Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Simultaneous Localization and Mapping in Dynamic Environments

Katrin Sarah Santner

Graz, Austria, May 2012

Thesis supervisors

Prof. Dr. Horst Bischof

Prof. Dr. Konrad Schindler

Nothing endures but change.

Heraklit

Abstract

The simultaneous localization and mapping (SLAM) problem of an autonomous mobile robot aims at answering two key questions: *"Where is the robot?"* and *"How does the world around it look like?"*. Solving the SLAM problem involves constructing a map of a previously unknown environment while simultaneously localizing within this map using observations from different sensors perceiving the outside world. Generally, SLAM is an essential prerequisite for autonomous mobile robots to accomplish higher level tasks such as path planning, obstacle avoidance or grasping. Nowadays, digital cameras are an interesting alternative to established sensor technologies as they are lightweight, widely-used, cheap, small and battery-saving.

Today's vision based solutions - called *vSLAM* - either assume a static environment or simply 'forget' old parts of the map to cope with map size constraints and scene dynamics. Especially when performing large-scale perpetual localization and mapping, one faces problems like memory consumption, computation time, scalability and robustness. Moreover, scenes containing repetitive and dynamic scene elements require robust data association methods.

In this thesis we propose a visual SLAM method able to cope with scene dynamics in large-scale environments by employing a novel map representation for sparse visual features. A new 3D point descriptor called Histogram of Oriented Cameras (HOC) encodes anisotropic spatial visibility information and the importance of each three-dimensional landmark. Consequently, dynamic elements do not affect localization performance as the constructed map implicitly adapts to dynamic changes during mapping. To gain efficiency and scalability during map building and loop closing, camera poses are organized in an undirected graph.

To evaluate the proposed methods, a new benchmark dataset aiming at long-term mapping within an ever changing world has been recorded. The developed SLAM system is extensively evaluated in a series of simulated and real-world experiments.

We demonstrate the ability of handling dynamic changes in the map, we can improve localization accuracy and data association and we are able to allow reasonable control of the map size.

Keywords: computer vision, mobile robotics, simultaneous localization and mapping, structure from motion, dynamic environments, benchmark dataset, SLAM, visual SLAM, histogram of oriented cameras

Kurzfassung

Die simultane Lokalisierung und Kartierung (SLAM) zielt auf die Beantwortung zweier zentraler Fragen ab: *"Wo ist der Roboter"?* Und *"Wie sieht die Welt um ihn herum aus?"*. Mit Hilfe verschiedener Sensordaten wird eine Karte einer bisher unbekannten Umgebung erstellt und zeitgleich die Position des Roboters innerhalb dieser Karte bestimmt. Das Lösen des SLAM Problems ist eine wesentliche Voraussetzung für weitere Aufgaben, wie Pfadplanung, Hindernisserkennung oder das Greifen von Gegenständen. Heutzutage sind digitale Kameras eine interessante Alternative zu etablierten Sensor-Technologien, da sie leicht, weit verbreitet, billig, kompakt und stromsparend sind.

Heutige kamera-basierte Lösungen - *vSLAM* genannt - nehmen entweder eine statische Umgebung an oder "vergessen" einfach veraltete Teile einer Karte um erhöhtem Speicherbedarf und dynamischen Umgebungen entgegenzuwirken. Insbesondere in großen Umgebungen steht man vor Herausforderungen wie Speicherbedarf, Rechenzeit, Skalierbarkeit und Robustheit. Darüber hinaus erfordern Szenen mit repetitiven Elementen und dynamische Objekten robuste Methoden der Datenassoziation.

Diese Arbeit stellt einen vSLAM Algorithmus vor, welcher mit dynamischen Szenen in großen Umgebungen umgehen kann. Dafür wurde ein neuer Deskriptor - genannt Histogram of Oriented Cameras (HOC) - entwickelt, der anisotrope, räumliche Sichtbarkeit und die Wichtigkeit eines Kartenpunktes kodiert. Daher wird die Lokalisierung von dynamischen Elementen nicht beeinflusst und die erstellte Karte passt sich implizit ihrer veränderten Umgebung an. Um Effizienz und Skalierbarkeit während der Schleifenschließung zu steigern, werden die einzelne Kamera-Positionen in einem ungerichteten Graphen organisiert.

Um die vorgeschlagenen Methoden zu evaluieren wurde eine neuer Benchmark-Datensatz in einer sich ständig ändernden Umgebung aufgezeichnet. Das entwickelte SLAM-System wurde in einer Reihe von simulierten