in figure 4.21a along with some example camera views.

Sequence		2D-ToF		3D-ToF		2D-LiDAR		3D-LiDAR		ORB-SLAM2	
		RMSE	fATE	RMSE	fATE	RMSE	fATE	RMSE	fATE	RMSE	fATE
EUROPA	[m]	**1.742**	**3.049**	6.828	12.759	**0.549**	**0.865**	2.512	4.540	6.298	12.379
	[%]	3.09	5.41	12.12	22.65	0.97	1.53	4.46	8.06	11.18	21.97

Table 4.4: Performance summary for all the VO configurations in the EUROPA park sequence. Results are reported in terms of RMSE (Root Mean Square Errors) as well as percentual over the total D-GPS trajectory length.

With this test we want to evaluate the capability of our VO of reconstructing the camera trajectory over longer distances while keeping a correct metric scale. The error metrics used for performance evaluation are the same as in the previous test, which are the Absolute Trajectory Error, the instantaneous scale drift and the trajectory length drift. Figure 4.19 reports all these errors computed over 20 runs of the algorithm in all configurations. The medians of all errors are plotted using solid lines while the color bands highlight the 25% and 75% percentiles. The sparsity of results gives in fact a qualitative description of the uncertainty of the estimation. To compute the errors with respect to the ground truth, firstly the trajectory estimates were ridigly aligned with the D-GPS data using Horn's method [66] knowing pose to pose correspondences from timestamps. While most widely used Visual SLAM evaluation pipelines [135] suggest to use all the available correspondences across the whole trajectory, we believe that for longer sequences this results in overestimating the reconstruction accuracy. By constraining all poses, the estimated trajectory is aligned over the ground truth such that the total misalignment is minimized, dampening the effect of angular drifts which cause the trajectories to depart from each other. For this reason we chose to align only the first 500 frames roughly corresponding to 15 meters of ground truth length.

The pose errors plotted in figure 4.19a confirm the findings of the previous tests. While the LiDAR configurations allow to obtain more accurate and repeatable results, using the noisy time of flight camera is always a viable solution especially in the 2D-ToF configuration. The 3D-ToF proves to be the most inaccurate and uncertain as it reaches in average errors higher than 10 meters at the end of the trajectory. The sources of this error are both scale drifts, which are visible observing the estimated trajectory length (figure 4.19c), as well as angular drifts which are visible in figure 4.20c. The 2D-ToF configuration performed however on par with the 3D-LiDAR one as far as scale stability is concerned. A lower angular drift also allowed to obtain a more fitting trajectory to the ground truth resulting in very low absolute pose errors. As in the previous sequences however, the highest performing configuration was 2D-LiDAR benefitting from more accurate and distributed range measurements across the camera field of view. Overall, all the VO configuration performed comparably or better than ORB-SLAM2 stereo in the same scenario, which showed very similar performances to our 3D-ToF setup. As a matter of fact, the camera views were always angled towards the ground, observing a rapidly moving environment in the close proximity of the rover.

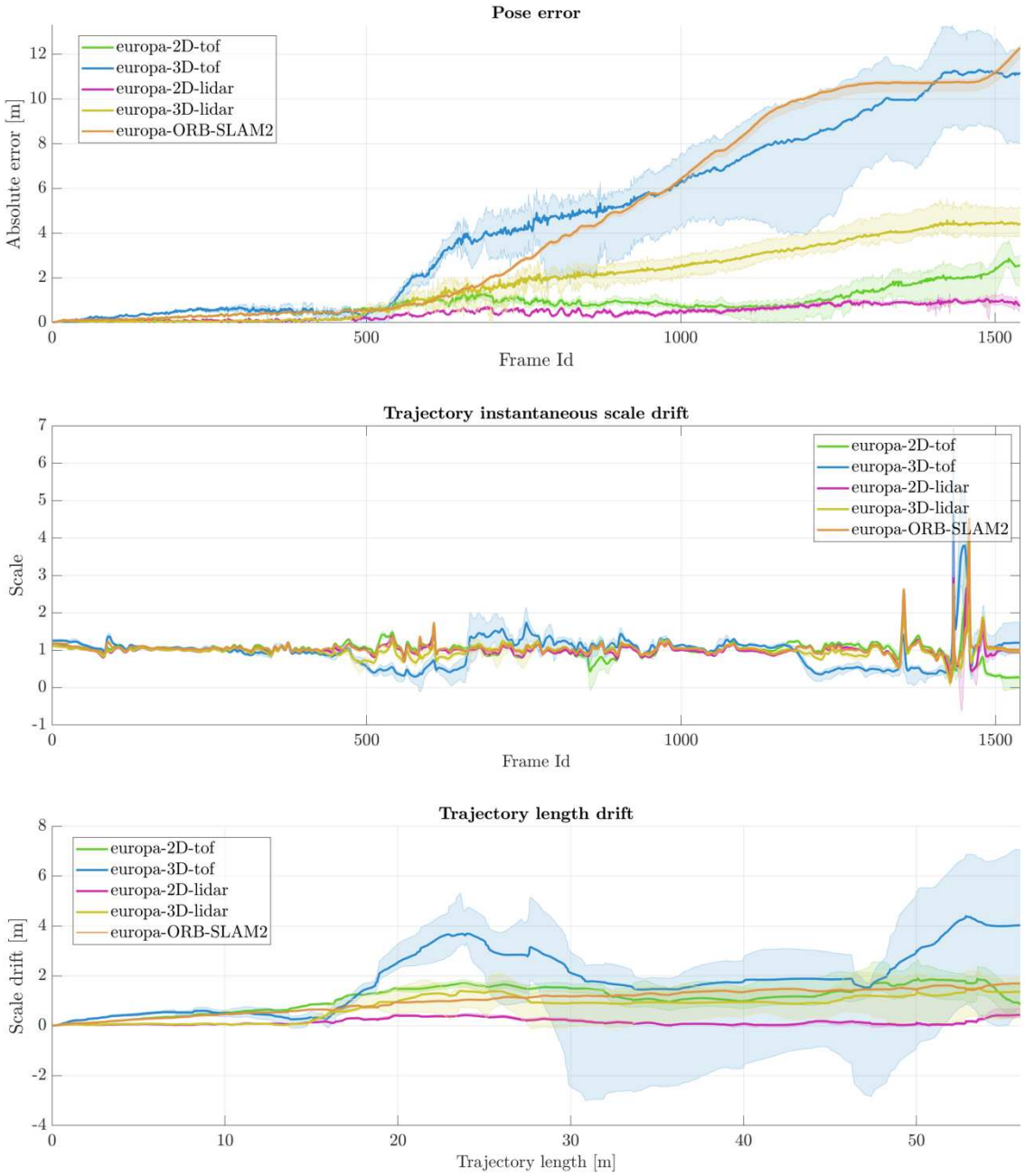
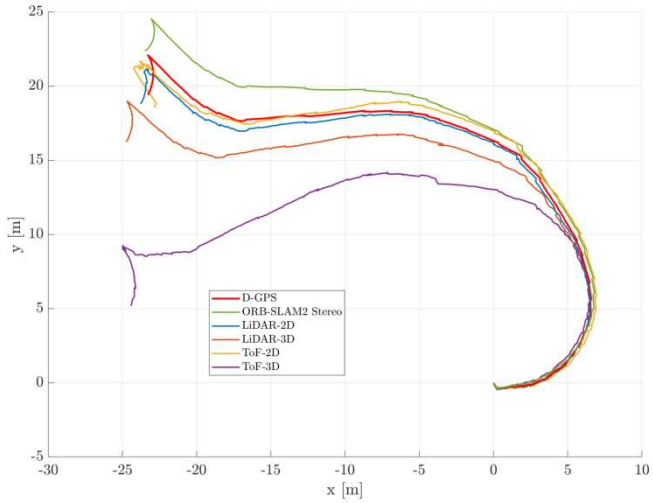
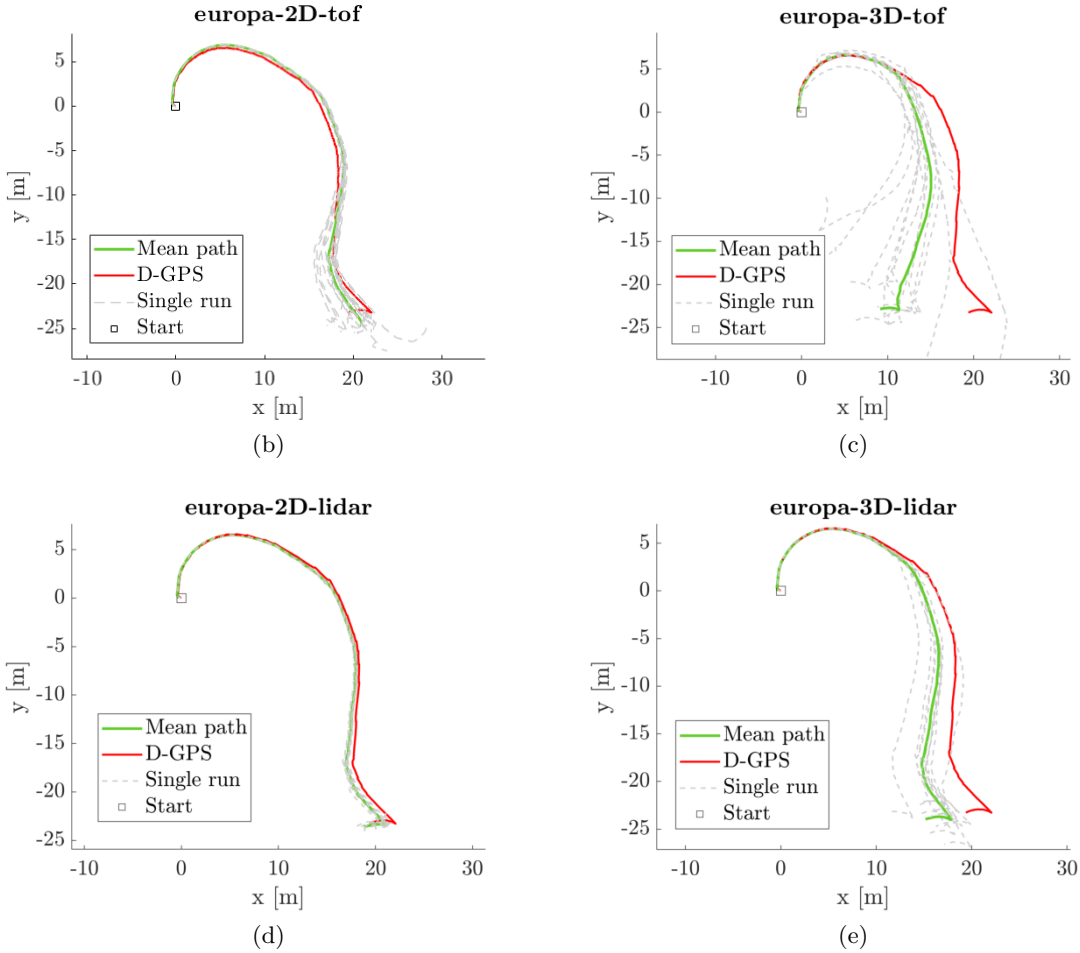


Figure 4.19: Absolute Trajectory Error, Instantaneous Scale Drift and Trajectory Length Drift for all configurations in the EUROPA park sequence. ORB-SLAM2 is tested for performance comparison with a state of the art stereo visual SLAM system. Solid lines and bands are respectively median and 1st-3rd quartiles over 20 runs.

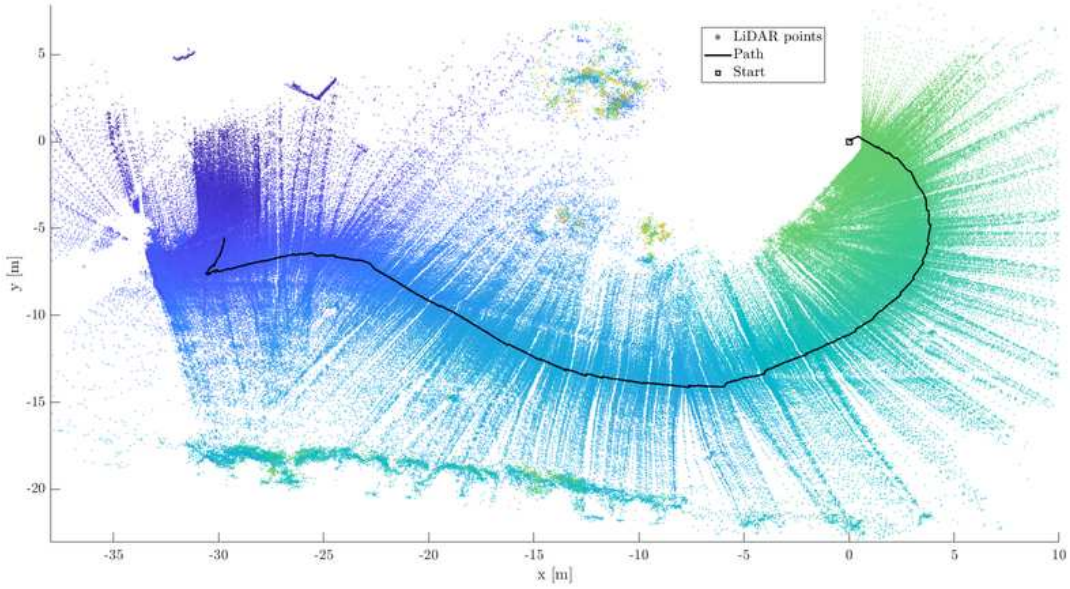


(a) Best trajectories compared

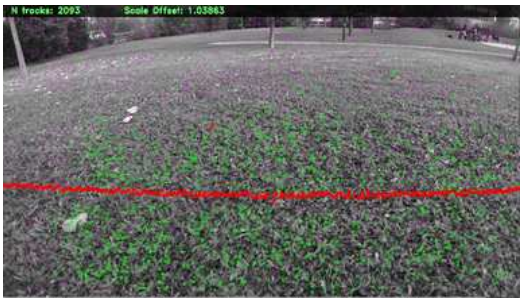


(b) (c) (d) (e)

Figure 4.20: Trajectories (top view) for the EUROPA sequence, the VO is tested in all configurations. Figure 4.20a compares the mean trajectory of all configurations including ORB-SLAM2 in stereo mode. The remaining plots reports, for each configuration, the mean trajectory and the 20 individual runs from which the mean is computed. This gives an overview of the repeatability of each sensor choice



(a) Trajectory from 2D-LiDAR and scan points



(b)

(c)

Figure 4.21: View from the EUROPA dataset using both the Time of Flight camera and the LiDAR sensor. Figure 4.21a shows the results of stacking 2D LiDAR scans along the trajectory using the estimated camera poses. In the point cloud can be recognized some features belonging to the environment such as fences, trees and a walking path. Although this evaluation is willing to prove that our algorithm allows to use light and low resource requiring range sensors, its generality with respect to the range sensor type opens the possibility of using 2D or 3D LiDARs for more complete mapping purposes.

Chapter 5

Scale Correct Monocular Visual Odometry Using a LiDAR Altimeter

This chapter presents a monocular Visual Odometry pipeline, which we refer to as *aVO*, designed to be fast and computationally inexpensive, making use of a range sensor of the lowest resolution possible (1 point) to correct *scale drift*. This algorithm was first presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018 [51]. Up to our knowledge, at the time of publication, this was the first attempt at integrating one-point range measurements from LiDAR data in a monocular visual odometry pipeline, embedding single range corrections factors in the optimization back-end.

5.1 Algorithm Overview

This section presents the Visual Odometry pipeline, main focus of this chapter, relatively to its functional structure and properties, highlighting the contributions to the state of the

Figure 5.1: Simple overview of range sensor types. From left to right, 3D sensors such as the Velodyne Puck family provide the highest information content at the price of a higher weight and power required. In the middle, 2D LiDARs such as the SICK LMS family represent a compromise between mass and power budget while on the right, LiDAR altimeters provide the lowest amount of information possible as well as being lightweight and inexpensive.

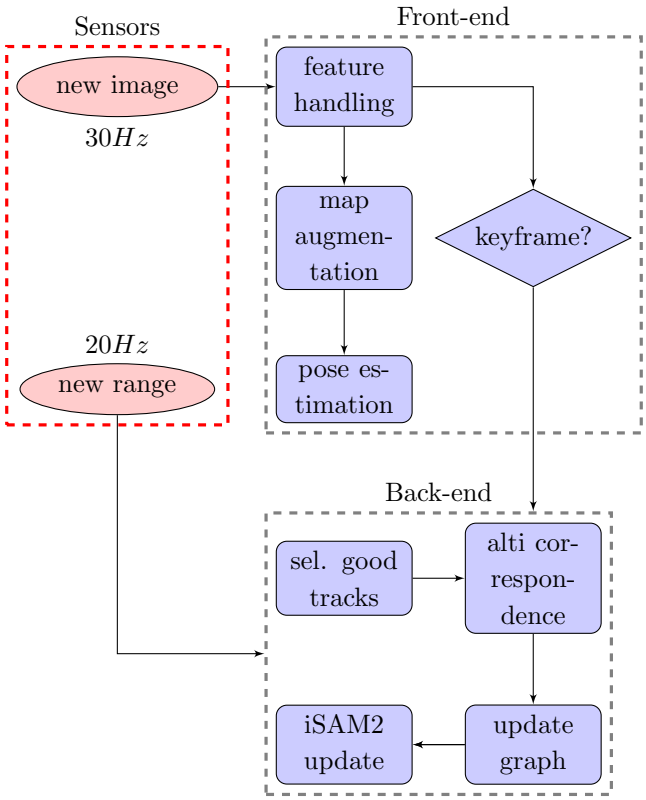


Figure 5.2: General overview of the complete aVO pipeline. Red colors denote the inputs such as images and range measurements. Blue colors denote the functional parts of the algorithm, such as the members of the front and back ends.

art that it provides. As figure 5.1 shows, multiple choices are available for the selection of LiDAR sensors, however the mass and power requirements of 3D and 2D LiDARs might not be suitable for small and lightweight flying vehicles. In addition, this work proves that by using single range information from a LiDAR altimeter, which is a commonly used sensor for UAV navigation, it is possible to achieve state of the art performances in terms of tracking accuracy even compared with scale aware RGB-D and stereo visual systems.

Figure 5.2 is a schematic representation of the aVO pipeline as a flow diagram of the main functional tasks involved. Inputs to the algorithm are images from a monocular vision system (single camera) provided usually with a frame rate of 30 Hz and range measurements from a laser altimeter (a LightWare SF-10B[1]) provided in our case with a rate of 20 Hz. Images are used to concurrently build a map of 3D *landmarks* as well as to compute the camera positions with respect to that map as more frames are received. This process is the main focus of the *Front-end* part, which occupies one of the two threads of the multi-thread architecture of aVO. The *Front-end* is also in charge of selecting a sparse set of frames called *keyframes*, whose position is optimized in the *Back-end*. The *Back-end* runs on a second thread to ensure *real time* performances and is in charge of optimizing a recent history of keyframe poses and 3D landmark coordinates in order to reduce the trajectory drift. In this step, altimeter measurements are used as an additional constraints in the

[1]https://lightware.co.za/

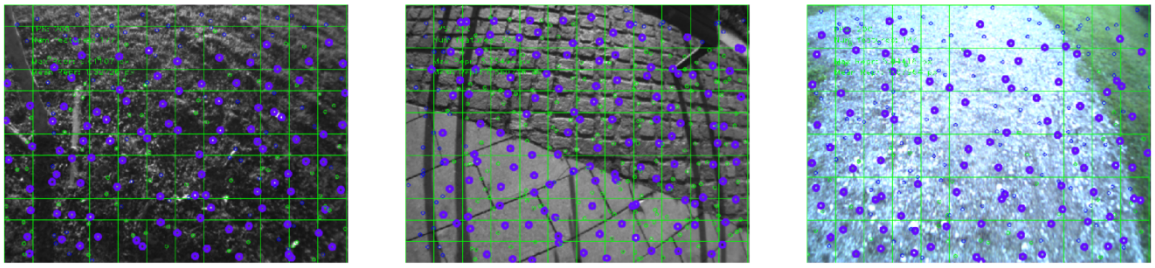
optimization to make sure that the distance between a limited set of 3D landmarks and the camera center is in accordance to the true scale. Once optimal values for the camera poses and map are computed, corrections are forwarded to the *Front-end* thread for pose tracking.

5.2 Algorithm Front-End

This section presents the *Front-end* of aVO. This is the functional part of the algorithm that is in charge of feature detection and tracking, pose estimation and map augmentation for each incoming image.

5.2.1 Feature Detection and Tracking

Between all the available feature detectors such as SIFT [84], SURF [8] (used in the previous chapter), and others, for the sake of computational efficiency, the FAST (Features from Accelerated Segment Test) [112] corner detector is selected. The FAST detector is in fact capable to extract corner features from 640x480 images requiring about 1 millisecond for 200 corners, therefore occupying a minor part of the theoretical 33 milliseconds per frame to accomplish the required 30 Hz update rate. Additionally, aVO does not rely on feature matching to obtain correspondences across multiple frames therefore a significant amount of time is saved from not having to compute feature *descriptors*. The number of features that are detected by the front end is limited in order to save unnecessary computational time. Having a big number of feature tracks, and therefore 3D landmarks, can be beneficial for robustness but not necessarily for tracking accuracy. For this reason, instead of selecting whatever number of features from an image giving a fixed threshold for the FAST detector, each image is divided in a grid of given dimension and only one corner is selected in each cell. The number of divisions in u and v directions is provided as a parameter to the algorithm and is selected depending on the type of environment. For outdoor scenarios, where images usually contain good visual information, dividing the image in 10x10 or 11x11 grid sizes (see figure 5.3) usually deliver good results. If images are lacking visual information in some parts, the grid can be set up with more divisions such that more features are searched. At feature detection time, each cell is searched for FAST features, and the one with highest corner score (regarding corner thresholds for the FAST detector, the reader is referred to reference [112]) is selected. Doing so, when necessary, features are extracted in the most uniform way possible and in the least number for having accurate pose estimation. As the camera moves and new images are acquired, only empty cells are searched for new features. If a cell is occupied by a tracked feature, no detection of new corners is performed. The detected FAST corners are tracked in subsequent frames using a *sparse optical flow* approach. Instead of computing *descriptors* and matching them across each consequent image, a Pyramidal Lukas Kanade Tracker [15] focuses on a small patch of pixels centered around each feature point and aligns it with the closest one in the next image. This approach is robust to small scale and rotation differences between consequent frames and allows to maintain tracks of the same features for long frame windows. While requiring a greater computational effort than FAST extraction, feature tracking does not



(a) example 1 (b) example 2 (c) example 3

Figure 5.3: Example views related to the aVO front end. The 3 views highlight the grid division (green lines) as well as the detected corners (green dots). Small blue dots are tracked observations of uninitialized landmarks and big blue dots are corners with the highest metrics.

impact excessively the timing budget per each frame, requiring in average 2 to 5 milliseconds for reliable tracking depending on the number of features and their relative displacements between frames. The Lukas Kanade tracker utilizes a patch window of fixed size which is set to 11 pixels as a tradeoff between computational complexity and robustness to fast motions.

Keyframe Selection

Being targeted at real time usage, the aVO algorithm process every image that is fed from the camera and estimates the related 6D pose. Due to computational limits and limited memory storage, it is not feasible to keep in memory all the images that are received. The camera trajectory is therefore summarized in *keyframes* which serves as snapshots of camera poses and landmark observations in sparse moments along the trajectory. More specifically, in the context of aVO, a keyframe K_i contains the pose of the camera at the time of keyframe selection $T^W_{C,i}$, a set of undistorted observations of initialized landmarks $\{x_0, x_1, \ ..., \ x_n\}_i$ and a range measurement provided from the altimeter with the closest timestamp d_i. All the information stored in K_i serve later in the optimization scheme, where both camera poses and 3D landmarks are modified to better fit the observations. This way, not all the poses computed at 30Hz are optimized but just those contained in keyframes. Keyframes are selected based on a decisional process which tries to satisfy an optimal sampling scheme as a tradeoff between computational complexity and estimation accuracy. The first requirement is satisfied by limiting the number of lost feature tracks between two consequent keyframes. If too little tracks are shared between two keyframes,

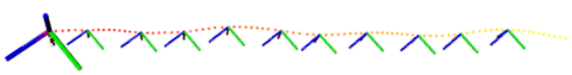


Figure 5.4: Trajectory and keyframes during a forward camera motion. The full trajectory is represented by colored dots while selected keyframes are represented as frames. The blue axis is the camera z axis, or the view line.

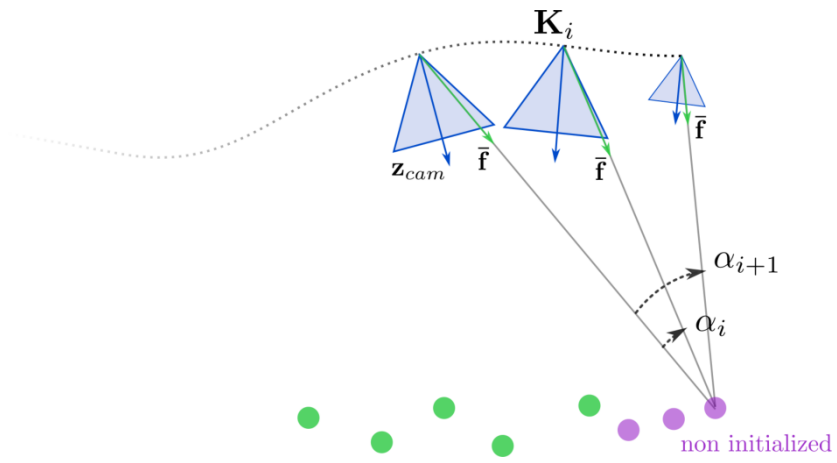


Figure 5.5: Parallax between observations of 3D landmarks between consequent views. Blue triangles represent camera positions, $\mathbf{K}_i$ denotes the last keyframe, while the smaller camera is the current position (not necessarily a keyframe). Green dots are initialized points, or points for which the depth is known while pink points are not initialized. α is the parallax between observations, which are represented as green arrows. The higher the parallax the lower will be the uncertainty over the 3D position of the landmark.

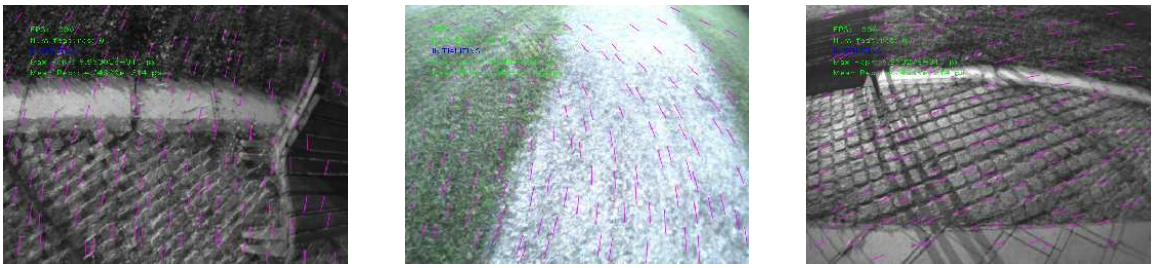
the optimization scheme can diverge and lead to inconsistent results as the pose sequence would have a soft constraint at some point. Contrarily, if all feature tracks are shared, the relative pose between the two keyframes is correct enough so that the optimization scheme does not have a significant impact. However, when the motion is dominant along the camera line of sight, all tracks could be kept and the distance between two consequent keyframes could be very high. In order to obtain a spatially uniform set of keyframes, a constraint on the geometric distance between them is imposed. Each new keyframe is therefore triggered if either one of the following conditions are met:

$$\begin{cases} \text{num}(\hat{x}(t_i))/\text{num}(\hat{x}(t_{i-1})) < 0.8 \\ ||t_i - t_{i-1}||/\rho_i > 0.15 \end{cases} \tag{5.1}$$

where $\text{num}(\hat{x})$ is the number of feature tracks shared between the current frame in position t_i and the last keyframe in position t_{i-1}. ρ_i is the mean depth of all landmarks observed from the last keyframe and $||t_i - t_{i-1}||$ is the euclidean distance between the current position and the last keyframe position. The effect of this keyframe selection scheme can be visualized in figure 5.4 where it is highlighted how keyframes are sampled from the full camera trajectory.

5.2.2 Map augmentation

Contrarily to stereo or RGB-D systems, single images lack depth information. In order to know the 3D location of the observed landmarks it is necessary to have at least two views of the same point from different camera positions. This way, *triangulation* allows to estimate a 3D location for the new landmark. New FAST corners extracted by the front end have no 3D information associated, therefore they are just tracked in each new frame without being useful for pose estimation. As the camera moves, parallax between the first and the current



(a) example 1 (b) example 2 (c) example 3

Figure 5.6: Example views related to the aVO initialization scheme. The 3 views show in purple lines the feature tracks across several frames which are used to compute the 2-view-geometry necessary for initializing a map and two camera poses.

observation grows. In order to triangulate a point with high accuracy, the parallax should be higher than a minimum threshold so that the influence of feature tracking noise over the 3D location of the triangulated points is minimized. Residual errors are corrected by the optimization back-end. Let be $\mathbf{p}_j$ the observed 3D landmark whose depth is unknown. Let also be $\mathbf{T}_i^W$ and $\mathbf{T}^W$ the pose of the i-th keyframe and the current camera respectively. Using the notation of figure 5.5, let be $\mathbf{f}_i$ and $\mathbf{f}$ the feature position in pixel coordinate of the point $\mathbf{p}$ in the i-th keyframe and current camera images. Multiplying the image features x_i and x to the inverse of the calibration matrix $\mathbf{K}$ returns the unit vectors $\mathbf{v}_i$ and $\mathbf{v}$ connecting the cameras optical center and $\mathbf{p}$.

$$\begin{aligned} \mathbf{v}^W &= \mathbf{T}_C^W \mathbf{K}^{-1} x \\ \mathbf{v}_i^W &= \mathbf{T}_{C,i}^W \mathbf{K}^{-1} x_i \end{aligned} \tag{5.2}$$

The angle between $\mathbf{v}_i$ and $\mathbf{v}$ expressed in the world reference frame is the parallax angle between the two observations:

$$\cos(\alpha) = \mathbf{v}^W \cdot \mathbf{v}_i^W \tag{5.3}$$

The parallax angle is computed for each non-initialized track in each new frame until a threshold is met, after which the 3D point $\mathbf{p}$ is triangulated using the DLT algorithm [62] (see also section 2.3.3) and inserted into the map. $\mathbf{p}$ can now be used for pose estimation and its coordinates will be optimized by the algorithm back-end.

5.2.3 Initialization

As the first image is received from the monocular Visual Odometry pipeline, the system is only aware of the projections of landmarks on the image frame. Depth and 3D coordinates are unknown and must be recovered by *initialization* of the monocular algorithm. Classical approaches from *Structure from Motion* [62] [121] [98] are here employed in order to recovery both the environment structure and camera poses from two overlapping views of an environment. From the first detection, corner features are tracked in consequent frames until a mean distance of tracks is reached, ensuring sufficient parallax. Let be x_j and $x_{j'}$ the normalized metric coordinates of matching features which lay respectively in the first

and second frame of the selected pair for initialization. In case the geometry of the observed environment is mostly planar, it is efficient to compute the *homography* transformation [73] [43] which transforms the observed plane in the two views:

$$x_j\prime = Hx_j \quad j = 1, ..., n_{matches} \tag{5.4}$$

H is the *homography* matrix and embeds the parameters defining the rotation and translation between the views. Eq. 5.4 is solved as a least-squares problem embedded in a RANSAC scheme to mitigate the effect of outliers (diverged feature tracks). According to [88], the matrix H can be decomposed in 4 sets of rotation R and translation t matrices which must be tested in order to ensure that all the triangulated points lie in front of both cameras (*chierality* constraint) and that both views observe the environment from the same side. In case the environment is not planar dominant, a homography can not describe robustly the two view geometry, in this case a more suitable representation of the problem comes from the *fundamental matrix*:

$$x_j\prime F x_j^T = 0 \quad j = 1, ..., n_{matches} \tag{5.5}$$

Equation 5.5 encodes a coplanarity constraint between pairs of matching features [101]. In short, the plane containing the 3D landmark $\mathbf{p}$ as well as the two camera optical centers contain also the image features x_j and $x_j\prime$. The fundamental matrix can also be expressed as:

$$F = K^{-T}[t]_x R K^{-1} \tag{5.6}$$

where $[t]_x$ is the cross product matrix of the translation vector t and R is the rotation matrix between the two views (or cameras). R, t can be obtained both from the fundamental or the homography matrix as explained in section 2.3.1 and 2.3.2. To conclude the initialization process, R, t and the 3D landmark coordinates p_j are optimized in a full Bundle Adjustment for the two views, more detail on the optimization process in section 5.4.

As mentioned previously, the process of recovering environment structure and camera poses from monocular estimations is affected by loss of scale information (see figure 2.3). In order to initialize the scale, correspondences are searched in the second image from landmark projections and the altimeter projection as explained in detail in the following section 5.3. A scale factor is computed as follows:

$$s = \frac{1}{n_{corr}} \sum_{j=1}^{n_{corr}} \frac{\hat{d}_j}{d} \tag{5.7}$$

where d is the range measurement from the altimeter and $\hat{d}_j$ are the depths (or L2 norms of X_j^C) of landmarks whose projections are neighbors to the altimeter projection in the frame (see figure 5.7). The scale factor s is multiplied to both the translation vector t and the 3D landmark coordinates X_j to obtain a scale correct initial reconstruction. Scale errors originating from uncertainty over 3D landmarks coordinates are mitigated successively during the Incremental Bundle Adjustment (section 5.4)

5.2.4 Pose Estimation

After the monocular pipeline has been initialized and a first map is available, the camera pose can be compute frame-wise by tracking the image features related to initialized landmarks (landmarks for which 3D location is known) and solving the following problem:

$$\underset{\mathbf{T}_C^W}{\operatorname{argmin}} \frac{1}{2}\sum_{j=1}^{n} \rho||f_j - \pi(X_j, \mathbf{T}_C^W)||^2 \tag{5.8}$$

where $\mathbf{T}_C^W$ is the pose of the camera with respect to a common global reference system, X_j are 3D coordinates for landmark j and x_j is the measured projection of that landmark, or the undistorted feature track (section 5.2.1). π denotes the *reprojection* function derived from the pinhole camera model:

$$x_j = K\mathbf{T}_C^W X_j^W / z_j^C \tag{5.9}$$

where z_j^C is the z coordinate of landmark j with respect to the current camera pose and K is the camera matrix. Equation 5.8 is solved as a non-linear least squares problem using the Levenberg-Marquardt algorithm. As with any (non-convex) optimization problem, in order for the solution to not stabilize in local minima, an initial estimate for the variable to optimize must be provided. In this case, as the problem in equation 5.9 is solved for each incoming frame, the initial estimate for $\mathbf{T}_C^W$ can be the pose for the previous frame $\mathbf{T}_{C,t-1}^W$ which in case of moderate camera speed should be significantly close to the solution. However, in order to cope with fast camera motions, the pose $\mathbf{T}_C^W$ is predicted employing a constant velocity model. For frame i, the instantaneous linear and angular velocities $\mathbf{v}_i$ and ω_i, computed from frames (i-1) and i, are used to compute the new camera position t_{i+1} and orientation q_{i+1} in quaternion form:

$$\begin{aligned} t_{i+1} &= v_i \Delta t \\ q_{i+1} &= q_i q(\omega \Delta t) \end{aligned} \tag{5.10}$$

where q_i is the quaternion of the $i-th$ pose while Δt is the time interval between frames (usually 33 milliseconds for a 30Hz frame rate) and ω is the estimated angular velocity. This helps also in reducing the computational impact of pose estimation as when the starting value for the optimization is closer to the solution, the number of iterations required are lower.

Equation 5.9 also include a loss function denoted by the term ρ whose effect is to damp the influence on outliers in the optimization scheme. The selected function is the Cauchy loss [80]:

$$\rho(s) = log(1+s) \tag{5.11}$$

where s is the L2 norm:

$$s = ||x_j - \pi(X_j, \mathbf{T}_C^W)||^2 \tag{5.12}$$

If a feature track has diverged, meaning that the 2D measurement of the landmark position drifted from the original FAST keypoint, the cost function for that particular landmark

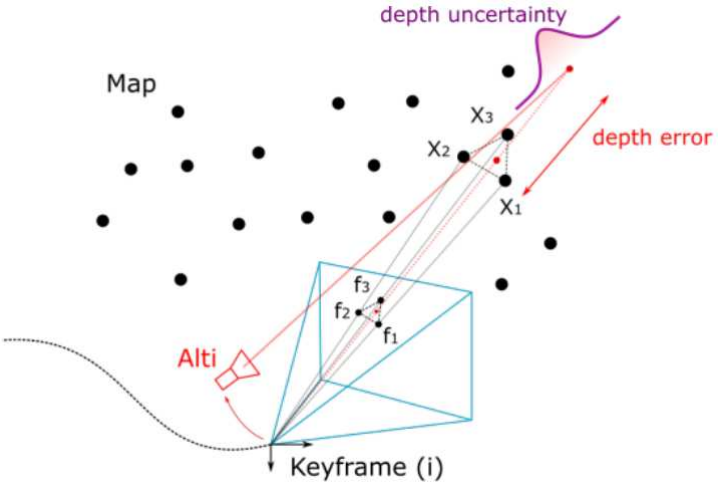


Figure 5.7: Simplified representation of the camera-altimeter association scheme. Range measures are interpreted as single 3D points in the camera local reference frame. It constrains the depth of landmarks whose reprojection on the image plane is close to the altimeter one.

will never converge to 0 and the Jacobians would divert the search for the global minimum of equation 5.8. Using the Cauchy loss function however the influence of outliers are minimized.

5.3 Range Information Fusion

In order maintain a correct global scale, range measurements must be correlated to visual information. When a new keyframe is created, the closest range measurement according to its timestamp is stored. Each range measurement is represented as a single 3D point in the camera local reference frame through knowledge of the extrinsic parameters (rotation and translation) which relate the camera and altimeter reference frames. 3D landmarks are associated to range measurements in terms of the proximity of their reprojection in the camera image plane to the projection of each altimeter measure $x_{i,a}$. Therefore, image features x_j of landmarks constrained by the depth d must satisfy the following constraint:

$$||x_j - x_{i,a}|| < t_{max} \qquad j = 1, ..., n \tag{5.13}$$

where t_{max} is a user-defined threshold (usually in the range from 10 to 20 pixels). $x_{i,a}$ is computed as:

$$x_{i,a} = \pi\Big(T_a^C d_i \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\Big) \tag{5.14}$$

where π is the projection in pixel coordinates on the camera image plane and T_a^C is the extrinsic transformation between the altimeter reference frame and the camera reference frame. A thorough explanation of the calibration procedure is given in section 5.5. For the sake of robustness during the incremental bundle adjustment, associations between

landmarks and range measurements are considered valid only if the depth of the candidate landmarks are similar. In other words, if the hypothesis of planar environment holds, as it is in the case of a downward looking camera from an UAV, the mean depth of candidates should be equal to the measured depth from the altimeter with a certain uncertainty. However, in case of large variations of candidate landmark depths, as in the case of edges of obstacles, it is not clear how to predict the correct depth values. Enforcing range constraints in this case could eventually lead to instabilities during the optimization. For this reason, an association is accepted only if the following constraint is validated:

$$\sqrt{\frac{1}{n_{cand}} \sum_{j=1}^{n_{cand}} ||d_j - \mu_d||^2} < 0.05\text{m} \quad j = 1, ..., n_{cand} \tag{5.15}$$

The threshold of 5 centimeters from the standard deviation of depths is a user defined parameter and in all tests was sufficient to avoid dangerous associations.

5.4 iSAM2-based Optimization Backend

Errors in camera pose estimation and landmarks triangulation results in trajectory *drift* and map inconsistency. Even if landmarks are triangulated with a sufficient parallax, their 3D position is still erroneous to a degree. Tracking of image features using a sparse optical flow approach is affected by errors depending on image noise and texture information. In addition, tracking is performed on raw distorted images for time complexity reasons therefore lens distortion can cause more tracking errors at the limits of the camera field of view. In order to cope with these errors, Bundle Adjustment is performed to optimize both the 3D landmark coordinates as well as a history of keyframe poses given the available observations. Let be $\mathbf{T}_{C,i}^{W},\ i = i_0, ..., n_c$ keyframe poses in the world reference frame and $X_j^W,\ j = j_0, ..., n_l$ landmark coordinates expressed in the world reference frame which are observed by the considered keyframes. Recalling the notation from the previous paragraphs, let be $\hat{d}_j$ the depths of landmarks which are constrained by the depth measurement d_i from the i-th keyframe. The process of performing Bundle Adjustment consists in solving the following non-linear least squares problem:

$$\underset{\mathbf{T}_{C,i}^{W}, X_j^W}{\operatorname{argmin}} \frac{1}{2} \sum_{j=j_0}^{n_l} \sum_{i=i_0}^{nc} ||x_j - \pi(X_j^W, \mathbf{T}_{C,i}^{W})||^2 \ + \ \frac{1}{2} \sum_{i=i_0}^{n_c} \sum_{j=j_0}^{n_a} ||\hat{d}_j - d_i||^2 \tag{5.16}$$

n_c is the total number of keyframes to be optimized, n_l is the total number of landmarks to optimize and n_a is the number of landmarks which are constrained by range measurements from each keyframe. Figure 5.8 shows a factor graph representation of the Bundle Adjustment problem formulated with equation 5.16. Green dots are keyframes poses and are connected by dashed lines to the landmarks (purple dots) that they observe. Dots, or parameters to be optimized are *nodes* in this graph, while arrows encode some cost functions whose parameters are the starting and ending positions. Grey dashed arrows represent *reprojection* factors, which are the first component of the full cost function in equation 5.16. Red arrows represent instead *depth* measurement and involve both the keyframe pose

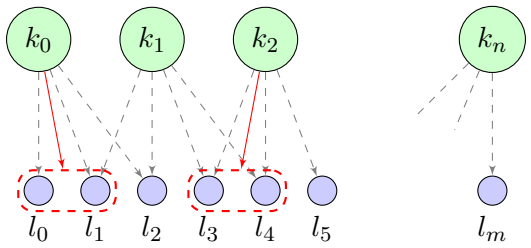


Figure 5.8: Factor graph representation of the monocular SLAM problem with altimeter measurements. Green and blue nodes are keyframe and landmark poses respectively. Grey dashed lines are image projection factors while red solid lines represent range factors. The red dashed lines enclose the landmark nodes that are immediate neighbours to the altimetric measure (singular 3D point expressed in the keyframe local reference frame).

as well as multiple landmarks between the observed ones. Multiple landmarks are in fact constrained by a range measurement if their projection is in a radius from the altimeter projection (see section 5.16 and figure 5.7). Red arrows depict the second contribution in the cost sum of equation 5.16. The figure highlights how keyframe poses are constrained by shared observations of the same landmarks, the more interconnected the keyframes, the more accurate the solution of the optimization. No constraints are here defined between keyframes, which could be established by other odometry sources such as integration of IMU terms. The Bundle Adjustment problem formulated for the aVO pipeline helps in maintaining consistency in the environment geometry and trajectory as well as constraining the global scale of the motion thanks to range measurement, which would eventually drift as well in a monocular system.

The iSAM2 [71] incremental optimizer from the GTSAM[2] library is choosen as the reference framework for implementing Bundle Adjustment. Contrarily to other optimization frameworks where the local window of keyframes and landmark to optimize must be explicitly defined, iSAM2 provides functionality to detect which *nodes* will be influenced the most by the current update step and marginalizes all the rest. This allows also to maintain a single graph of keyframes and landmarks from which covariances can be extracted, if requested, for performing sensor fusion tasks.

In order to define a well-posed optimization problem, the structure of the graph must be built efficiently. Landmarks observed by only two keyframes and then discarded, as an example, have a limited impact on the optimization results and add useless computational complexity to the problem. Landmarks constrained by depth measurements, if triangulated from wrong feature tracks, can immediately lead to divergence of the optimization since their measurements are false. Additionally, since the back-end of the iSAM2 pipeline is a Gauss-Newton algorithm, to little constrained landmarks (triangulated from low parallax) leads to instability of the regression because the derivatives of the cost function will tend to zero. To summarize, at frame trigger time, good landmarks for being involved in the optimization are selected if the following requirements are fulfilled in order:

- a landmark must be observed by *at least* three keyframes.

[2]https://borg.cc.gatech.edu

- the reprojection error for each landmark must be limited to a certain threshold.
- each landmark must have positive z component in the camera reference frames of all keyframes.
- the depth of each landmark must be contained in a multiple of the mean scene depth

5.5 Altimeter-Camera System Calibration

Fusion of range measurement from a 1D sensor such as a LiDAR altimeter with visual information poses a great challenge from the implementation perspective as the system must be extrinsically calibrated. In order to refer the 1D range information to the camera reference frame, where landmarks are expressed, the transformation between the altimeter and camera must be fully known. This section provides a detailed formulation and validation of a novel calibration scheme to be used for evaluating the aVO pipeline on a proper test setup.

5.5.1 Extrinsic Calibration

The extrinsic calibration of a LiDAR altimeter and RGB camera system is challenging due do the 1-D nature of the range sensor. Typical solutions for calibrating 2D or 3D LiDARs and cameras rely on estimating the rototranslation between the sensors by aligning common observations of custom targets. The tranformation between the reference systems is infact univoquely defined. A LiDAR altimeter is a single range sensor whose measures can be represented as semilines, starting from the optical center of the altimeter and ending to the point of beam reflection. Referring an altimetric measurement to the camera reference frame can not be achieved by using a rotation matrix and translation vector since one of the three rotations is undefined (the one collinear to the optical axis). In this work, the altimeter is represented as a vector in the camera reference frame and the calibration procedure aims at estimating the origin and measurement direction of the sensor. Let be P_A a 3D point identified by a range measurement ρ and expressed in the camera reference frame. Its coordinates can be defined as:

$$\begin{aligned} P_A &= h(\rho, \text{calib}) \\ &= \rho \begin{pmatrix} cos(\alpha) \\ sin(\beta) \\ sin(\alpha) \end{pmatrix} + \begin{pmatrix} x_A \\ y_A \\ z_A \end{pmatrix} \end{aligned} \tag{5.17}$$

where the 5 DoF (Degrees of Freedom) calibration parameters α, β, x_A, y_A, z_A are defined according to fig. 5.9. The calibration procedure involves the usage of a planar checkerboard which defines a calibration plane. The target is observed with appropriately distributed orientations ensuring that each altimeter measure P_A lies on the calibration plane. From each checkerboard detection, the reference system of the target is determined and normals n_i as well as origins O_i are stored. In a first time, point-to-plane constraints identify a over-constrained linear system which is solved using least-squares minimization in a RANSAC

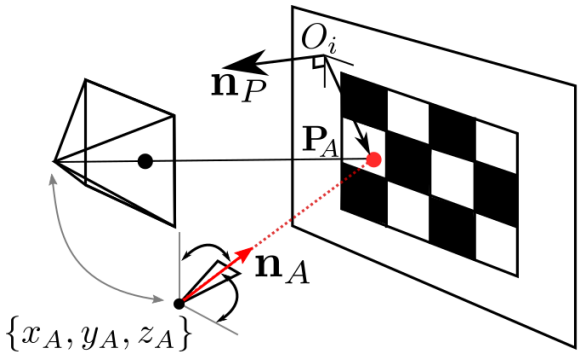


Figure 5.9: Extrinsic setup

scheme. The initial extrinsic parameters obtained in the first step are then refined in a non-linear least squares fashion using Levenberg-Marquardt.

Point to Plane RANSAC

From each checkerboard detection, the following equation defines a constraint for the altimeter extrinsic parameters:

$$d(P_A, \pi_i) = 0 \tag{5.18}$$

where π_i represents the plane where the checkerboard pattern lies, and d represents the point to plane distance between P_A and π_i. This relationship is equivalent to the following expression:

$$\overline{O_i P_A} \cdot n_i = 0 \tag{5.19}$$

By explicitating $\overline{O_i P_A}$ as:

$$\overline{O_i P_A} = \begin{pmatrix} x_A - x_{Oi} \\ y_A - y_{Oi} \\ z_A - z_{Oi} \end{pmatrix} \tag{5.20}$$

and substituting eq. 5.18 and eq. 5.20 in eq. 5.19, the following relationship is obtained:

$$\begin{aligned} 0 \quad = \quad & x_A(n_{xi}) + y_A(n_{yi}) + z_A(n_{zi}) + \\ & cos(\alpha)(\rho n_{xi}) + sin(\beta)(\rho n_{yi}) + sin(\alpha)(\rho n_{zi}) - \\ & (x_{Oi} n_{xi} + y_{Oi} n_{yi} + z_{Oi} n_{zi}) \end{aligned} \tag{5.21}$$

For each calibration target, and by assigning:

$$\begin{aligned} \phi &= cos(\alpha) \\ \theta &= sin(\alpha) \\ \psi &= sin(\beta) \end{aligned} \tag{5.22}$$

the set of each point-plane constraint form a linear system of equations which can be expressed in matricial form as $Y = AX$ where:

$$X = \begin{pmatrix} X_A \\ Y_A \\ Z_A \\ \phi \\ \theta \\ \psi \end{pmatrix} \tag{5.23}$$

$$\begin{aligned} Y_i &= x_{Oi}n_{xi} + y_{Oi}n_{yi} + z_{Oi}n_{zi} \\ A_i &= [n_{xi} \quad n_{yi} \quad n_{zi} \quad \rho n_{xi} \quad \rho n_{yi} \quad \rho n_{zi}] \end{aligned} \tag{5.24}$$

Assuming that the rows of A define a set of linearly independent equations, the extrinsic parameters of the camera-altimeter setup can be obtained by solving this problem for X, which is done by pre-multiplying with the pseudo-inverse of A:

$$X = (A^T A)^{-1} A^T Y \tag{5.25}$$

Non-linear Refinement

The previous section defines a closed-form solution for computing the extrinsic parameters of a lidar altimeter-camera system. The extrinsics are here refined by solving a non-linear least squares problem using the previous results as starting points. For the linear least squares problem the distance to be minimized was the orthogonal distance between each couple of points and planes. By instead considering the along-line distance between each altimeter measure and ray intersection with the planes, errors are accentuated when evaluating distances with the most inclined planes. This error measure should lead to more accurate and representative constraints. Let $\varepsilon_{P\pi}$ represent this new error measure. From

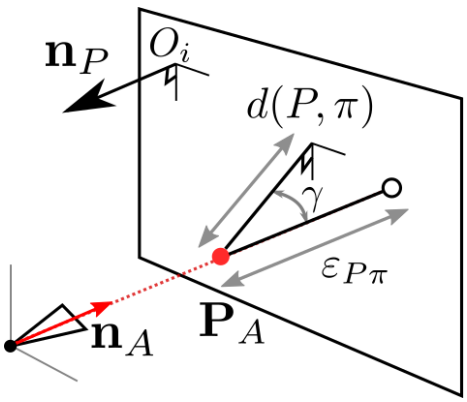


Figure 5.10: Representation of the residuals used in the closed form solution (orthogonal point-plane distance) and in the non-linear optimization (along-ray distance)

fig. 5.10, the error term $\varepsilon_{P\pi}$ can be obtained from $d(P,\pi)$ considering that:

$$\begin{aligned} d(P,\pi) &= \varepsilon_{P\pi} * cos(\gamma) \\ d(P,\pi) &= \varepsilon_{P\pi} * \frac{n_A \cdot n_P}{|n_A||n_P|} \end{aligned} \tag{5.26}$$

If n_A and n_P are unit vectors, the new error term is defined then as:

$$\varepsilon_{P\pi} = \frac{d(P,\pi)}{n_A \cdot n_P} \tag{5.27}$$

n_A encodes the direction of the laser beam and it is defined in the camera reference frame from the extrinsic parameters $\{\phi, \theta, \psi\}$. Using the Levenberg-Marquardt algorithm, the following problem is solved for the optimal extrinsics e:

$$argmin_e \sum_{i=1}^{N} (\varepsilon_{P\pi}(e)) \qquad i = 1 : N_{planes} \tag{5.28}$$

5.5.2 Tests

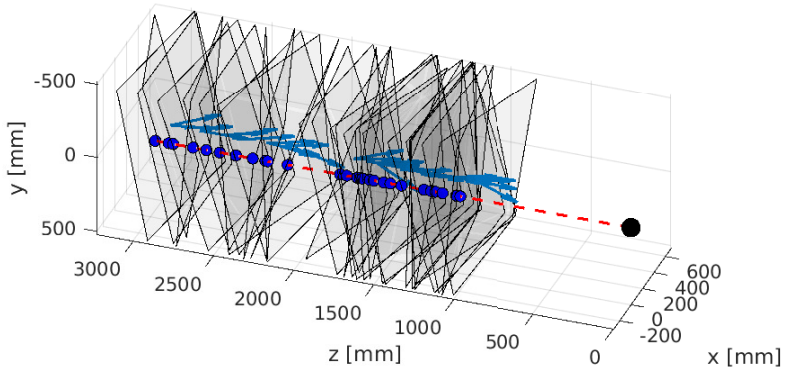
The calibration procedure is tested in two stages: first it is validated on a synthetic benchmark. That is done in order to exhamine the calibration accuracy while varying free parameters such as the number of calibration samples as well as their range and orientation with respect to the range sensor origin and view point. It is investigated also the effect of range measurement noise on both accuracy and repeatability of the calibration procedure. This step should both validate the approach and provide a guideline for a better calibration of the sensor setup. Secondly, a real-world test is performed using a UEye U-1226 monocular camera with resolution 640x480 and a Lightware SF-10B LiDAR Altimeter.

Synthetic Dataset

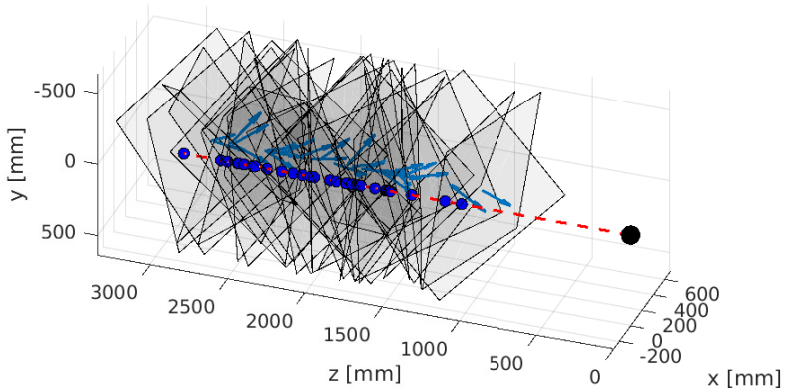
A synthetic benchmark is performed by generating a set of calibration samples with known orientation and position. A calibration sample is here defined as a pair defined as follows:

$$C_i = \langle T, \rho \rangle$$

where T is the transformation matrix from the camera reference frame to a virtual checkerboard reference frame and ρ is the measured range, affected by noise. T is defined such that the z direction of the local reference frame is normal to the plane, while the directions x and y are both contained in the plane. The transformation matrix T is derived by defining first a normal unit vector and an origin. Two angles defining the orientation of the normals are sampled from uniform distributions of angles of given limits. The plane origins are instead genereated with fixed x and y coordinates (in the camera reference system) and with z coordinates sampled from an uniform distribution between 1 and 3 meters, which are typical viewing distances of calibration checkerboards to a camera. In figure 5.11 are represented two examples of synthetic datasets generated as such. The virtual altimeter



(a) Angles in range (-20; 20) degrees



(b) Angles in range (-60; 60) degrees

Figure 5.11: Examples of synthetic datasets generated according to the procedure in section 5.5.2

extrinsics are here chosen to be:

$$\{X_A, Y_A, Z_A, D_X, D_Y, D_Z\} = \{0, 0, 0, 0, 0, 1\}$$

so that it is located coincident to the camera optical center and parallel to the z camera coordinate. For each plane, the coordinate of each intersection point with the range sensor view line are determined, and Gaussian noise is added to simulate measurement noise. For each test configuration (max angle, number of planes, measurement noise), datasets are sampled 100 times and each of them is used for the calibration procedure. This allows to capture the randomness of the results that can be expected during calibration session.
The calibration accuracy is here investigated by evaluating the distance of the sensor origin with respect to the ground truth. If $O_A = \{X_A, Y_A, Z_A\}$, the error metric is the L2 norm of O_A since the true origin is set to $\bar{0}$:

$$\varepsilon_O = \sqrt{X_A^2 + Y_A^2 + Z_A^2}$$

Regarding the orientation of the range sensor, it is evaluated the angle between the unit vectors representing the measured orientation and true orientation. If $\hat{n_A}$ is the computed orientation and $\hat{n_T}$ is the truth, then the angular error metric is defined as:

$$\varepsilon_\alpha = \arccos \frac{\hat{n_A} \cdot \hat{n_T}}{|\hat{n_A}||\hat{n_T}|}$$

Influence of plane orientation

The first test to characterize the performances of this calibration pipeline is related to the angles of the normal plane vector to the camera z axis, or the view axis. The orientation of normals impose multiple constraints over the orientation of the range sensor, these constraints are evident from the right half of the A matrix in the linear least squares problem. Well distributed orientations infact give full rank to the coefficient matrix. During this test, the number of planes is fixed to 30 and the sensor noise is modeled as zero-mean Gaussian noise with $10mm$ standard deviation. The free parameter is $\alpha_{max} = \beta_{max} = -\alpha_{min} = -\beta_{min}$ which is varied from 10 to 60 degrees. The choice is motivated by the fact that a minimum inclination is required to give full rank to the A matrix, and inclinations higher than 60 degrees might not allow checkerboard pattern detection algorithms to work accurately if at all. Figure 5.12 shows the convergence properties of the calibration procedure fixing the number of observed planes to 30 and varying just the maximum orientation angle of the calibration planes. Here the sensor noise is set to $10mm$ and each test configuration is repeated 100 times.
Figure 5.12b highlights the dependency of the origin estimation error from the plane orientation. The calibration suffers significantly from an excessive parallelism of the calibration planes showing mean position errors of nearly 100 mm in average for 10 degrees of maximum inclination. The accuracy however quickly increases with more inclined planes reaching a mean error of less than 2 cm in this test. The orientation error shows the same tendency, however it appears to be less heavily affected by lower inclinations, since for a maximum

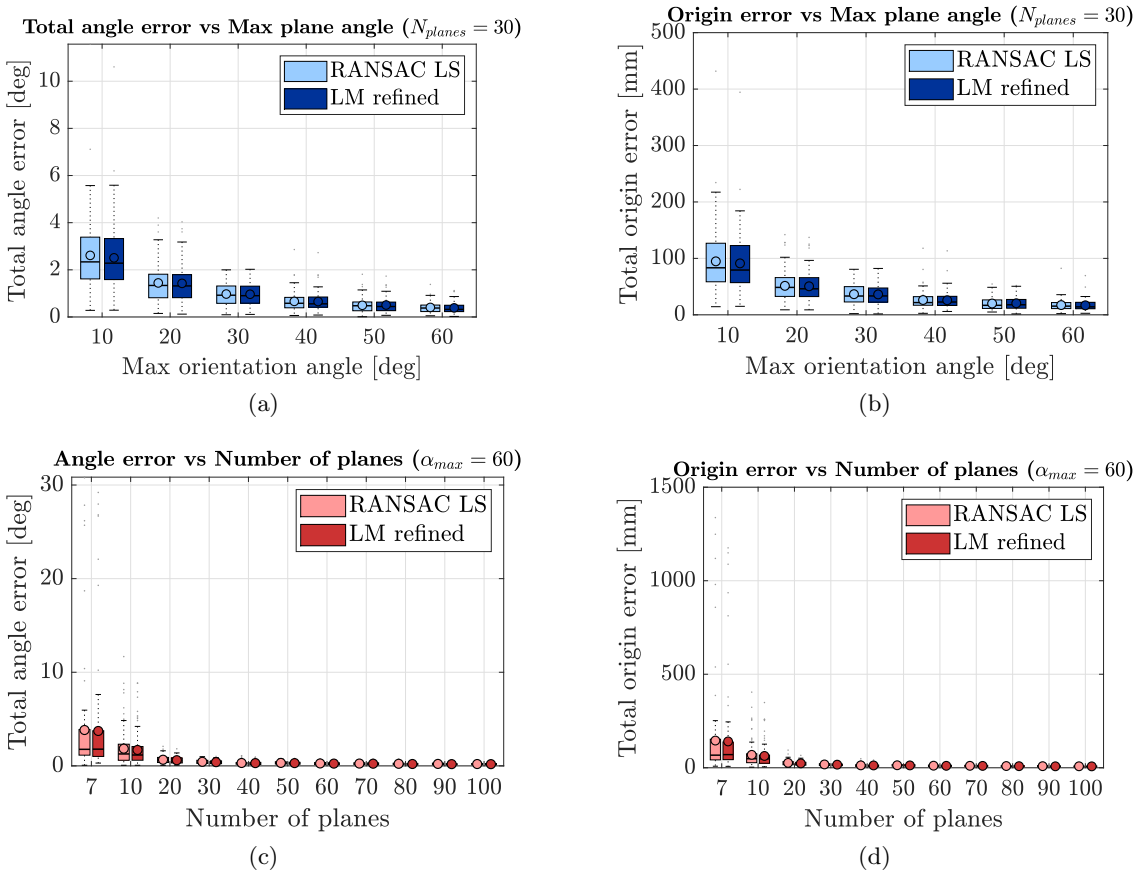


Figure 5.12: Synthetic benchmark test: (a) and (b) represent the convergence behavior of the calibration procedure while varying the maximum plane orientation in the uniform distribution. (c) and (d) represent the convergence behavior while varying the number of calibration planes

angle of 10 degrees, the mean angular error in 100 tries is just over 2 degrees. For a maximum inclination of 60 degrees, the angular error reaches an average of 0.39 degrees.

Influence of plane number

It is tested also the dependency of the calibration accuracy on the number of calibration samples. The sequence of tests here involves number of planes from the following set: $\{7, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. The lowest number of calibration samples is 6, since that is the number of free parameters to estimate in the linear system. The full rank of the A matrix is also guaranteed by sampling orientations and distance of planes from uniform distributions. However, more observations impose additional constraints to the extrinsic parameters and therefore lead to a more accurate estimation. Figure 5.12c depicts the trend of angular errors increasing the number of calibration planes, as expected the error rapidly decreases in the range between 7 and 30 planes and shows lower accuracy gains for more observations. For 100 planes, the mean angular error observed is 0.18 degrees but the results can be considered valid for a number of 30 planes which lead to the results in the previous section. The origin error also shows a quick convergence in the same range

Product Name	Datasheet Accuracy [mm]
AccuRange AR4000	±2.5mm
LightWare LW20	±10mm
LIDAR-Lite 3	±25mm
LightWare SF-10B	±50mm
LeddarTech LeddarOne	±50mm

Table 5.1: Examples of off-the-shelf LiDAR altimeters measurement noise

of observations, leading to lower increases for more observations. The mean errors for 30 and 100 plane observations are respectively 16.9 mm and 7.6 mm. Comparing these results to the sensor noise set to 10mm, the calibration can be considered successfull even for 30 observations.

Influence of measurement noise

Measurement noise has a significant effect on the calibration accuracy of the range sensor. Increasing the number of calibration planes serves to minimize the impact of noisy data by overcostraining the target calibration parameters. In table 5.1, multiple commercial single-point LiDAR rangefinders are reported with their declared range detection accuracy. Since no additional information is available, the reported accuracy can be interpreted as a zero-mean Gaussian with given σ. In this section it is tested the effect of noise over the calibration accuracy. All tests are performed by varying the σ from the set of following samples:

$$\sigma \in \{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30\} \quad mm$$

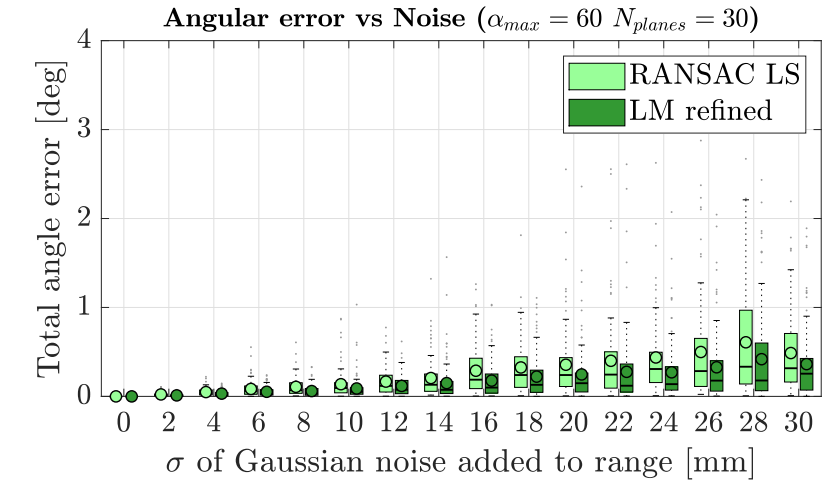
such that the generated ranges are computed as:

$$\rho_{test} = \rho + N(0, \sigma)$$

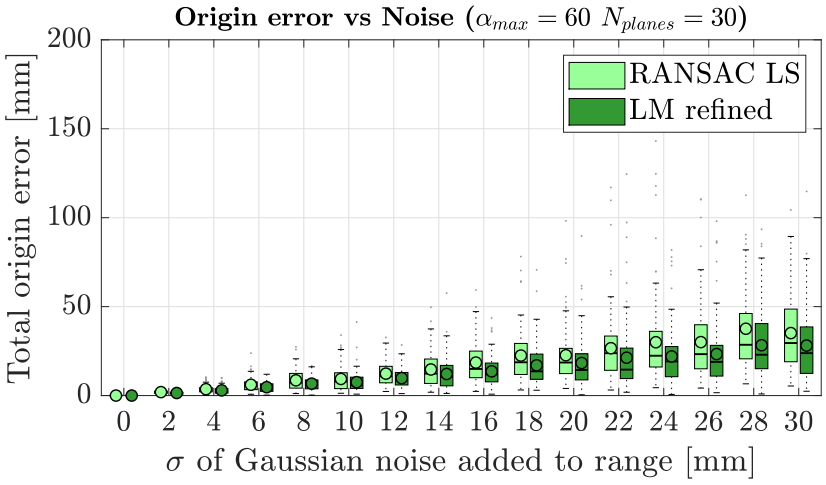
As in the previous sections, each synthetic dataset is randomly generated 100 times to capture the sparsity of the problem. Figure 5.13 represents the influence of range noise to the calibration parameters, as expected all errors increase with larger range noise. This test, however capture the beneficial effect that the non-linear optimization step has on refining the results. The percentual performance gain for all test cases is reported in figure 5.14, showing a particular influence of LM optimization in refining the orientation angle. The mean gain is estimated at around 50% for the orientation and 30% for the origin location.

Camera-Altimeter Setup

In this section, the calibration algorithm is used in a real-world test of a setup comprising a monocular camera and a lidar altimeter. The camera is an IDS uEye L-1226 color camera with a resolution of 752 x 480 pixels. The lens adopted is a wide-angle optic with a focal length of 2.5 mm. The LiDAR altimeter is a LightWare SF-10B LiDAR altimeter (see figure. 5.15), which has a range resolution of 1 cm and a maximum detectable range of 50 m. Both the camera and the altimeter are processed using the ROS (Robot Operating

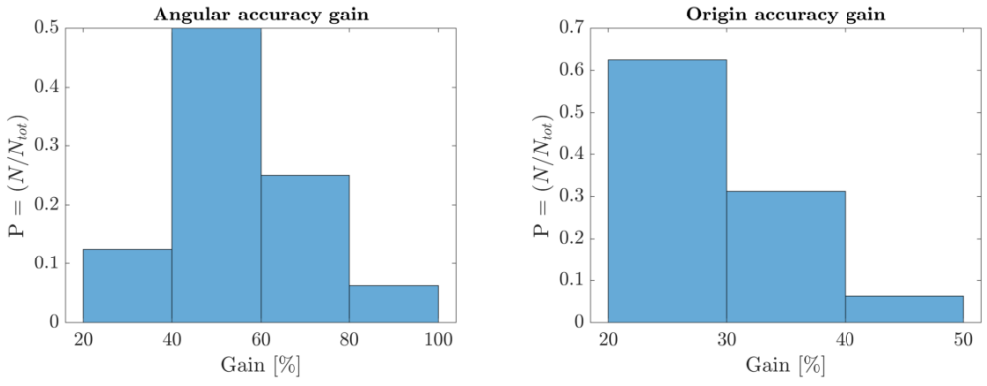


(a)



(b)

Figure 5.13: Orientation and origin errors as a function of added Gaussian noise to the synthetic ranges. Errors are reported before and after non-linear optimization. The number of planes is set to 30 and the maximum angle is set to 60 degrees.



(a) (b)

Figure 5.14: Histograms of percentual performance gain for the non linear optimization step. These results are related to the tests of fig. 5.13

System) library, images and ranges are timestamped according to the system clock and synchronized. The monocular camera is firstly calibrated using Zhang's method, so that its intrinsic parameters are known. The calibration procedure proceeds by observing a calibration target which is composed of a planar target with a checkerboard applied on the center. The checkerboard is printed on a A3 paper sheet and the square side dimension is 38.4mm.

The calibration proceeds as described in the previous sections, firstly by estimating an initial value and then by refining it via a non-linear least squares optimization. In figure 5.16 the calibration results are graphically represented. The representation is camera-centric therefore the camera lies at the origin of the coordinate system and the z-axis is the line of sight. The translucent squares represents the calibration targets as observed by the camera, please note that they do not reflect the entire dimensions of the target plane but only the checkerboard pattern. The altimeter origin is represented by a red circle just below the camera center and the altimeter measurement direction is the dashed red line. Blue markers denote the 3D location of the altimetric measures and they should lie close to the intersection between the planes and the red line.

As a "ground truth" for the extrinsic parameters between the camera and the altimeter is not available, the calibration quality can be determined firstly by visually comparing the graphical representation in figure 5.16, then by evaluating the distribution of the optimization residuals and finally by examining convergence properties.

Residual distribution

Examining the residuals distribution gives valuable information regarding the termination of the calibration procedure. Residuals represent in fact the point-plane distance as computed in eq. 5.27 which is going to be minimized, convergence of the non-linear optimization should therefore result in little residuals. Figure 5.17 is an histogram of residuals whose number is the inlier count after the RANSAC optimization scheme. The distribution of residuals has its maximum close to zero and most of the residuals are lower than 10mm which is the resolution of the sensor. For this particular calibration run, the number of inliers was 58, of which 43 below 10mm and 15 higher than 10mm. Since the 74% of the residuals is lower than the sensor resolution, the calibration can be considered successfull.

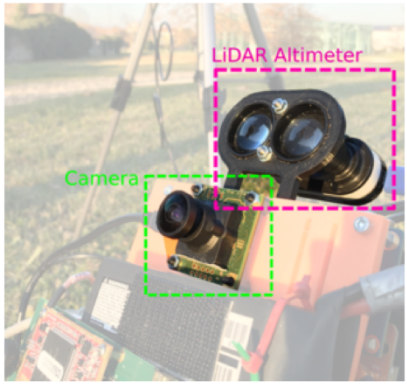
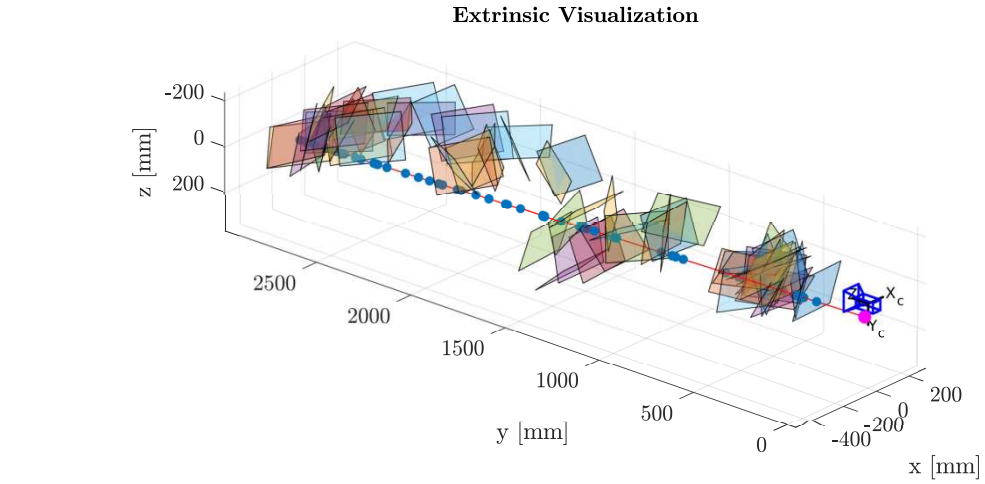
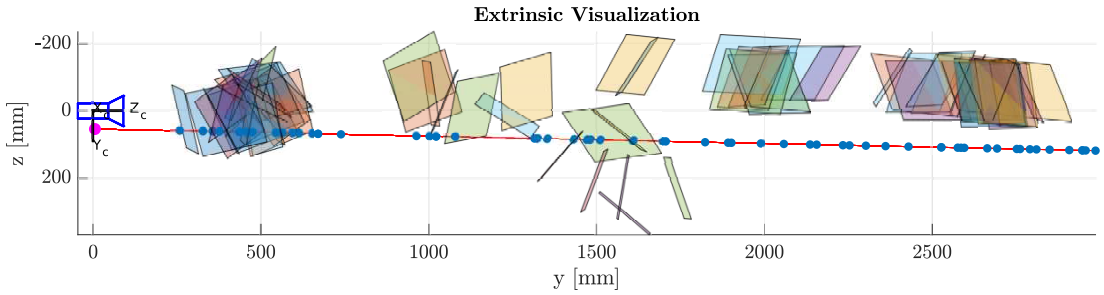


Figure 5.15: Setup of a monocular RGB camera and a LiDAR altimeter



(a) 3D view



(b) Side View

Figure 5.16: Calibration planes and altimeter extrinsics representation. The plot is in camera coordinates, z axis is coincinent with optical axis, x points to the right and y points downwards. The altimeter line of sight is represented as a red dashed line. A red dot represents the altimeter optical center and blue dots are 3D points of beam reflection

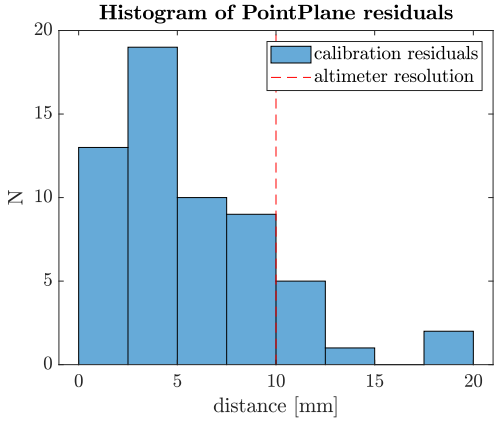


Figure 5.17: Histogram of residuals for the calibration in section 5.5.2. Vertical dashed line is the altimeter resolution of 10 mm. The majority of residuals is lower than the resolution of the sensor, suggesting a correct behavior of the calibration procedure.

Convergence properties

The calibration procedure tries to constrain the 6 parameters which define the position and orientation of the altimeter with respect to the camera. For this reason a minimum

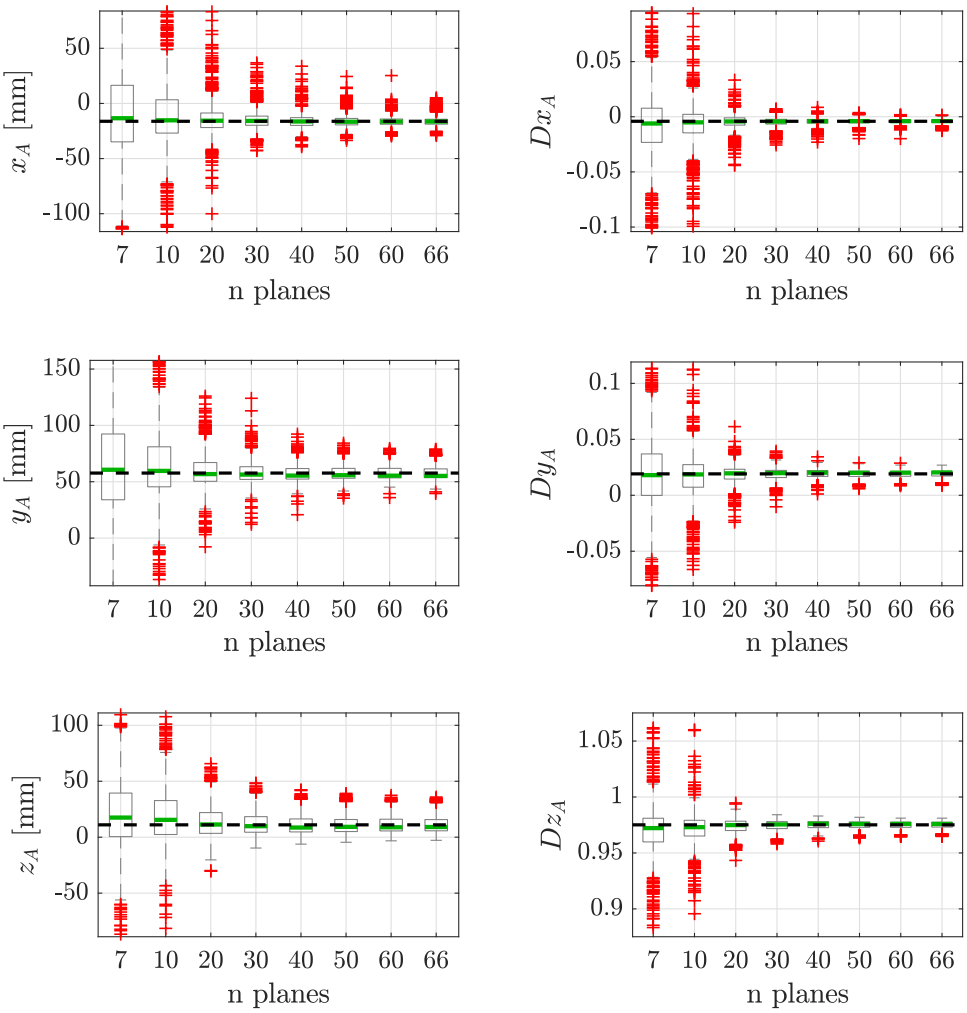


Figure 5.18: Convergence characteristics of the calibration algorithm. Each subfigure represents the calibration results for each of the 6 extrinsic parameters. Boxplots graphically denote the variability of results as a function of the number of planes used. The mean terminal value for each parameter is highlighed with a dashed line.

number of equations, or plane observations, must be satisfied. However, measurement noise both from the altimeter and the camera can propagate in the final estimate. A higher number of observations is then required to over-constrain the solution. It is evaluated here how varying the number of calibration planes propagates to the 6 extrinsic parameters estimation accuracy.

This test is conducted in a Monte Carlo fashion, where for each test number of planes N_p the algorithm is run 1000 times in order to capture the randomness of the RANSAC outlier rejection scheme. For each test also, N_p are sampled randomly from the total set of observations in order to capture the randomness of calibration target positions, which varies during different calibrations. Figure 5.18 represents the sparsity of results for this test highlighting in particular the effect of using a minimal amount of planes. For 7 planes, random configurations of calibration targets leads to errors of the order of meters on determining the altimeter origin. It can be seen that from 40 planes and above, the calibration procedure reaches the minimum variance of the results.

Confidence of the results

After estimating the extrinsic parameters between the camera and the altimeter it is useful to know the confidence interval associated to them. Following the non-linear calibration procedure, the confidence intervals are computed as a function of the residual magnitude and value of the Jacobian [3]. For the Monte Carlo test using the maximum number of planes, Gaussian probability distributions are fitted on the residuals after 1000 iterations. In table 5.2 the estimated and computed standard deviations are compared. From this comparison emerges the fact that uncertainty estimated after non linear optimization is very consistent with the Monte Carlo test, which captures the real sparsity of the results. The calibration results can then be trusted both in term of parameter and confidence estimates. This is crucial when performing sensor fusion, where the uncertainty estimates must be consistent with the real measurement uncertainties.

5.6 Results and Discussion

The aVO pipeline is evaluated on different datasets with ground truth information in order to provide information about the accuracy on reconstructing the camera trajectory,

[3] https://it.mathworks.com/help/stats/nlparci.html

Extr. param	Predicted	MC
x	6.847	4.598
y	7.638	6.391
z	5.873	7.677
D_x	0.003	0.002
D_y	0.005	0.003
D_z	0.003	0.003

Table 5.2: Comparison between the standard deviation on the solution estimated after the nonlinear optimization and from the Monte Carlo test in figure 5.19

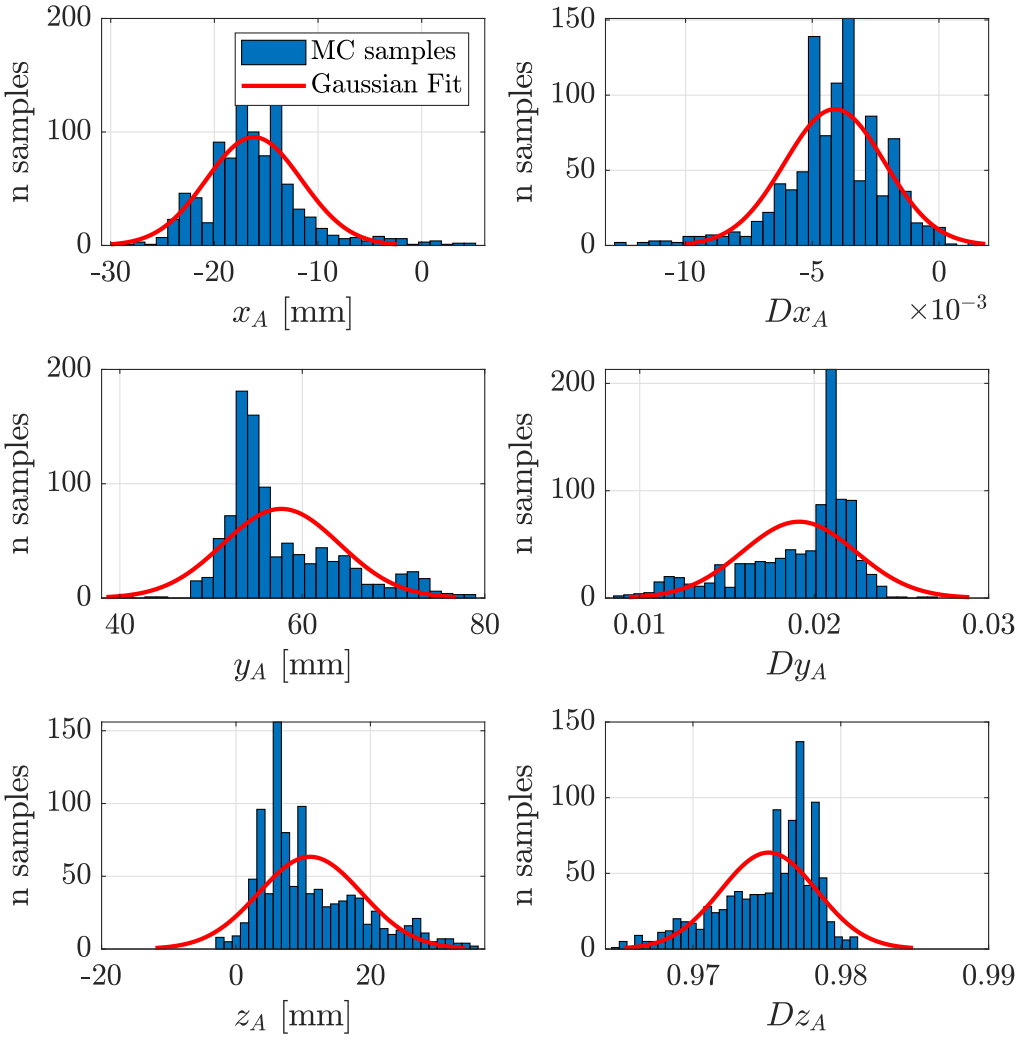


Figure 5.19: Monte Carlo test to capture the variability of results. The number of test is 1000. Normal distributions are fitted on the parameter distributions to validate the uncertainty estimations from the non-linear stage outputs.

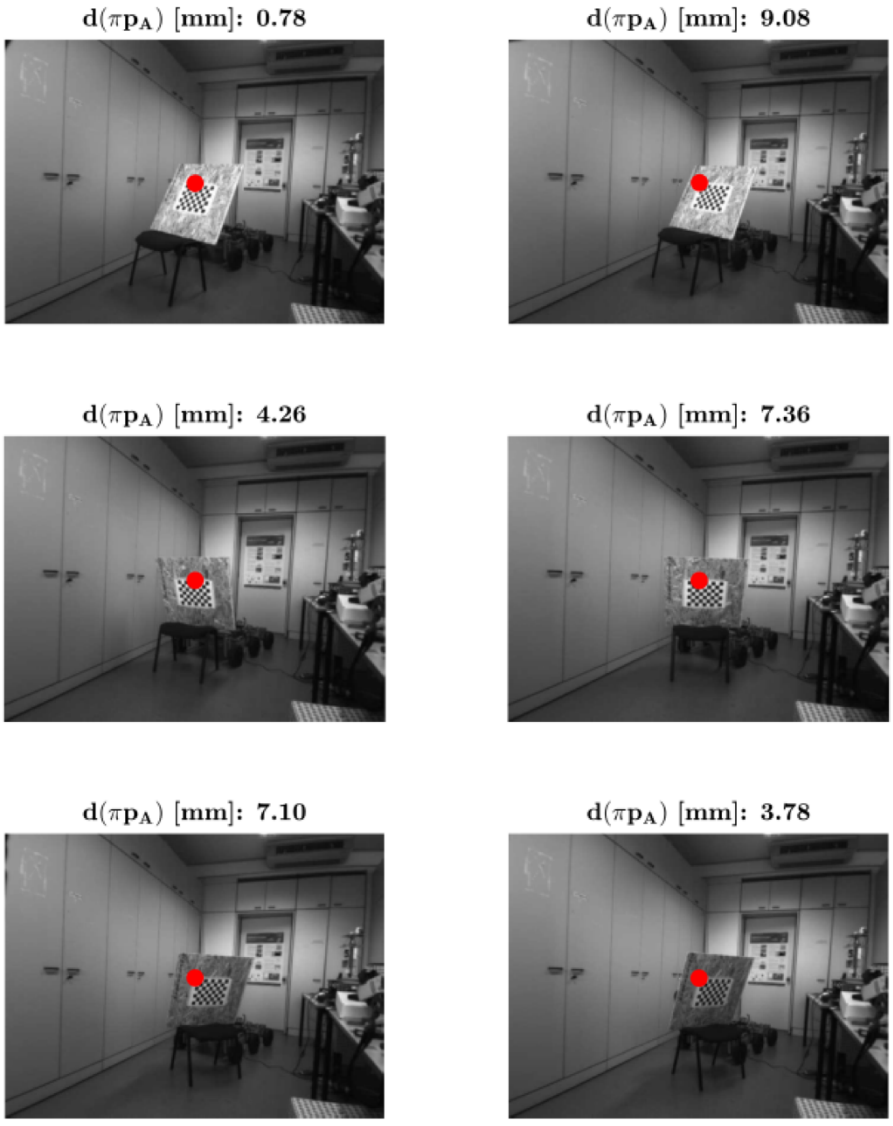


Figure 5.20: Overlays of altimeter detection point to the camera images. Sub-figure titles report the respective optimization residuals. All residuals of this example picture are lower than the altimeter resolution

which is directly related to the ability of this algorithm to retrieve a global scale from sparse range data. In addition, computational time is investigated to demonstrate real-time performances, or the capability of process each frame of an incoming sensor stream. aVO is tested as integrated in the ROS (Robot Operating System) environment[4], which allows easy integration with heterogeneous types of sensors as well as robotic platforms if needed. Testing the algorithm as a ROS *package* requires to stream collected data (images and range measures) in real-time by playback of a *rosbag* file. Each sensor message is timestamped and fed to each listening package as it is published by the actual sensor. This way, playing the algorithm *online* or *offline* does not imply any performance difference and allows for an accurate characterization. Figure 5.21 represents as a graph the ROS architecture during usage of aVO. The Visual Odometry node receives range data and images, as well as D-GPS information for logging, and publishes a transformation between the camera and world reference frames. In addition, the full trajectory and map are also available for visualization and further usage. All tests are performed on a standard consumer laptop equipped with a dual core Intel i5-3230 CPU (2.6 GHz nominal clock) and 8 Gigabytes of RAM memory, therefore timings are comparable, even if slightly higher, to embedded platforms such as the nVidia Jetson TX2.

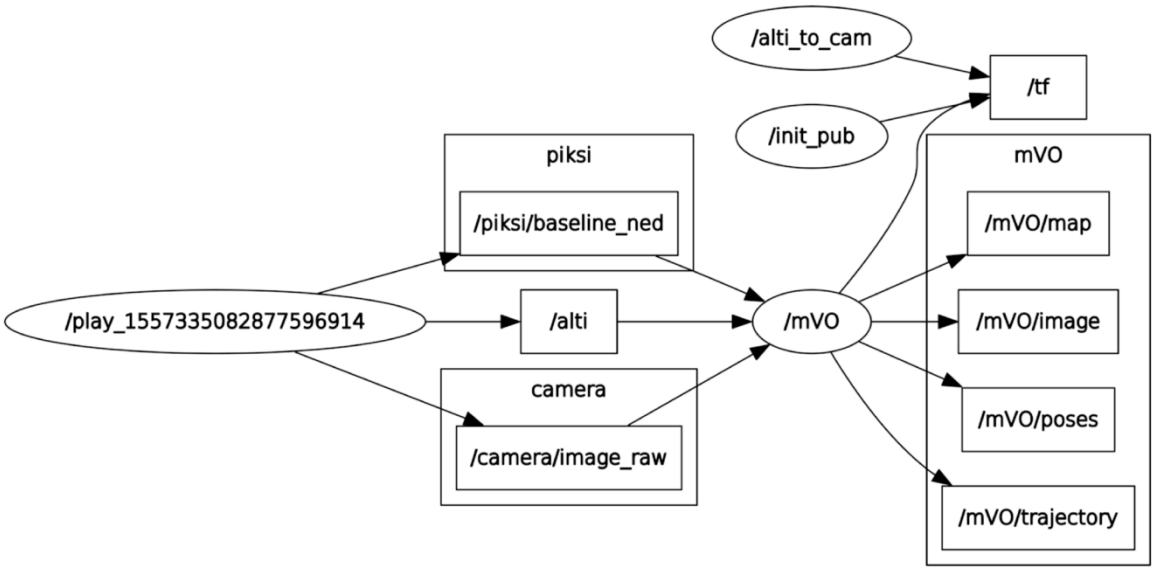


Figure 5.21: Graph of the ROS architecture for aVO. Round containers represents *nodes*, or applications, square containers represent *topics*, or message streams. The node to the left is the playback of the rosbag file feeding images and range measurements to the aVO node. The visual odometry outputs a transformation for the camera reference frame with respect to the world as well as trajectory and map information.

Three datasets are used for evaluating the performances of aVO which include different sensor types and observed scenarios. The first is a common dataset for evaluating Visual Odometry performances, the TUM Dataset for RGB-D (Red Green Blue and Depth) cameras [135]. This dataset provides images and depths captured by a Microsoft Kinect camera in indoor environments and provides an accurate ground truth from a Vicon tracking sys-

[4]https://www.ros.orgs

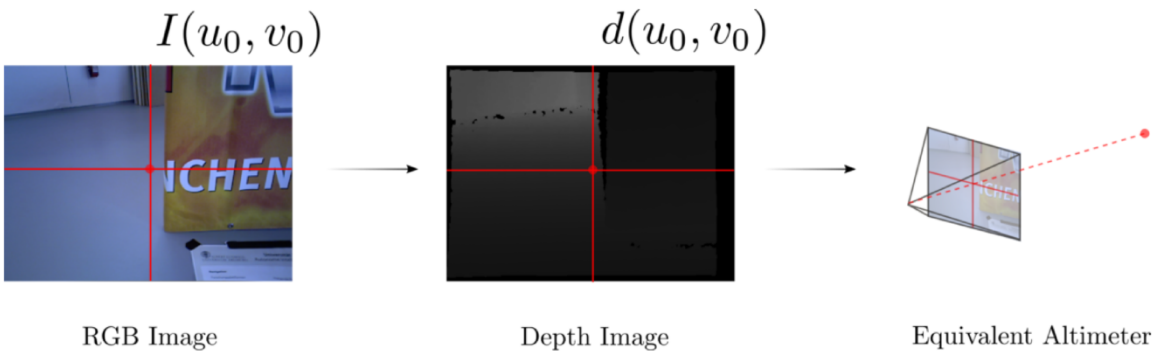


Figure 5.22: Preprocessing step for the TUM RGB-D benchmark dataset. Depth images are used to extract a single range value, for simplicity from the central pixel $\{u_0, v_0\}$. This single depth value simulates range measurements from a LiDAR altimeter.

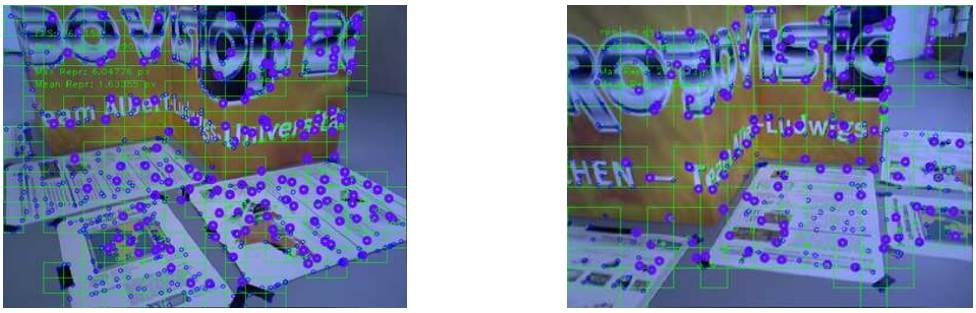
tem. The second dataset is captured in an outdoor scenario using a ZED Stereo Camera and a LightWare SF-10B altimeter. At the time of capturing this dataset no D-GPS was available therefore ground truth information is provided by aerial imagery. The last dataset comprises outdoor scenes captured using a monocular camera, an uEye 1226 with a wide angle lense, in conjunction with the LightWare altimeter and ground truth is given by a D-GPS (Piksi Multi). These datasets permit to evaluate the proposed Visual Odometry algorithm in a wide variety of scenarios and compare its performances to scale aware visual systems such as RGB-D or stereo cameras.

5.6.1 RGB-D Benchmark Tests

The TUM RGB-D benchmark [135] is a collection of indoor sequences captured with a Microsoft Kinect RGB-D sensor which produces RGB and Depth images at 640x480 pixels resolution at a frame rate of 30 Hz. The Kinect sensor is tracked by a motion capture system which the authors of [135] claim to deliver pose measurements with errors lower than 10 mm and orientations with errors lower than 0.5 degrees. As this dataset is well known in the SLAM community, evaluating our monocular pipeline one some of its sequences deliver performance information easier to compare with the VO/SLAM literature. In addition it permits to evaluate how well aVO recovers the global scale of the environment using just one range information per keyframe instead of $\sim 3 \cdot 10^5$ (640x480) depth values per image for full RGB-D SLAM systems. To adapt the dataset sequences to our system, virtual altimeter measurements are generated by extracting single range data from depth images. Figure 5.22 shows how the depth information related to the central image pixel is used to simulate an altimeter whose principal axis is coincident with the camera z axis.

Pose estimation accuracy is evaluated both in absolute terms as well as relative. An Absolute Trajectory Error is computed from L2 distances between the estimated pose and the ground truth. According to the notation in [77], being x_i^* the ground truth pose and x_i the estimations, a mean Absolute Trajectory Error is computed as follows:

$$\mu(\varepsilon) = \frac{1}{n} \sum_{i=1}^{n} ||x_i^* - x_i|| \tag{5.29}$$



(a) Example View (b) Example View

Figure 5.23: Example views from the dataset *fr3_structure_texture_far*

where x_i^* and x_i are coupled by the proximity of their timestamps. Additional error metrics consider also the amount of translational drift, or the error accumulated in the instantaneous direction of motion. If δ_i^* and δ_i are $||x_{i+1}^* - x_i^*||$ and $||x_{i+1} - x_i||$, the following metric estimates an absolute value for the average translational drift between consecutive poses:

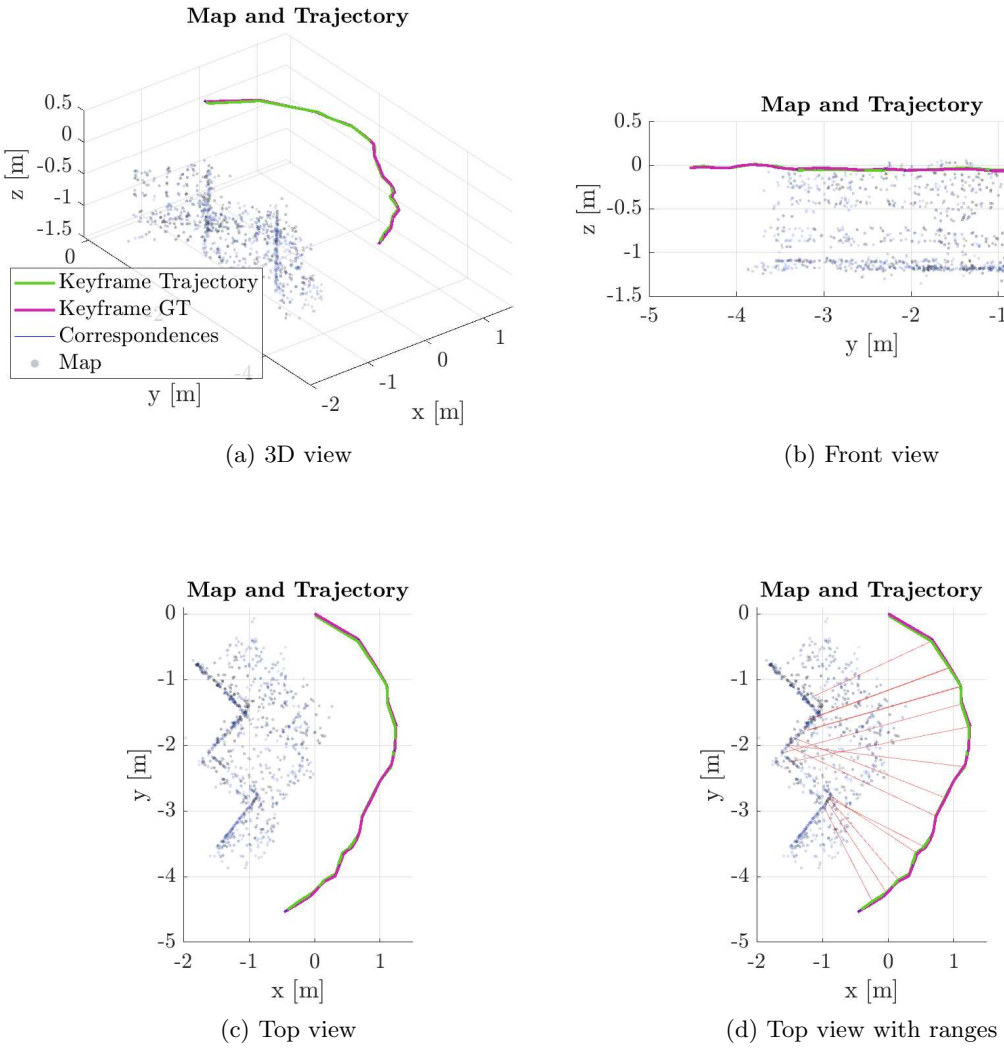
$$|\Delta| = \frac{1}{n}\sum_{i=1}^{n} ||\delta_i^* - \delta_i|| \tag{5.30}$$

A second metric evaluate instead a net value of the translational drift, as a trajectory can drift in random directions between each consecutive pose without accumulating significant errors over time:

$$\Delta = \frac{1}{n}\sum_{i=1}^{n} (\delta_i^* - \delta_i) \tag{5.31}$$

Fr3_structure_texture_far

The first test sequence is the *fr3_structure_texture_far* where the camera orbits around a structured environment providing sufficient amount of texture for feature detection and tracking. In this sequence, the camera is pointing slightly upwards observing both the ground as well as the scene, for a total length of 5.96 meters in length, see figure 5.23 for two example views from the sequence. For this reason, as the structure is not planar, initialization is performed by decomposition of the *essential* matrix. Figure 5.24 shows multiple views of the results from the aVO pipeline. It is represented both the map of the environment as well as the reconstructed camera trajectory compared with the motion capture system ground truth. From a qualitative point of view it can be appreciated how the pipeline can build accurate 3D maps of the observed environment, as suggested by the top views. It can also be observed how one of the flat vertical surfaces has a slight alias a few centimeters apart. This is due to the online optimization of the scale using altimeter measurements, a few points were observed and triangulated just before performing a Local Bundle Adjustment that detected and corrected a significant scale drift. In terms of errors, table 5.4 reports a comparison of the metrics introduced previously and computed over a series of 10 runs in order to capture all the random effects that occurs each time. As the implementation run real time it is affected by the task scheduling from the operating



(a) 3D view

(b) Front view

(c) Top view

(d) Top view with ranges

Figure 5.24: Views of trajectory and map for the sequence *fr3_structure_texture_far* as computed by the algorithm. Colored dots are 3D landmarks triangulated and optimized during the sequence, the green line is the estimated trajectory and in magenta the ground truth from the motion capture system. (d) represents also altimeter measurements from the set of accepted associations (see section 5.3) to highlight the recovery of a correct scale

		μATE	σATE	$\lvert\Delta\rvert$	Δ
Ours	$[m]$	0.048	0.023	0.016	0.003
	[%]	0.815	0.381	0.270	0.049
- (no alti)	$[m]$	0.088	0.041	0.016	0.006
	[%]	1.514	0.823	0.280	0.097
ORB-SLAM2	$[m]$	0.027	0.011	0.017	0.001
	[%]	0.438	0.176	0.278	0.009
RTAB-MAP	$[m]$	0.029	0.012	0.019	0.001
	[%]	0.498	0.197	0.330	0.013
RGBD-SLAM	$[m]$	0.051	0.038	0.022	0.002
	[%]	0.891	0.652	0.385	0.042

Table 5.3: Performance comparison with RGB-D algorithms

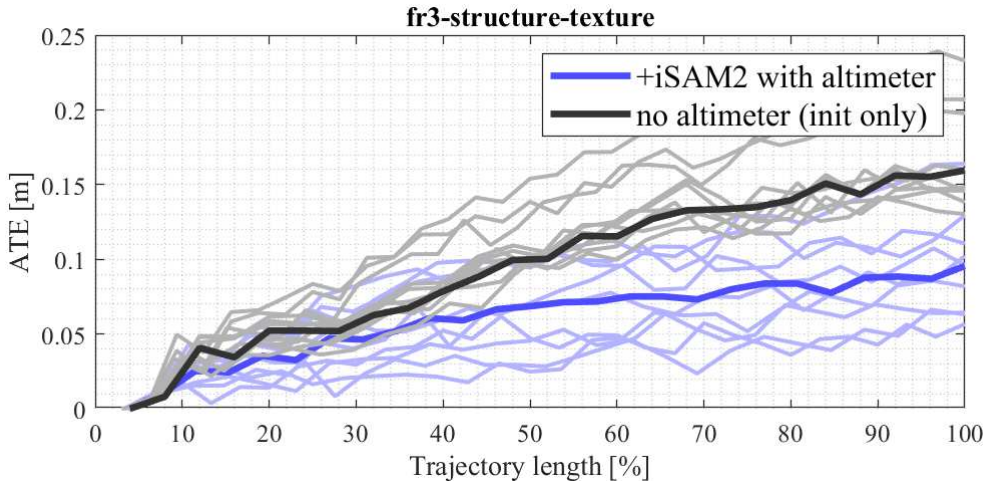
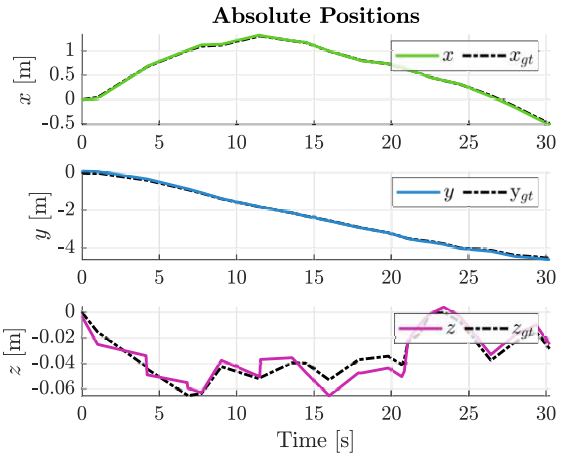
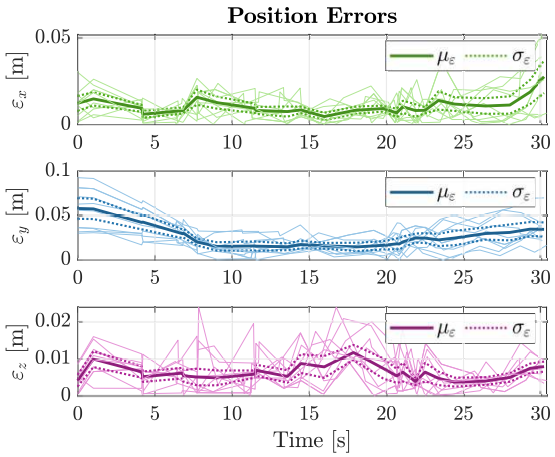


Figure 5.25: Absolute Trajectory Error for 10 runs of the visual odometry on the *fr3_structure_texture_far* sequence both with and without range inclusion in the Bundle Adjustment step. Bold lines represent the mean error over the 10 runs.

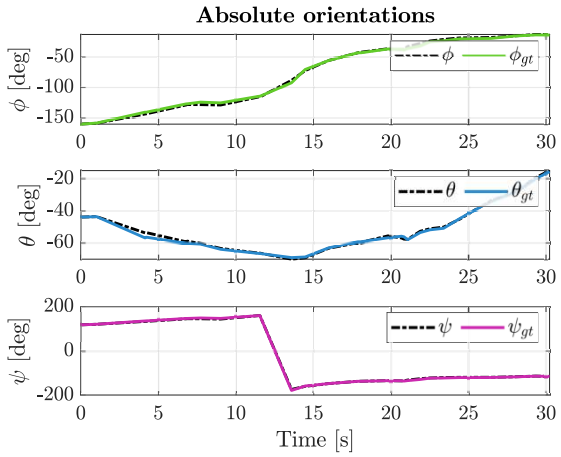
system, which have a significant impact on the multi-threaded architecture of the visual odometry pipeline. For this reason, both a mean ATE and a standard deviations over all runs are reported. The algorithm is tested against ORB-SLAM2 [98] in RGB-D configuration, RTAB-MAP [79] and RGBD-SLAM [36], state of the art system using full RGB-D images to compute scale aware reconstructions. The effectiveness of the proposed range fusion approach is proven by the fact that our algorithm reaches very similar accuracies in pose estimation compared to all the other algorithms, even outperforming RGBD-SLAM. Overall, the relative Absolute Trajectory Error normalized over the trajectory length, is lower than 0.8% which is very limited even for scale aware RGB-D and stereo vision systems for any kind of trajectories and environments. Performances also in terms of translation drift are on par with the other tested algorithms. To stress the importance of embedding range information in the Bundle Adjustment, the pipeline is tested also including range only for the initialization step. The second row of table 5.4 reports the errors obtained without online scale correction, which are almost doubled. Figure 5.25 plots the Absolute Trajectory Error over the length of the sequence. Each light line is the ATE over the 10 runs averaged in table 5.4 and bold lines are the average errors. It is reported the effect of including range measurements in the Bundle Adjustment over just using them for initialization of the pipeline. The advantage of correcting scale online is evident over the course of the whole sequence. Figure 5.26 reports a detailed analysis of the translational and rotational errors as well as the absolute coordinates from keyframe poses. After aligning the estimated trajectory over the reference one from the motion capture system, translational errors are mostly in-plane, or distributed over the x and y directions which are parallel to the ground floor. The y coordinate exhibits a significant drift which can be likely caused by slight angular errors. Orientation errors are generally contained in the $[0, 5]$ degrees range while being in average in the order of 1 degree or less.



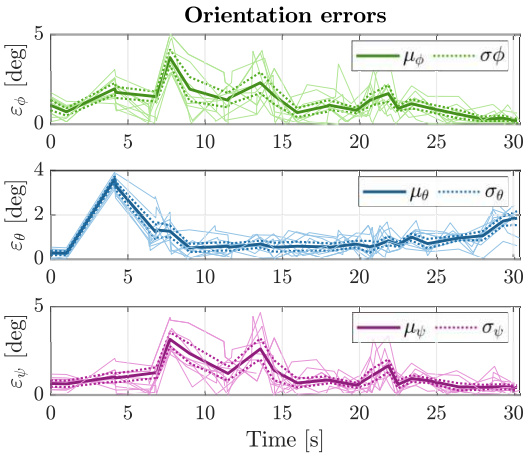
(a) Absolute Position



(b) Position Error

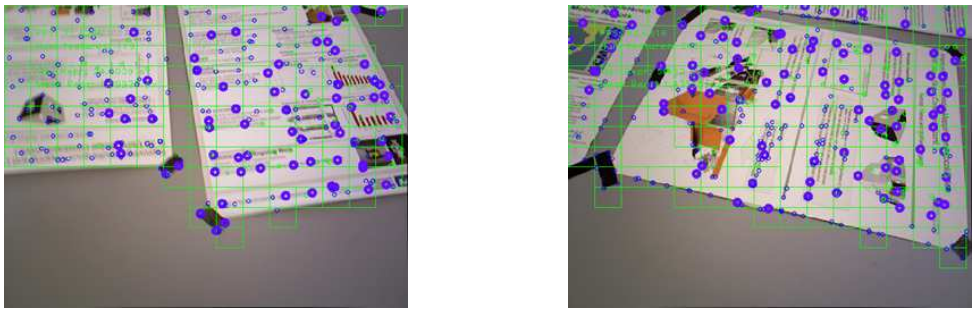


(c) Absolute Orientations



(d) Orientation Error

Figure 5.26: Poses, orientations and related errors for the *fr3_structure_texture_far* sequence. Errors report also for visual reference the maximum error reported by the dataset authors for the camera localization using the motion capture systems. Maximum position error is estimated to be 10mm and maximum orientation error to be 0.5 degrees. Note that the step in (c) for the ϕ angle is due to the conversion between quaternions and Euler angles.



(a) Example View (b) Example View

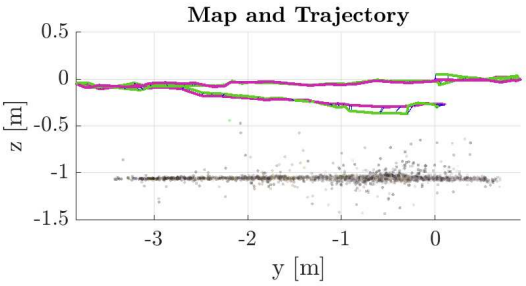
Figure 5.27: Example views from the dataset *fr3_nostructure_texture_near_withloop*

Fr3_nostructure_texture_near_withloop

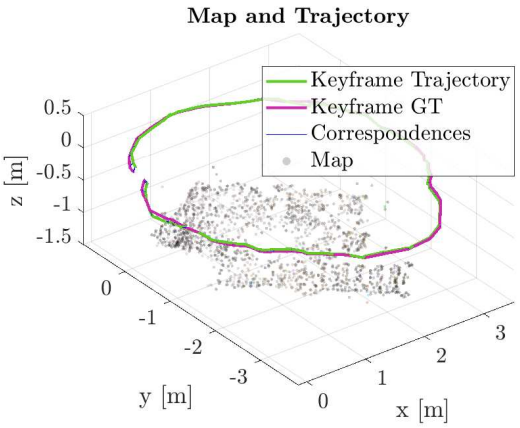
The second sequence in the TUM RGB-D dataset where the pipeline is tested is denoted *Fr3_nostructure_texture_near_withloop* and comprises a planar scene textured with posters to deliver some visual information for tracking. The camera moves in a circular motion looking mainly downwards in a trajectory of length 13.85 meters so that most of the keyframes can associate range information from the virtual altimeter to the observed landmarks. The trajectory ends near the same spot from which it started allowing for easier evaluations of pose drifts. As for the previous sequence, the image is divided in a 15x15 grid for feature extraction, the image feature uncertainty is set to $\sigma_{u,v} = 1$ pixel and the range uncertainty is set to $\sigma_{range} = 1$ cm which is a reasonable amount for an RGB-D sensor in indoor environments. Figure 5.27 shows two example views from the dataset. Note that in some images not all the field of view of the camera contains texture information. In many keyframes the virtual altimeter can not be associated to any 3D landmark. Figure 5.28 shows some views of the reconstructed trajectory and map for one run of the sequence. Compared to the previous sequence, this one can be sightly more difficult to process correctly as in some part the field of view of the camera contains very little texture information and localized in just one part of the image. This results in both poor constraints over the camera pose when solving equation 5.8 as well as scale drift as the virtual altimeter measurements are

		μATE	σATE	$\lvert\Delta\rvert$	Δ
Ours	$[m]$	0.064	0.044	0.023	0.001
	[%]	0.464	0.315	0.165	0.009
- (no alti)	$[m]$	0.103	0.041	0.022	0.001
	[%]	0.759	0.301	0.158	0.010
ORB-SLAM2	$[m]$	0.037	0.025	0.023	0.003
	[%]	0.278	0.190	0.170	0.022
RTAB-MAP	$[m]$	0.068	0.043	0.022	0.003
	[%]	0.507	0.323	0.160	0.019
RGBD-SLAM	$[m]$	0.050	0.035	0.016	0.001
	[%]	0.370	0.261	0.116	0.006

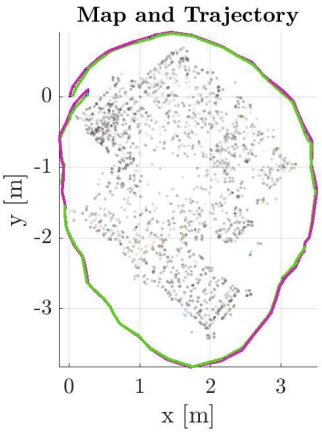
Table 5.4: Performance comparison with state of the art RGB-D algorithms, sequence *fr3_nostructure_texture_near_withloop*. ORB-SLAM2 is set in RGB-D mode



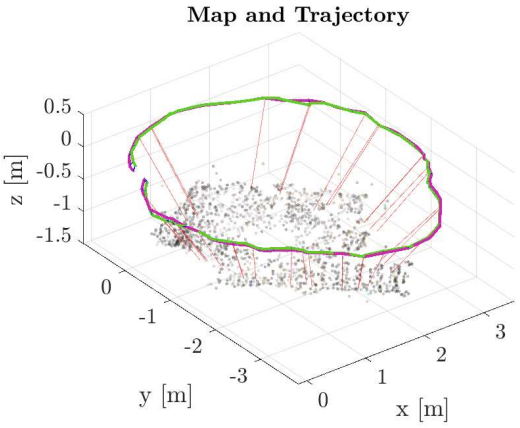
(a) Front view



(b) 3D view



(c) Top view



(d) Top view with ranges

Figure 5.28: Views of trajectory and map for the sequence *fr3_nostructure_texture_near_withloop* as computed by the algorithm. Colored dots are 3D landmarks triangulated and optimized during the sequence, the green line is the estimated trajectory and in magenta the ground truth from the motion capture system. (d) represents also altimeter measurements from the set of accepted associations (see section 5.3) to highlight the recovery of a correct scale

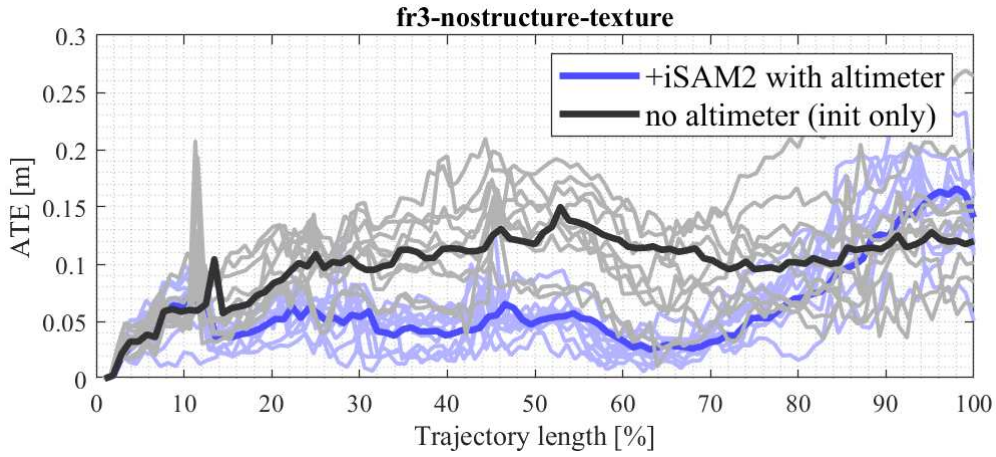


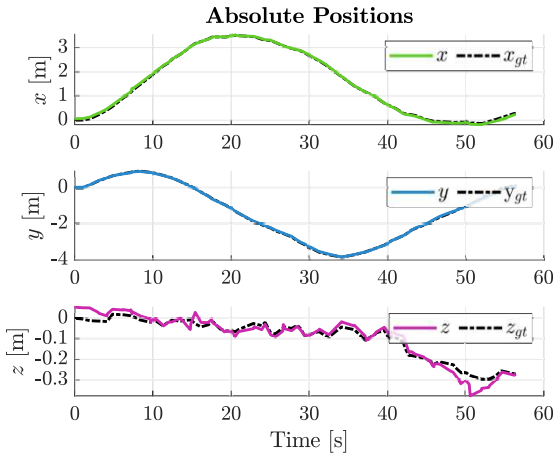
Figure 5.29: Absolute Trajectory Error for 10 runs of the visual odometry on the *fr3_nostructure_texture_near_withloop* sequence both with and without range inclusion in the Bundle Adjustment step. Bold lines represent the mean error over the 10 runs.

not associated to any 3D landmark. This effect is particularly visible in the reconstructions during the end of the sequence. Looking at the reconstruction in figure 5.28b, the trajectory evolves clockwise and ends a few centimeters below the starting point. All views highlight the pose drift that occurs near the end, where the camera observes a gap between the posters without texture. Eventually the scale is recovered in the last few keyframes where texture is present and scale information is inserted again in the Bundle Adjustment.

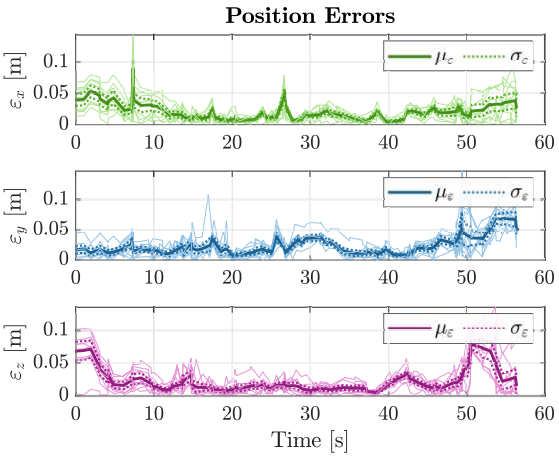
Table 5.4 reports the error metric computed as shown in the previous section, confirming the findings from the last test sequence. While ORB-SLAM2 in RGB-D configuration outperforms our pipeline, the performances are on par with other RGB-D SLAM systems even outperforming slightly RTAB-MAP in this scenario. Again, the effect of neglecting scale information in the Bundle Adjustment have a significant impact in pose estimation accuracy as demonstrated also by figure 5.29. Figure 5.30 reports a detailed view of all errors related to one run the algorithm. It is evident also here how pose errors concentrate on the last part of the trajectory when little visual information is present, and how the z error decreases in the very last part of the sequence as new range information can be exploited in the Bundle Adjustment. Similar considerations regarding angular errors can be made with respect to the previous test sequence in this dataset.

Timings in the RGB-D Dataset

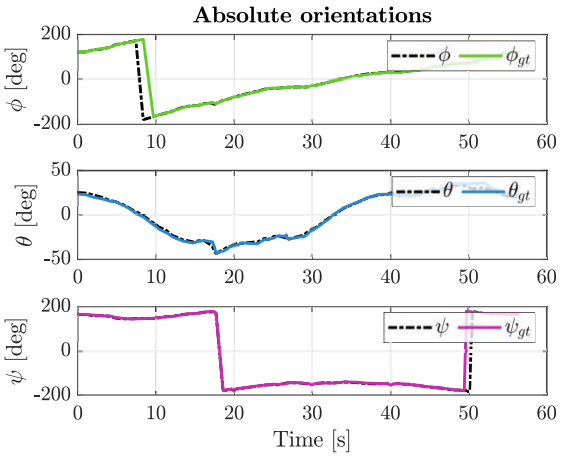
Figure 5.31 reports timings related to the front-end thread of the pipeline, which is devoted to compute the camera pose with respect to each new incoming frame and the map being built. The algorithm is designed to be able of obtaining reconstructions with accurate scales as well as keeping the computational time low so that no frames, or a limited number of them, are skipped. In both sequences the task of detecting FAST features arranged in a grid is the least time consuming, requiring in average from 1 ms to 3 ms. The feature tracking step requires in average 5 to 10 milliseconds in the worst cases because, depending on the motion, more features can concentrate in the same parts of the image, for example when



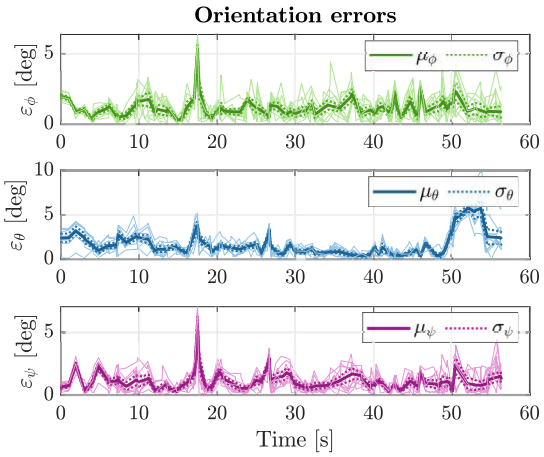
(a) Absolute Position



(b) Position Error

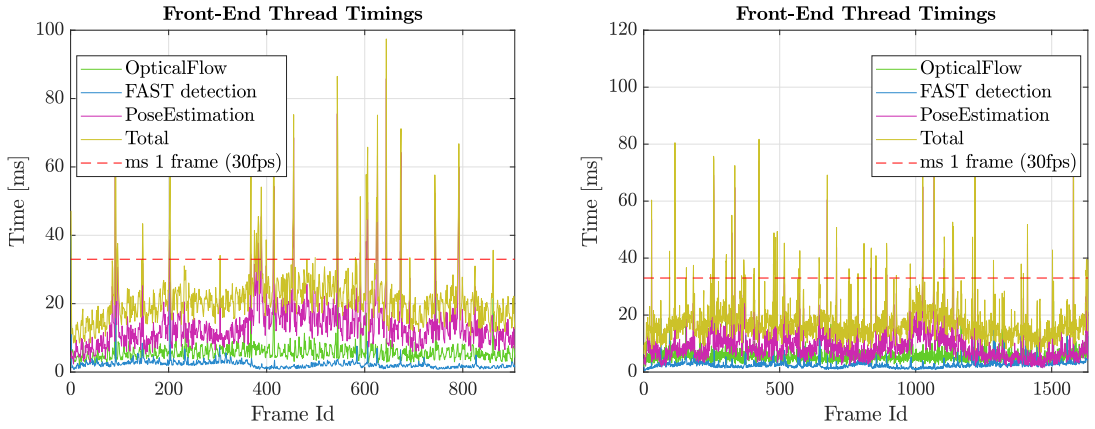


(c) Absolute Orientations



(d) Orientation Error

Figure 5.30: Poses, orientations and related errors for the *fr3_nostructure_texture_near_withloop* sequence. Errors report also for visual reference the maximum error reported by the dataset authors for the camera localization using the motion capture systems. Maximum position error is estimated to be 10mm and maximum orientation error to be 0.5 degrees. Note that the step in (c) for the ϕ angle is due to the conversion between quaternions and Euler angles.



(a) *fr3_structure_texture_far* (b) *fr3_nostructure_texture_near_withloop*

Figure 5.31: Timings in the TUM RGB-D dataset sequences

the camera orbits around an observed location. Further optimization can decrease this time by deciding to track only the best scoring landmarks in each cell of the image (shown as big blue dots in figures 5.23 and 5.27) as the one with lowest reprojection errors and longest life in terms of frames span of the track. Pose estimation is probably the most variable step as the computational time depends on how the optimization problem in section 5.2.4 is well conditioned. The time spent each frame in computing the pose can reach up to 20 milliseconds while in average requiring around 10 milliseconds or lower. Overall the total time spent for each frame is well below the nominal 33 milliseconds of a 30 Hz refresh rate video stream. Some spikes are visible causing a few frame drops and might be related to concurrency issues between threads or little optimization of the research code.

5.6.2 Outdoor Long Range Tests. Comparison with stereo SLAM

While the previous test was dedicated to showing the tracking performances of the algorithm in confined scenarios, in this part the visual odometry is tested on longer distances in order to highlight the stability of scale over time. The experimental system here comprises

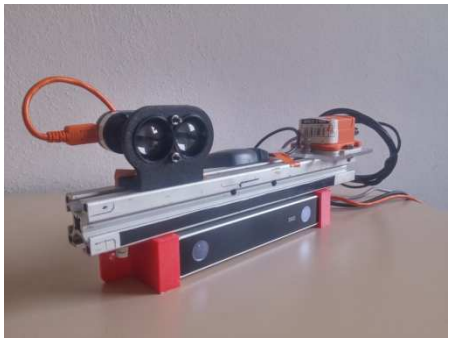


Figure 5.32: Experimental setup comprising for the long range tests: a Stereolabs ZED camera (left images used for monocular visual odometry), a LightWare SF-10B LiDAR altimeter and an Xsens MTi-G IMU and GPS (not used for ground truth)

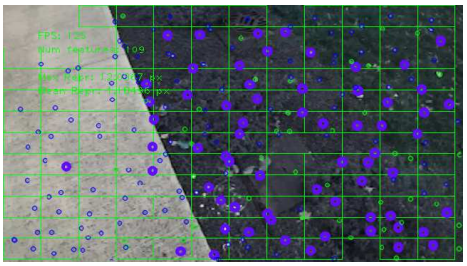
a stereo camera (Stereolabs ZED camera) and a LightWare SF-10B altimeter, which allows to test performances against a scale aware state of the art stereo visual SLAM algorithm, ORB-SLAM2. At the time of performing this experiment, no differential GPS was available, therefore ground truth is given by aerial imagery and precise knowledge of the starting and ending positions. The GPS measurements will be shown for reference but were too inaccurate for giving a precise reference track because, as stated from the manifacturer of the XSens MTi-G, at a CEP (Circular Error Probability) of 50% the reference value for the pose error is 2.5 meters. The stereo camera from the setup visible in figure 5.32 was set to record images at a resolution of 762x376 pixels with a framerate of 30 Hz. A virtual monocular camera and altimeter system was created by selecting only the left image from each stereo frame. As the GPS track was unusable for evaluation, Google Maps services were used instead to track accurately in a second moment the camera path knowing the starting and ending locations. As any temporal correlation was not available between the reference path and the camera estimated trajectory, path *length* was evaluated to highlight any translational drift due to scale inaccuracy. In addition, we used the Iterative Closest Point algorithm [113] to align the trajectories estimated by our algorithm and ORB-SLAM2 to the ground truth and used the residual RMSE error between couples of corresponding points to estimate how each trajectory fitted the reference one. In table 5.5 are reported the results for this test. ORB-SLAM2 set in stereo configuration is able to estimate a close trajectory in terms of path accuracy even if, as shown in figure 5.33 the aligned trajectories appears to be very similar. However, as far as the translational drift is concerned, the length for the trajectories estimated by our monocular pipeline and ORB-SLAM2 stereo show equal error in relation to the ground truth suggesting that scale drift is very minimal and well contained from our approach.

5.6.3 Outdoor Tests. Comparison with D-GPS

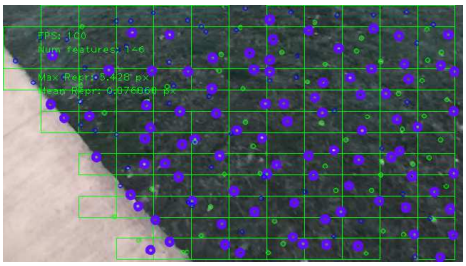
The last test sequences are captured with an hand-held sensor setup comprising a monocular camera, a LiDAR altimeter and a differential GPS for a precise ground truth with centimeter level accuracy. The camera is a IDS uEye 1226 with a resolution 752x480 pixels and a frame rate of 30 Hz. The camera is equipped with a wide angle lens of 2.1 mm focal length and is provided with a global shutter. The altimeter is a LightWare SF-10B and the D-GPS is a Swift Piksi Multi which in our test, at the best level of satellite coverage provided poses with respect to a base-station at a rate of 100Hz and an average uncertainty of around 2 cm both longitudinal and vertical. The setup is shown in figure 5.15. In these sequence of tests, the camera is moved always facing the ground as it would being mounted

		RMSE ICP	Traj. Length (% length error)
Ours	$[m]$	1.22	86.58
	[%]	1.42	1.03
ORB-SLAM2	$[m]$	1.07	84.69
	[%]	1.24	1.18

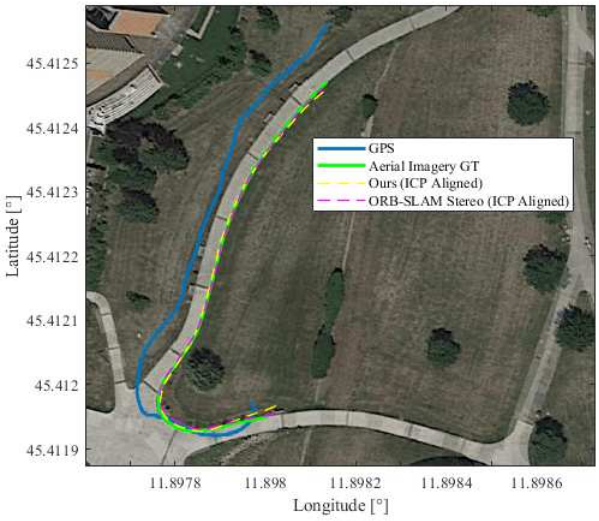
Table 5.5: Performance comparison with ORB-SLAM2 (STEREO)



(a) Example View 1



(b) Example View 2



(c) Results

Figure 5.33: (a) and (b) are example views from the outdoor dataset and (c) shows the estimated trajectories from the aVO pipeline and ORB-SLAM2 in stereo configuration. The green line is a manually annotated ground truth from aerial imagery. The blue line shows the very imprecise GPS track for visual reference

on a UAV for providing relative localization. In each sequence different kinds of motion are tested. In the first of them, labeled OUT_CIRCLE, the vision system is carried over a circular path which begins and ends approximately in the same position. Doing so it is evaluated both quantitatively and qualitatively how the estimated trajectory diverges from the correct path for a relatively long sequence. The second test, labeled OUT_HOVER, involves a hovering motion of the camera over a fixed spot and with moderate altitude changes, while the last sequence, labeled OUT_LANDING, shows the camera hovering for a while, then reaching the ground to simulate a landing motion and takes off again. Each sequence is tested with the metrics used for the RGB-D Dataset 5.6.1, however this time it is not made any comparison with existing state of the art algorithms as the used setup provides already an accurate ground truth for reference and no available scale-aware visual algorithm can be used with just a monocular camera.

		μATE	σATE	$\|\Delta\|$	Δ
Ours	$[m]$	0.082	0.033	0.052	0.004
	[%]	0.481	0.189	0.296	0.023

Table 5.6: Error metrics for the OUT_CIRCLE test sequence

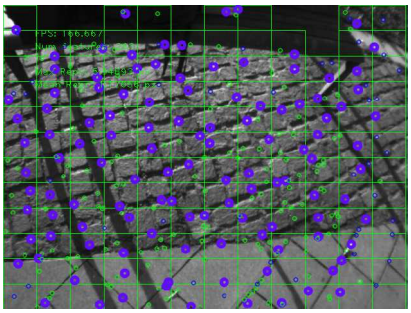
		μATE	σATE	$\|\Delta\|$	Δ
Ours	$[m]$	0.042	0.024	0.055	0.003
	[%]	0.271	0.158	0.360	0.022

Table 5.7: Error metrics for the OUT_HOVER test sequence

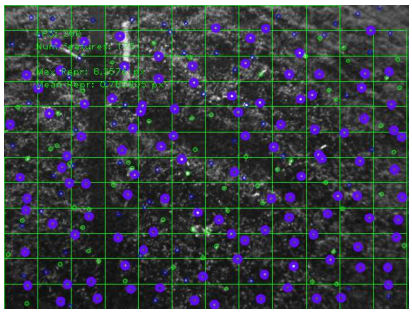
5.6.4 OUT_CIRCLE

The first sequence involves a round trip departing and ending from approximately the same location. After a first brief hovering motion (small circle in trajectories of figure 5.34), the camera moves on a longer circular trajectory observing a feature-rich natural environment. The length of the D-GPS track is 19.2 meters, long enough to highlight the amount of translational drift that can occur. Correlations between the D-GPS data and the estimated poses are established on the basis of timestamps by finding the closest ground truth and estimated poses in time. Correspondences are denoted with blue segments connecting the closest D-GPS measurement to each keyframe in figure 5.34. D-GPS and keyframe poses must be referred to the same reference system in order to estimate the errors since the D-GPS is expressed in a system centered on the base station and the visual odometry select the first camera frame as the global coordinate system. Given a set of correspondences between the two trajectories, Horn alignment is applied for computing a fitting rotation as suggested by the authors of [135]. The same error metrics used for the tests in section 5.6.1 are used here to evaluate tracking performances and are reported in table 5.6 showing a mean Absolute Trajectory Errors of about 8cm which constitutes the 0.4% relatively to the entire trajectory length. Figure 5.34.c reports a side view of the trajectory and reconstruction highlighting on the left side a slight curvature of the environment. This could be either related to the actual geometry of the observed scenario or very little errors in determining the distortion coefficients for the camera, which is observed also by the authors of [150] and could be corrected by fusion of attitude measurements from an IMU.

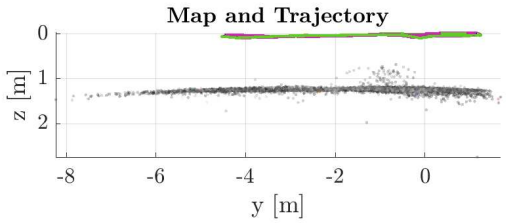
Figure 5.37 reports the Absolute Trajectory Errors for this sequence after aligning D-GPS data. The errors are mainly related to the x and y components of the motion, which are approximately parallel to the ground. z errors amount in average to 0.01-0.02 meters and are always lower than 0.05 meters. Note that the D-GPS track is accurate at centimeter level and the manufacturer software reports an uncertainty estimate for each pose. Most of the time the errors are contained in the 1 σ range so it is just possible to infer that z drifts are negligible. This demonstrates that the algorithm is able to recover a robust and stable scale factor all throughout the motion with minimal or absent drift.



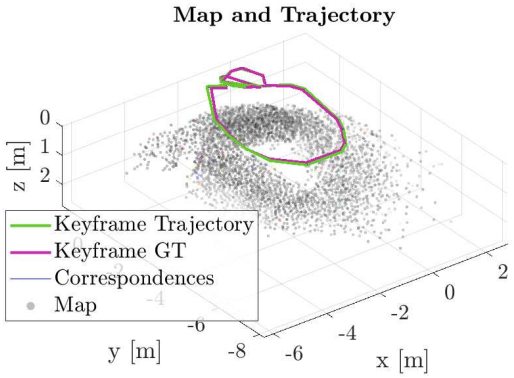
(a) Example view 1



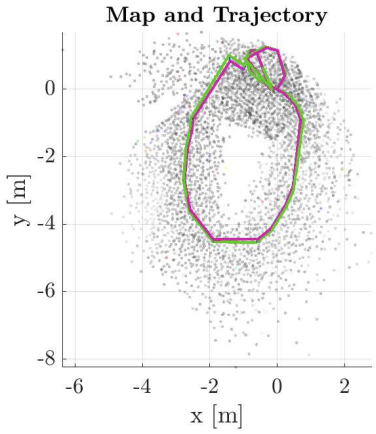
(b) Example view 2



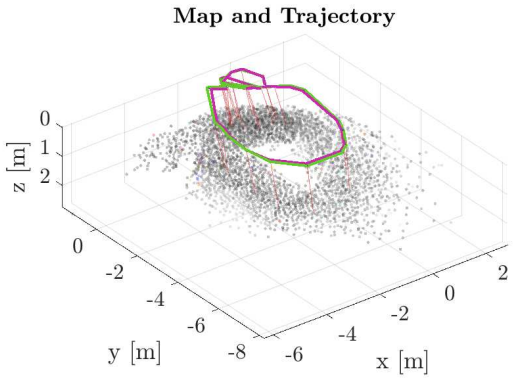
(c) Front view



(d) 3D view

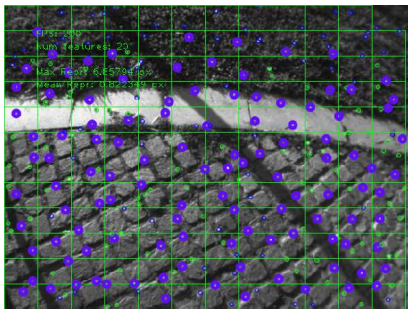


(e) Top view

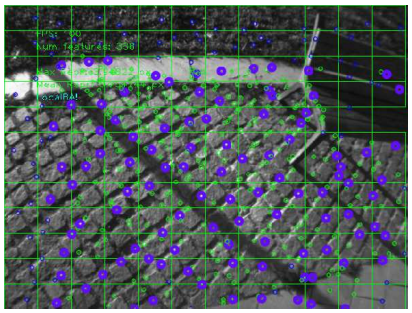


(f) 3D view with ranges

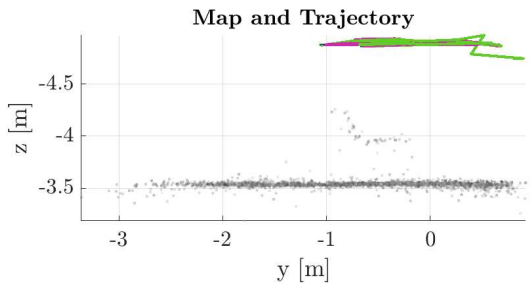
Figure 5.34: Views of trajectory and map for the sequence *OUT_CIRCLE* as computed by the algorithm. Colored dots are 3D landmarks triangulated and optimized during the sequence, the green line is the estimated trajectory and in magenta the ground truth from the motion capture system. (d) represents also altimeter measurements from the set of accepted associations (see section 5.3) to highlight the recovery of a correct scale



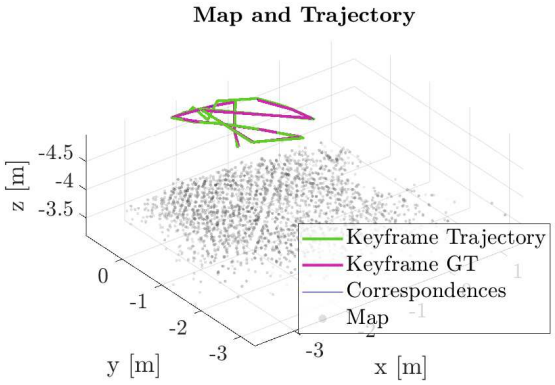
(a) Example view 1



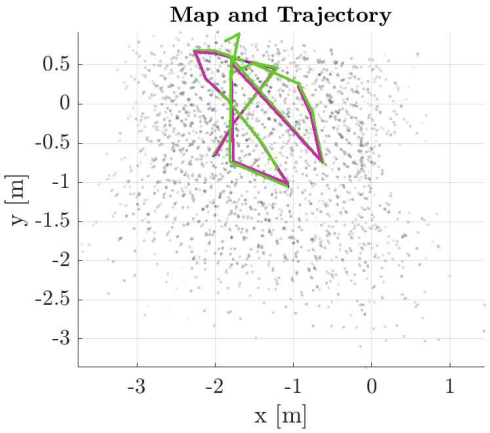
(b) Example view 2



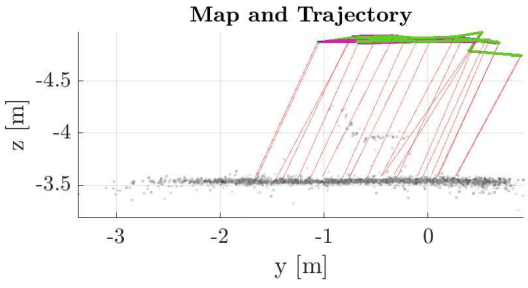
(c) Front view



(d) 3D view



(e) Top view



(f) Front view with ranges

Figure 5.35: Views of trajectory and map for the sequence *OUT_HOVER* as computed by the algorithm. Colored dots are 3D landmarks triangulated and optimized during the sequence, the green line is the estimated keyframe trajectory and in magenta the ground truth from the motion capture system. (d) represents also altimeter measurements from the set of accepted associations (see section 5.3) to highlight the recovery of a correct scale

		μATE	σATE	$\lvert\Delta\rvert$	Δ
Ours	$[m]$	0.022	0.013	0.023	0.006
	[%]	0.459	0.255	0.46	0.124

Table 5.8: Error metrics for the OUT_LANDING test sequence

5.6.5 OUT_HOVER

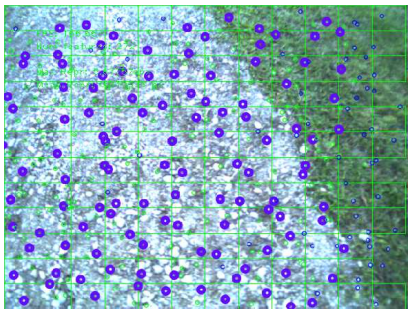
In this test, the motion is constrained in an approximately square box of 2x2 meters simulating a stable hover. Even though the motion is constrained in a very small area, the trajectory length is still meaningful for evaluating eventual drift being 15.6 meters in length. Figure 5.35 shows some example view from this sequence as well as the trajectories and reconstructions from various angles. As for the previous sequence is plotted the keyframe trajectory therefore appearing a sequence of segments instead of a smooth path. Only keyframe poses are in fact optimized in the *back-end*. Figure 5.35 demonstrates the flatness of the reconstructed man-made environment (the points not belonging to the ground belongs to the chair visible in figure 5.35.b). Figure 5.35.f shows also the altimeter measurements form the keyframes where depth associations were embedded in the Bundle Adjustment. As reported by table 5.7 and by figure 5.37 the Absolute Trajectory Errors in this sequence are very low being contained in the 5cm range for x and y and always below the 1 σ uncertainty of the D-GPS for the z coordinate. As the hovering motion of the camera keeps track of landmarks for a high number of keyframes, the pose drift is minimal as the algorithm is computing the pose based on the same landmarks for most of the time without triangulating new ones. This motivates the usefulness of the proposed approach for UAV navigation contrarily to pure incremental visual odometry approaches [90] [106].

5.6.6 OUT_LANDING

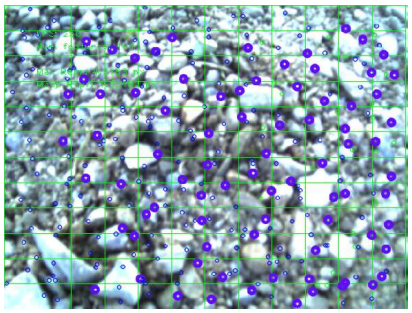
The last of the three sequences demonstrates the performance of the algorithm during rapid altitude changes. The sensor setup is moved in a spiraling motion toward the ground (from figure 5.35.a to 5.35.b) starting from a height of just over 1 meters and ending at a few centimeters from the ground. For this sequence and the previous, D-GPS coverage is not available for the entire sequences as the differential fix with the base station was unstable causing the track to drift. The D-GPS track was then filtered removing the regions with highest uncertainty in pose, which during fix loss were higher than 1 meter in all the three directions. Even if ground truth is available for a limited part of the trajectory, the errors shown in table 5.7 and in figure 5.37 are consistent with the other sequences and the reconstructed map visible in figures 5.35.c and 5.35.d suggest good tracking accuracy also during the remaining part of the trajectory.

5.6.7 Scale robustness and timings

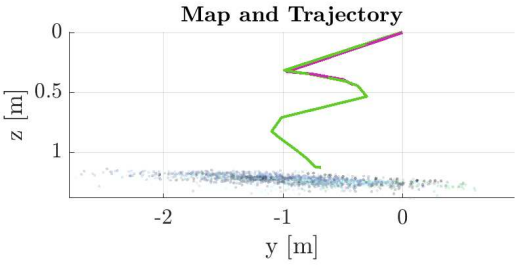
Figure 5.38 reports for all sequences, the difference between the altimeter measurements and the depth of the associated landmarks for each keyframe. This allows to evaluate the instantaneous scale drift over time. As the figure shows, depth errors are always in average contained in the 2 to 3 centimeter range even during the OUT_LANDING sequence where



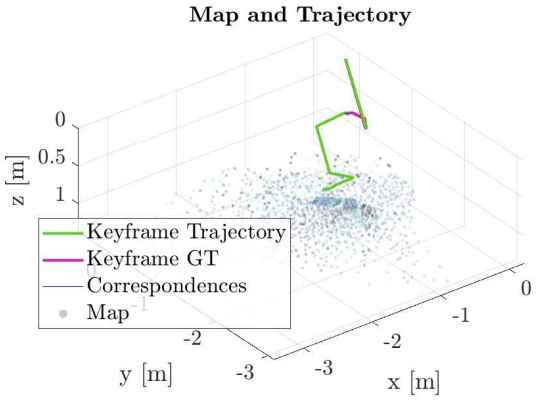
(a) Example view 1



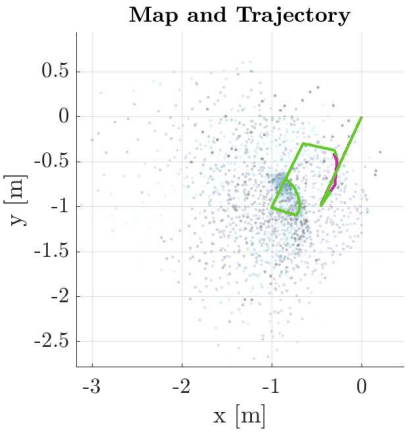
(b) Example view 2



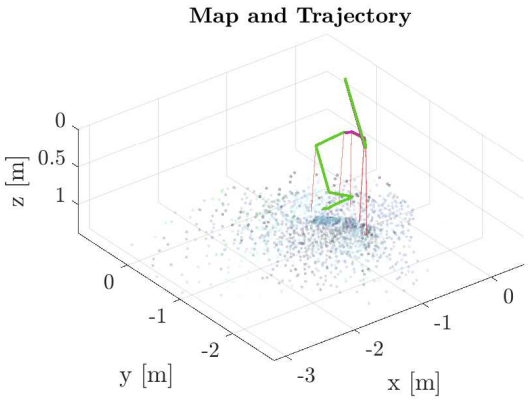
(c) Front view



(d) 3D view

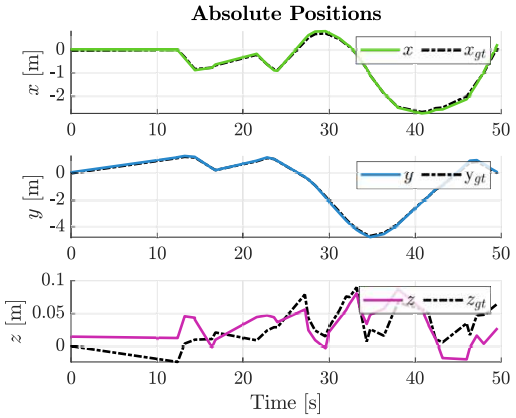


(e) Top view

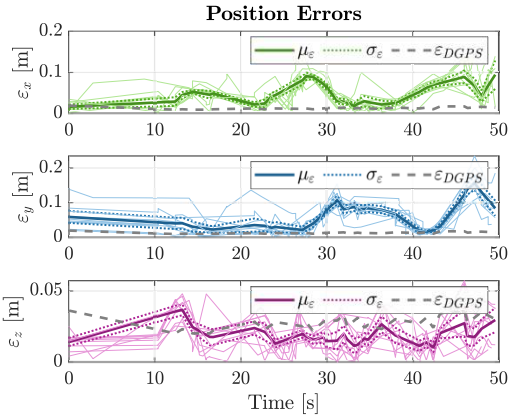


(f) 3D view with ranges

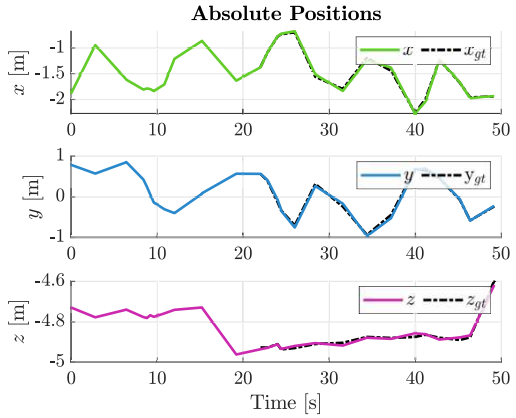
Figure 5.36: Views of trajectory and map for the sequence *OUT_LANDING* as computed by the algorithm. Colored dots are 3D landmarks triangulated and optimized during the sequence, the green line is the estimated keyframe trajectory and in magenta the ground truth from the motion capture system. (d) represents also altimeter measurements from the set of accepted associations (see section 5.3) to highlight the recovery of a correct scale



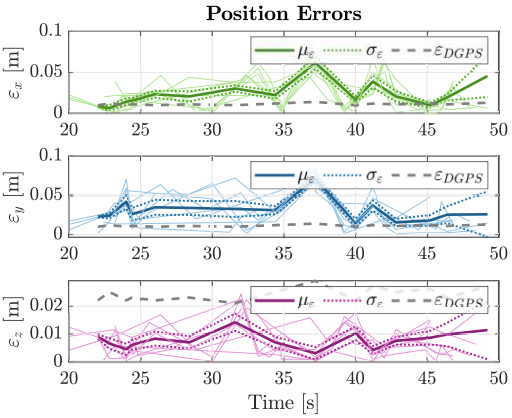
(a) OUT_CIRCLE



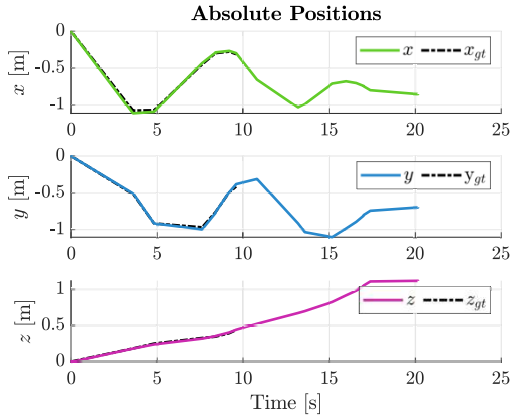
(b) OUT_CIRCLE



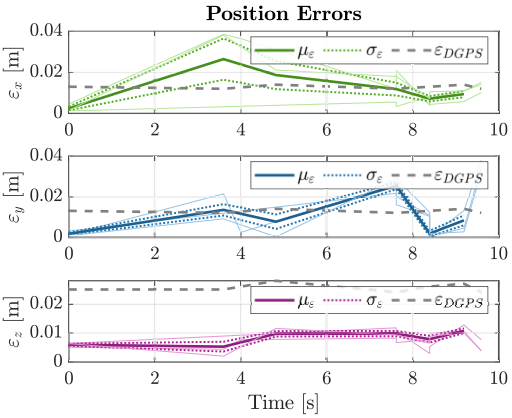
(c) OUT_HOVER



(d) OUT_HOVER



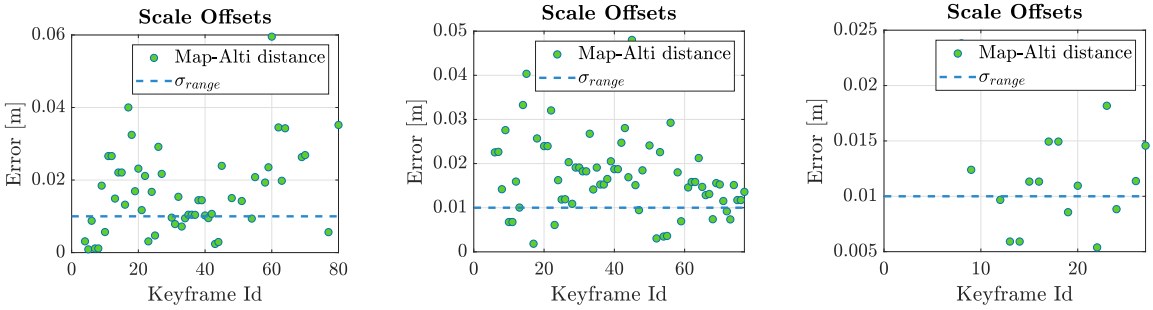
(e) OUT_LANDING



(f) OUT_LANDING

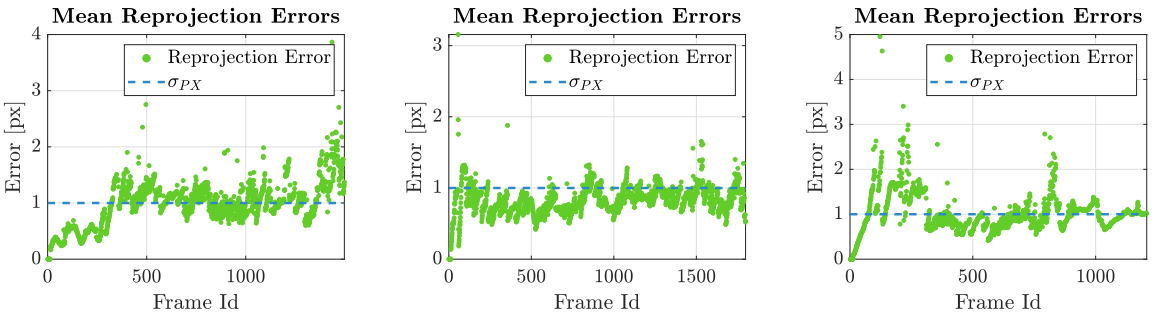
Figure 5.37: Trajectory coordinates and errors for the three sequences OUT_CIRCLE, OUT_HOVER, OUT_LANDING. Estimated trajectories in the left column are plotted against the D-GPS aligned trajectories while Absolute Trajectory Errors in the right column are plotted against the estimated uncertainty of the D-GPS poses provided for each pose by the manufacturer software. Note that errors along the z direction are low and consistent with the measurement accuracy demonstrating consistency in the scale estimates. Ground truth is lacking in some areas for the second and third trajectories due to high uncertainty in the D-GPS estimates

the range between the camera and ground was changing rapidly. Furthermore, errors are always lower than 5 cm range which is the accuracy of the LightWare SF-10B LiDAR altimeter as reported in the datasheet. This result demonstrates the consistency between the sensor properties and the results of the visual odometry as an estimator for the camera pose. In addition, figure 5.39 reports the average landmarks reprojection errors for all frames in each sequence. The reprojection errors (eq 5.9) define also the cost function for regressing the camera pose and are limited by the accuracy of feature tracking from sparse optical flow. During the Bundle Adjustment, a σ of 1 pixel is associated to each reprojection factor as a value which generally describe tracking accuracy. It is evident in figure 5.39 as the mean reprojection error is always contained or centered around the 1 pixel range. This demonstrates firstly that the optimization scheme is able to contain the drift in camera pose accurately up to the feature tracking uncertainty and also that the selected uncertainty value is well representative of the error model for tracking. It can be also observed that the reprojection error increases slightly during time and decreases instantaneously in a constant "zig-zag" motion pattern which shows the effect of each Bundle Adjustment triggered at keyframe rate and optimizing for the lowest reprojection error (other than depth). Figure 5.40 reports timings for the both the *front-end* and *back-end* during the outdoor sequences. This time, a more powerful Intel i7 processor was used instead of the Intel i5 for the RGB-D dataset tests. However, while multi-core performances are double, single core performances are very comparable and only marginally better. It can be seen that almost all frames in the sequences are processed with a higher rate than the nominal 30 Hz of the video stream (or 33 milliseconds per frame) suggesting very good performances also in terms of efficiency. Spikes are visible from time to time due to lack of optimization at code level, especially related to concurrency between the two computational threads for the *front-end* and *back-end*. Figure 5.40 also reports the timing required for each Bundle Adjustment at keyframe generation time, the implemented incremental optimization scheme proves here to be very efficient requiring in average less than 100 milliseconds (or less than a 3 frames time span) to adjust the local trajectory and map depending on the number of landmarks involved in the optimization.



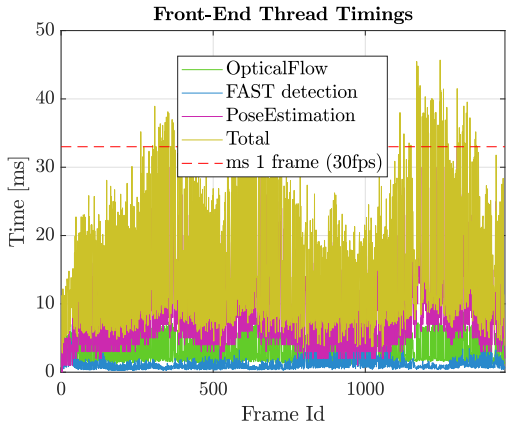
(a) OUT_CIRCLE (b) OUT_HOVER (c) OUT_LANDING

Figure 5.38: Scale offsets during the three sequences. Offsets are the average depth difference between the 3D landmarks associated to an altimeter projection and its measure. It is highlighted for reference the σ_{range} of the altimeter measurements given by the manufacturer. These scale offset values are computed before Bundle Adjustments at keyframe level.

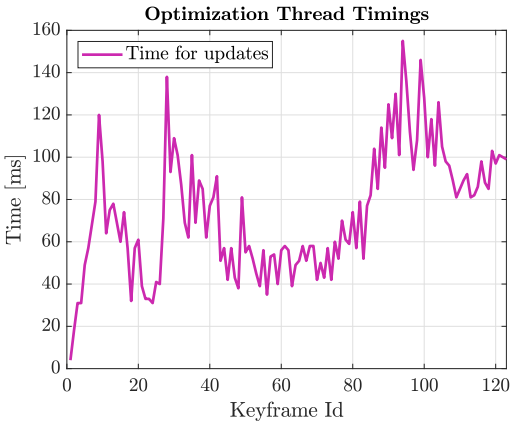


(a) OUT_CIRCLE (b) OUT_HOVER (c) OUT_LANDING

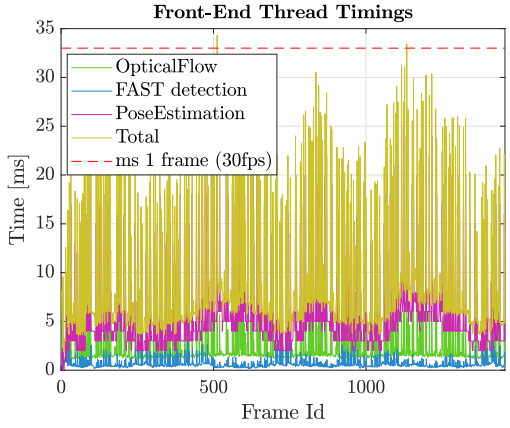
Figure 5.39: Mean reprojection errors for the three sequences. The error is computed as the norm of the 2 element vector in eq. 5.9 and it is averaged for all tracked landmark in each frame. To all plots is highlighted for visual reference the defined uncertainty over each feature detection in the image (1 pixel for the u and v coordinates). Note how the developed optimizer is able to constrain both the reprojection error, indicating consistency of the 3D map with visual measurements, and scale (figure 5.38)



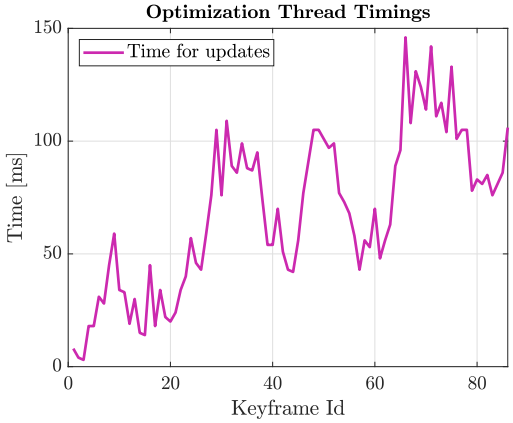
(a) OUT_CIRCLE



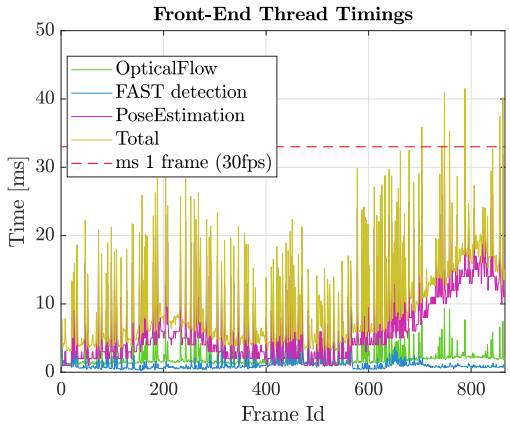
(b) OUT_CIRCLE



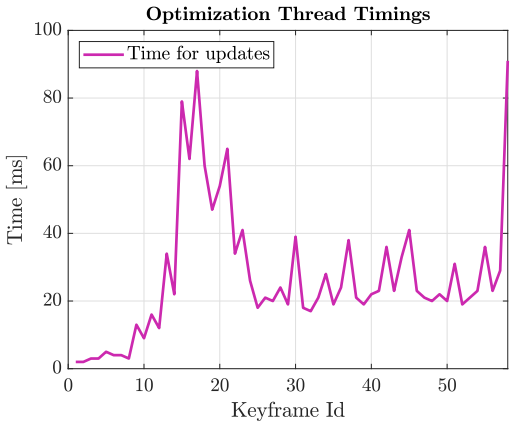
(c) OUT_HOVER



(d) OUT_HOVER



(e) OUT_LANDING



(f) OUT_LANDING

Figure 5.40: Left column: computational times per each frame for the *front-end* part of the algorithm. It is highlighted in each plot the maximum time allowed to process each frame from a 30Hz image stream. Right column: computational times per each iSAM2 update. In each sequence, the optimization *back-end* spends in average less than 100 milliseconds for Bundle Adjustment. All these times are related to a laptop-grade i7 CPU.

Chapter 6

Conclusions

In the first part of this thesis work a robust re-localization pipeline based on stereo submaps has been presented and tested. We showed how 3D local descriptors can be binarized and enriched with texture information to obtain a precise and distinctive local signature, fast to extract and match using the Hamming distance. We demonstrated how the Bag-of-Words paradigm can be applied to 3D submaps, as containers of descriptors, deriving an extremely light representation to compute their similarity in a fast and efficient way. The proposed candidate selection scheme was validated in laboratory datasets, where submaps from the LRU and LRU2 rovers, developed at the DLR Robotics and Mechatronics Center, were compared and matched in milliseconds with precisions higher than 70%. A novel outlier rejection scheme was introduced outperforming traditional RANSAC-based approaches to validate candidate submap correspondences. We demonstrated the effectiveness of our approach in a challenging outdoor scenario on Mount Etna, designated as a planetary analogous environment. The complexity of the Etna datasets is due to the fact that visual information is very repetitive and therefore ambiguous, leading to the state-of-the-art visual SLAM system ORB-SLAM2 to fail in this scenario. Our algorithm on the other hand was able to re-localize on multiple mapping sessions detecting matching submap pairs with 100% precision after validation and accurately aligning the 3D maps.

In the second part of this thesis is demonstrated that monocular vision systems can be fused efficiently with low resolution range sensors to recover and maintain a correct metric scale. A scale-aware monocular Visual Odometry prototype was tested comparing pose estimation accuracy and scale drift using a low resolution Time of Flight camera delivering 64 range measurements in an 8.7x8.7 degrees Field of View and a heavy and precise 2D LiDAR measuring 360 scan points in a 180 degrees angle. Results showed that while using a more accurate range sensor such as the 2D LiDAR delivers slightly higher performances, pose estimates employing the ToF camera are still comparable with state of the art stereo Visual SLAM while requiring less computational effort due to handling a single image stream. Having demonstrated that our sensor fusion approach is effective, a real-time implementation of a scale correct monocular Visual Odometry has been presented. A minimalistic range sensing setup comprising a single-point off-the-shelf LiDAR altimeter has been employed to initialize and correct the metric scale using an incremental optimizer running on an independent thread. The proposed algorithm has been tested in the RGB-D

TUM dataset outperforming RGB-D SLAM while using just one range measurement per keyframe. In addition, a proper test setup comprising a color camera and a LightWare SF-10 altimeter was used in conjunction with a differential GPS in outdoor scenarios testing the pose estimation accuracy simulating a ventral camera setup for an UAV. In all tests the maximum error did not exceed the 0.5% of the total travelled length, demonstrating state of the art performances using very limited resources.

Bibliography

[1] T. Ahonen, A. Hadid, and M. Pietikainen. "Face description with local binary patterns: Application to face recognition". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12 (2006), pp. 2037–2041.

[2] H. Alismail, L. D. Baker, and B. Browning. "Automatic Calibration of a Range Sensor and Camera System". In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*. 2012, pp. 286–292. DOI: 10.1109/3DIMPVT.2012.52.

[3] F. Andert and L. Mejias. "Improving Monocular SLAM with Altimeter Hints for Fixed-Wing Aicraft Navigation and Emergency Landing". In: *Proc. International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015.

[4] F. Andert et al. "Optical-Aided Aircraft Navigation using Decoupled Visual SLAM with Range Sensor Augmentation". In: *Intelligent Robotic Systems* (2017).

[5] D. Arthur and S. Vassilvitskii. "K-means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

[6] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology Behind Search*. 2nd. USA: Addison-Wesley Publishing Company, 2008. ISBN: 9780321416919.

[7] B. Balaram et al. "Mars Helicopter Technology Demonstrator". In: *2018 AIAA Atmospheric Flight Mechanics Conference*. DOI: 10.2514/6.2018-0023. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-0023. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2018-0023.

[8] H. Bay et al. "SURF: Speeded Up Robust Features". In: *Computer Vision and Image Understanding (CVIU)* 110.3 (2008), pp. 346–359.

[9] S. Benhimane and E. Malis. "Homography-based 2d visual tracking and servoing". In: *The International Journal of Robotics Research* 26.7 (2007), pp. 661–676.

[10] P. Biber and W. Strasser. "The normal distributions transform: a new approach to laser scan matching". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 3. 2003, 2743–2748 vol.3. DOI: 10.1109/IROS.2003.1249285.

[11] J.-P Bibring et al. “The Rosetta Lander (Philae) Investigations”. In: *Space Science Reviews* 128 (Feb. 2007). DOI: 10.1007/s11214-006-9138-2.

[12] S. Birchfield and C. Tomasi. “A pixel dissimilarity measure that is insensitive to image sampling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.4 (1998), pp. 401–406.

[13] J. Biswas and M. Veloso. “Depth camera based indoor mobile robot localization and navigation”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2012).

[14] D. F. Blake et al. “Curiosity at Gale Crater, Mars: Characterization and Analysis of the Rocknest Sand Shadow”. In: *Science* 341.6153 (2013). ISSN: 0036-8075. DOI: 10.1126/science.1239505. eprint: https://science.sciencemag.org/content/341/6153/1239505.full.pdf. URL: https://science.sciencemag.org/content/341/6153/1239505.

[15] J. yves Bouguet. “Pyramidal implementation of the Lucas Kanade feature tracker”. In: *Intel Corporation, Microprocessor Research Labs* (2000).

[16] C. Brand et al. “Stereo-vision based obstacle mapping for indoor/outdoor SLAM”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1846–1853.

[17] C. Brand et al. “Submap matching for stereo-vision based indoor/outdoor SLAM”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5670–5677.

[18] D. Brown, J. Wendel, and D. Agle. *Mars Helicopter to Fly on NASA Next Red Planet Rover Mission*. 2018. URL: https://www.nasa.gov/press-release/mars-helicopter-to-fly-on-nasa-s-next-red-planet-rover-mission (visited on 05/11/2018).

[19] W. Burgard et al. “Coordinated multi-robot exploration”. In: *IEEE Transactions on Robotics* 21.3 (2005), pp. 376–386. ISSN: 1552-3098. DOI: 10.1109/TRO.2004.839232.

[20] S. Campagnola et al. “Mission analysis for the Martian Moons Explorer (MMX) mission”. In: *Acta Astronautica* 146 (2018), pp. 409 –417. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2018.03.024. URL: http://www.sciencedirect.com/science/article/pii/S0094576517317575.

[21] C.-F. Chen, M. Bolas, and E. S. Rosenberg. “Rapid creation of photorealistic virtual reality content with consumer depth cameras”. In: *IEEE Virtual Reality (VR)* (2017).

[22] H. Chen and B. Bhanu. “3D free-form object recognition in range images using local surface patches”. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3. 2004, 136–139 Vol.3. DOI: 10.1109/ICPR.2004.1334487.

[23] W. Cheung and G. Hamarneh. "n -SIFT: n -Dimensional Scale Invariant Feature Transform". In: *IEEE Transactions on Image Processing* 18.9 (2009), pp. 2012–2021. ISSN: 1057-7149. DOI: 10.1109/TIP.2009.2024578.

[24] S. Chiodini et al. "Monocular visual odometry aided by a low resolution time of flight camera". In: *Proc. IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. 2017. DOI: 10.1109/MetroAeroSpace.2017.7999572.

[25] S. Chiodini et al. "MORPHEUS: A Field Robotics Testbed for Soil Sampling and Autonomous Navigation". In: *Proc. 1st Symposium on Space Educational Activities*. 2015.

[26] C.-S. Chua and R. Jarvis. "Point Signatures: A New Representation for 3D Object Recognition". In: *International Journal of Computer Vision* 25 (Oct. 1997), pp. 63–85. DOI: 10.1023/A:1007981719186.

[27] T. Conceição et al. "Joint Visual and Time-of-Flight Camera Calibration for an Automatic Procedure in Space". In: 2018.

[28] A. Concha and J. Civera. "RGBDTAM: A cost-effective and accurate RGB-D tracking and mapping system". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 6756–6763.

[29] K. P. Cop, P. V. K. Borges, and R. Dubé. "Delight: An Efficient Descriptor for Global Localisation Using LiDAR Intensities". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3653–3660. DOI: 10.1109/ICRA.2018.8460940.

[30] M. Cummins and P. Newman. "Appearance-only SLAM at large scale with FAB-MAP 2.0". In: *The International Journal of Robotics Research* 30.9 (2011), pp. 1100–1123.

[31] M. Cummins and P. Newman. "FAB-MAP: Probabilistic localization and mapping in the space of appearance". In: *The International Journal of Robotics Research* 27.6 (2008), pp. 647–665.

[32] M. Donoser and H. Bischof. "Efficient maximally stable extremal region (MSER) tracking". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. Ieee. 2006, pp. 553–560.

[33] G. Dorian and J. D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197. ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2197158.

[34] R. Dubé et al. "An online multi-robot SLAM system for 3D LiDARs". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1004–1011. DOI: 10.1109/IROS.2017.8202268.

[35] R. Dubé et al. "Incremental-Segment-Based Localization in 3-D Point Clouds". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1832–1839. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2803213.

[36] F. Endres et al. “3D Mapping with an RGB-D Camera”. In: *IEEE Transactions on Robotics* (2014).

[37] J. Engel, J. Sturm, and D. Cremers. “Scale-aware navigation of a low-cost quadrocopter with a monocular camera”. In: *Robotics and Autonomous Systems* (2014).

[38] T. Estlin et al. “Coordinating multiple spacecraft assets for joint science campaigns”. In: (2010).

[39] S. Farboud-Sheshdeh, T. D. Barfoot, and R. H. Kwong. “Towards Estimating Bias in Stereo Visual Odometry”. In: *Canadian Conference on Computer and Robot Vision, CRV 2014, Montreal, QC, Canada, May 6-9, 2014*. 2014, pp. 8–15. DOI: 10.1109/CRV.2014.10. URL: https://doi.org/10.1109/CRV.2014.10.

[40] O. D. Faugeras and F. Lustman. “Motion and structure from motion in a piecewise planar environment”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 2.03 (1988), pp. 485–508.

[41] M. A. Fischler and R. C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[42] C. Forster, M. Pizzoli, and D. Scaramuzza. “Air-ground localization and map augmentation using monocular dense reconstruction”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 3971–3978. DOI: 10.1109/IROS.2013.6696924.

[43] C. Forster et al. “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems”. In: *IEEE Transactions on Robotics* 33 (2017).

[44] D. Galvez-Lopez. *Place and Object Recognition for Real-time Visual Mapping*. 2013.

[45] S. Garcia et al. “Indoor SLAM for Micro Aerial Vehicles Control using Monocular Camera and Sensor Fusion”. In: *Proc. International Conference on Autonomous Robot Systems and Competitions*. 2016.

[46] A. Gawel et al. “3d registration of aerial and ground robots for disaster response: An evaluation of features, descriptors, and transformation estimation”. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE. 2017, pp. 27–34.

[47] A. Geiger et al. “Automatic camera and range sensor calibration using a single shot”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3936–3943. DOI: 10.1109/ICRA.2012.6224570.

[48] A. Geiger, J. Ziegler, and C. Stiller. “StereoScan: Dense 3D Reconstruction in Real-time”. In: 2011.

[49] Y. Girdhar et al. “Streaming Scene Maps for Co-Robotic Exploration in Bandwidth Limited Environments”. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019). DOI: 10.1109/icra.2019.8794132. URL: http://dx.doi.org/10.1109/ICRA.2019.8794132.

[50] R. Giubilato et al. “An evaluation of ROS-compatible stereo visual SLAM methods on a nVidia Jetson TX2”. In: *Measurement* 140 (2019), pp. 161–170.

[51] R. Giubilato et al. “Scale Correct Monocular Visual Odometry Using a LiDAR Altimeter”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*. 2018, pp. 3694–3700. DOI: 10.1109/IROS.2018.8594096. URL: https://doi.org/10.1109/IROS.2018.8594096.

[52] X. Gong, Y. Lin, and J. Liu. “3D LIDAR-camera extrinsic calibration using an arbitrary trihedron”. In: *Sensors* 13.2 (2013), pp. 1902–1918.

[53] J. Graeter, A. Wilczynski, and M. Lauer. “LIMO: Lidar-Monocular Visual Odometry”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7872–7879.

[54] I. Grixa et al. “Appearance-Based Along-Route Localization for Planetary Missions”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 6327–6334.

[55] M. Grott et al. “Low thermal conductivity boulder with high porosity identified on C-type asteroid (162173) Ryugu”. In: *Nature Astronomy* (2019), pp. 1–6.

[56] J. Guo et al. “Local Descriptor for Robust Place Recognition Using LiDAR Intensity”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1470–1477. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2893887.

[57] Y. Guo et al. “A comprehensive performance evaluation of 3D local feature descriptors”. In: *International Journal of Computer Vision* 116.1 (2016), pp. 66–89.

[58] R. Haarmann et al. “Mobile Payload Element (MPE): Concept study for a sample fetching rover for the ESA Lunar Lander Mission”. In: *Planetary and Space Science* 74.1 (2012), pp. 283–295.

[59] M. Hansard et al. “Cross-Calibration of Time-of-flight and Colour Cameras”. In: *Computer Vision and Image Understanding*. Image Understanding for Real-world Distributed Video Networks 134 (May 2015), pp. 105–115. DOI: 10.1016/j.cviu.2014.09.001. URL: https://hal.inria.fr/hal-01059891.

[60] R. Hänsch, T. Weber, and O. Hellwich. “Comparison of 3D interest point detectors and descriptors for point cloud fusion”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2.3 (2014), p. 57.

[61] C. Harris and M. Stephens. “A combined corner and edge detector”. In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151.

[62] R. Hartley and A. Zissermann. *Multiple Views Geometry in Computer Vision*.

[63] R. I. Hartley. “In defense of the eight-point algorithm”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (1997), pp. 580–593. ISSN: 0162-8828. DOI: 10.1109/34.601246.

[64] H. Hirschmuller. "Stereo processing by semiglobal matching and mutual information". In: *IEEE Transactions on pattern analysis and machine intelligence* 30.2 (2008), pp. 328–341.

[65] H. Hirschmuller, P. R. Innocent, and J. M. Garibaldi. "Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics". In: *7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002.* Vol. 2. IEEE. 2002, pp. 1099–1104.

[66] B. K. P. Horn. "Closed-form solution of absolute orientation using unit quaternions". In: *J. Opt. Soc. Am. A* 4.4 (1987), pp. 629–642. DOI: 10.1364/JOSAA.4.000629. URL: http://josaa.osa.org/abstract.cfm?URI=josaa-4-4-629.

[67] A. S. Huang et al. "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera". In: *Robotics Research : The 15th International Symposium ISRR*. 2017.

[68] A. E. Johnson and M. Hebert. "Using spin images for efficient object recognition in cluttered 3D scenes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.5 (1999), pp. 433–449. ISSN: 0162-8828. DOI: 10.1109/34.765655.

[69] A. E. Johnson et al. "Robust and Efficient Stereo Feature Tracking for Visual Odometry". In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 39–46. DOI: 10.1109/ROBOT.2008.4543184.

[70] J. Jung et al. "Time-of-Flight Sensor Calibration for a Color and Depth Camera Pair". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.7 (2015), pp. 1501–1513. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2014.2363827.

[71] M. Kaess et al. "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree". In: *Intl. J. of Robotics Research, IJRR* 31.2 (2012), pp. 217–236.

[72] G. Kim and A. Kim. "Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Madrid, 2018.

[73] G. Klein and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*. Nara, Japan, 2007.

[74] L. Kneip, D. Scaramuzza, and R. Siegwart. "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation". In: *IEEE Computer Vision and Pattern Recognition (CVPR)* (2011).

[75] S. Kodgule, A. Candela, and D. Wettergreen. "Non-myopic Planetary Exploration Combining In Situ and Remote Measurements". In: *arXiv preprint arXiv:1904.12255* (2019).

[76] C. Krause and U. A. et al. "MASCOT. a Mobile Lander on-board Hayabusa2 Spacecraft, Status and Operational Concept for the Asteroid Ryugu". In: *2018 SpaceOps Conference*. DOI: 10.2514/6.2018-2418. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2418. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2018-2418.

[77] R. Kummerle et al. “On measuring the accuracy of SLAM algorithms”. In: *Autonomous Robots* 27.4 (2009), p. 387. ISSN: 1573-7527. DOI: 10.1007/s10514-009-9155-6. URL: https://doi.org/10.1007/s10514-009-9155-6.

[78] K. Kwak et al. “Extrinsic calibration of a single line scanning lidar and a camera”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 3283–3289. DOI: 10.1109/IROS.2011.6094490.

[79] M. Labbé and F. Michaud. “Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation”. In: *IEEE Transactions on Robotics* 29.3 (2013).

[80] G. H. Lee, F. Fraundorfer, and M. Pollefeys. “Robust pose-graph loop-closures with expectation-maximization”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 556–563. DOI: 10.1109/IROS.2013.6696406.

[81] K. Lenc and A. Vedaldi. “Large scale evaluation of local image feature detectors on homography datasets”. In: *arXiv preprint arXiv:1807.07939* (2018).

[82] V. Lepetit, F.Moreno-Noguer, and P.Fua. “EPnP: An Accurate O(n) Solution to the PnP Problem”. In: *International Journal of Computer Vision* 81.2 (2009).

[83] E. Lopez et al. “A Multi-Sensorial Simultaneous Localization and Mapping (SLAM) system for Low-Cost Micro Aerial Vehicles in GPS-Denied Environments”. In: *Sensors* (2017).

[84] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (2004).

[85] M. Maimone et al. “Autonomous navigation results from the Mars Exploration Rover (MER) mission”. In: *Experimental robotics IX*. Springer, 2006, pp. 3–13.

[86] M. W. Maimone, P. C. Leger, and J. J. Biesiadecki. “Overview of the Mars Exploration Rovers Autonomous Mobility and Vision Capabilities”. In: 2007.

[87] J Maki et al. “The Mars science laboratory engineering cameras”. In: *Space science reviews* 170.1-4 (2012), pp. 77–93.

[88] E. Malis and M. Vargas. “Deeper understanding of the homography decomposition for vision-based control”. In: (2007), p. 90.

[89] J. Matijevic et al. “The Pathfinder Microrover”. In: *Journal of Geophysical Research* 102 (1997), pp. 3989–4002.

[90] L. H. Matthies and S. A. Shafer. “Error modeling in stereo navigation”. In: *IEEE J. Robotics and Automation* 3.3 (1987), pp. 239–248. DOI: 10.1109/JRA.1987.1087097. URL: https://doi.org/10.1109/JRA.1987.1087097.

[91] S. May et al. “Three-dimensional mapping with time-of-flight cameras”. In: *Journal of Field Robotics* (2009).

[92] T. Miki, P. Khrapchenkov, and K. Hori. “UAV/UGV Autonomous Cooperation: UAV assists UGV to climb a cliff by attaching a tether”. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019). DOI: 10.1109/icra.2019.8794265. URL: http://dx.doi.org/10.1109/ICRA.2019.8794265.

[93] M. J. Milford and G. F. Wyeth. "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 1643–1649.

[94] A. Mishchuk et al. "Working hard to know your neighbor's margins: Local descriptor learning loss". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4826–4837.

[95] H. P. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* Tech. rep. Stanford Univ CA Dept of Computer Science, 1980.

[96] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[97] R. Mur-Artal and J. D. Tardós. "Fast relocalisation and loop closing in keyframe-based SLAM". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 846–853.

[98] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *abs/1610.06475* (2016). arXiv: 1610.06475. URL: http://arxiv.org/abs/1610.06475.

[99] O. Naroditsky, A. Patterson, and K. Daniilidis. "Automatic alignment of a camera with a line scan LIDAR system". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3429–3434. DOI: 10.1109/ICRA.2011.5980513.

[100] R. A. Newcombe et al. "Kinectfusion: Real-time dense surface mapping and tracking." In: *ISMAR*. Vol. 11. 2011. 2011, pp. 127–136.

[101] D. Nister. "An efficient solution to the five-point relative pose problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (2004), pp. 756–770. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.17.

[102] D. Nister and H. Stewenius. "Scalable Recognition with a Vocabulary Tree". In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2161–2168. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.264. URL: http://dx.doi.org/10.1109/CVPR.2006.264.

[103] J. Novatnack and K. Nishino. "Scale-Dependent/Invariant Local 3D Shape Descriptors for Fully Automatic Registration of Multiple Sets of Range Images". In: (Mar. 2008).

[104] D. L. Olson and D. Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[105] M. Pertile et al. "Calibration of extrinsic parameters of a hybrid vision system for navigation comprising a very low resolution Time-of-Flight camera". In: *2017 IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. 2017, pp. 391–396. DOI: 10.1109/MetroAeroSpace.2017.7999604.

[106] M. Pertile et al. “Uncertainty comparison of three visual odometry systems in different operative conditions”. In: *Measurement* 78 (2016), pp. 388–396.

[107] S. M. Prakhya et al. “B-SHOT: a binary 3D feature descriptor for fast Keypoint matching on 3D point clouds”. In: *Autonomous Robots* 41.7 (2017), pp. 1501–1520.

[108] A Prusak et al. “Pose estimation and map building with a pmd-camera for robot navigation”. In: *Proceedings of the Dynamic 3D Imaging Workshop in Conjunction with DAGM (Dyn3D)*. Vol. 1. 2007.

[109] J. Reill et al. “Development of a mobility drive unit for low gravity planetary body exploration”. In: *12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA), ESA/ESTEC, Noordwijk, Netherlands*. 2013.

[110] J. Reill et al. “MASCOT asteroid lander with innovative mobility mechanism”. In: *ASTRA* (2015).

[111] F. Ropero, P. Muñoz, and M. D. R-Moreno. “TERRA: A path planning algorithm for cooperative UGV–UAV exploration”. In: *Engineering Applications of Artificial Intelligence* 78 (2019), pp. 260–272.

[112] E. Rosten and T. Drummond. “Machine learning for high-speed corner detection”. In: *Proc. European Conference on Computer Vision (ECCV)* (2006).

[113] S. Rusinkiewicz and M. Levoy. “Efficient variants of the ICP algorithm”. In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. 2001, pp. 145–152. DOI: 10.1109/IM.2001.924423.

[114] R. B. Rusu, N. Blodow, and M. Beetz. “Fast Point Feature Histograms (FPFH) for 3D registration”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[115] R. B. Rusu et al. “Aligning point cloud views using persistent feature histograms”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 3384–3391. DOI: 10.1109/IROS.2008.4650967.

[116] R. B. Rusu, N. Blodow, and M. Beetz. “Fast point feature histograms (FPFH) for 3D registration”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 3212–3217.

[117] A. Saxena, S. H. Chung, and A. Y. Ng. “Learning depth from single monocular images”. In: *Advances in neural information processing systems*. 2006, pp. 1161–1168.

[118] K. Schmid, F. Ruess, and D. Burschka. “Local reference filter for life-long vision aided inertial navigation”. In: *17th International Conference on Information Fusion (FUSION)*. IEEE. 2014, pp. 1–8.

[119] N Schmitz et al. “The Camera of the MASCOT Asteroid Lander on Board Hayabusa 2—Science Objectives, Imaging Sequences, and Instrument Design”. In: *Lunar and Planetary Science Conference*. Vol. 49. 2018.

[120] J. L. Schonberger et al. “Comparative evaluation of hand-crafted and learned local features”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1482–1491.

[121] J. L. Schönberger and J.-M. Frahm. “Structure-from-Motion Revisited”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[122] M. J. Schuster et al. “Distributed stereo vision-based 6D localization and mapping for multi-robot teams”. In: *Journal of Field Robotics* 36.2 (2019), pp. 305–332.

[123] M. J. Schuster et al. “The Lightweight Rover Unit (LRU), its Success in the SpaceBotCamp Challenge , and Beyond”. In: ().

[124] H. Sedlmayr et al. “MASCOT: Asteroid Lander with innovative Mobility Mechanism”. In: June 2015.

[125] J. Shi and C. Tomasi. *Good features to track*. Tech. rep. Cornell University, 1993.

[126] S. M. Siam and H. Zhang. “Fast-seqslam: A fast appearance based place recognition algorithm”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 5702–5708.

[127] E. Simo-Serra et al. “Discriminative learning of deep convolutional feature point descriptors”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 118–126.

[128] Sivic and Zisserman. “Video Google: a text retrieval approach to object matching in videos”. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.

[129] J. Sivic and A. Zisserman. “Video Google: A text retrieval approach to object matching in videos”. In: *null*. IEEE. 2003, p. 1470.

[130] S. W. Squyres et al. “The Opportunity Rover’s Athena Science Investigation at Meridiani Planum, Mars”. In: *Science* 306.5702 (2004), pp. 1698–1703. ISSN: 0036-8075. DOI: 10.1126/science.1106171. eprint: https://science.sciencemag.org/content/306/5702/1698.full.pdf. URL: https://science.sciencemag.org/content/306/5702/1698.

[131] S. W. Squyres et al. “The Spirit Rover’s Athena Science Investigation at Gusev Crater, Mars”. In: *Science* 305.5685 (2004), pp. 794–799. ISSN: 0036-8075. DOI: 10.1126/science.3050794. eprint: https://science.sciencemag.org/content/305/5685/794.full.pdf. URL: https://science.sciencemag.org/content/305/5685/794.

[132] B. Steder et al. “NARF: 3D Range Image Features for Object Recognition”. In: *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Taipei, Taiwan, 2010.

[133] F. Stein and G. Medioni. “Structural indexing: efficient 3-D object recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 125–145. ISSN: 0162-8828. DOI: 10.1109/34.121785.

[134] K. H. Strobl and G. Hirzinger. “More accurate camera and hand-eye calibrations with unknown grid pattern dimensions”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 1398–1405.

[135] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. 2012.

[136] K. Tateno et al. “CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction”. In: *CoRR* abs/1704.03489 (2017). arXiv: 1704.03489. URL: http://arxiv.org/abs/1704.03489.

[137] F. Tombari and L. Di Stefano. “Object Recognition in 3D Scenes with Occlusions and Clutter by Hough Voting”. In: *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*. 2010, pp. 349–355. DOI: 10.1109/PSIVT.2010.65.

[138] F. Tombari, S. Salti, and L. Di Stefano. “A combined texture-shape descriptor for enhanced 3D feature matching”. In: *2011 18th IEEE International Conference on Image Processing*. IEEE. 2011, pp. 809–812.

[139] F. Tombari, S. Salti, and L. Di Stefano. “Unique Signatures of Histograms for Local Surface Description”. In: *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*. ECCV’10. Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 356–369. ISBN: 3-642-15557-X, 978-3-642-15557-4. URL: http://dl.acm.org/citation.cfm?id=1927006.1927035.

[140] P. H. Torr, A. Zisserman, and S. J. Maybank. “Robust detection of degenerate configurations while estimating the fundamental matrix”. In: *Computer vision and image understanding* 71.3 (1998), pp. 312–333.

[141] F. Vasconcelos, J. P. Barreto, and U. Nunes. “A Minimal Solution for the Extrinsic Calibration of a Camera and a Laser-Rangefinder”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2097–2107. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.18.

[142] Y. Verdie et al. “TILDE: A Temporally Invariant Learned DEtector”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). DOI: 10.1109/cvpr.2015.7299165. URL: http://dx.doi.org/10.1109/CVPR.2015.7299165.

[143] P. Viola and W. M. Wells. “Alignment by maximization of mutual information”. In: *Proceedings of IEEE International Conference on Computer Vision*. 1995, pp. 16–23. DOI: 10.1109/ICCV.1995.466930.

[144] R. Volpe, T. Litwin, and L. Matthies. “Mobile robot localization by remote viewing of a colored cylinder”. In: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. Vol. 1. 1995, 257–263 vol.1. DOI: 10.1109/IROS.1995.525805.

[145] R. Volpe et al. “Rocky 7: a next generation Mars rover prototype”. In: *Advanced Robotics* 11.4 (1996), pp. 341–358. DOI: 10.1163/156855397X00362.

[146] C. Wang et al. “Learning Depth from Monocular Videos using Direct Methods”. In: *arXiv preprint arXiv:1712.00175* (2017).

[147] R. Wang, M. Schworer, and D. Cremers. "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras". In: *International Conference on Computer Vision (ICCV)*. Venice, Italy, 2017.

[148] S. Wang et al. "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks". In: *CoRR* abs/1709.08429 (2017). arXiv: 1709.08429. URL: http://arxiv.org/abs/1709.08429.

[149] A. Wedler et al. "LRU-lightweight rover unit". In:

[150] S. Weiss et al. "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium". In: *Journal of Field Robotics* 30.5 (2013), pp. 803–831.

[151] T. Whelan et al. "ElasticFusion: Dense SLAM Without A Pose Graph". In: *IEEE Robotics Science and Systems* (2015).

[152] N. Yang et al. "Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry". In: *eccv*. 2018.

[153] K. M. Yi et al. "Lift: Learned invariant feature transform". In: *European Conference on Computer Vision*. Springer. 2016, pp. 467–483.

[154] J. Zhang and S. Singh. "LOAM: Lidar Odometry and Mapping in Real-time". In: *Robotics: Science and Systems Conference (RSS)*. 2014.

[155] J. Zhang and S. Singh. "Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. 2015.

[156] L. Zhang and S. Rusinkiewicz. "Learning to Detect Features in Texture Images". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6325–6333. DOI: 10.1109/CVPR.2018.00662.

[157] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000).

[158] Y. Zhong. "Intrinsic shape signatures: A shape descriptor for 3D object recognition". In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. 2009, pp. 689–696. DOI: 10.1109/ICCVW.2009.5457637.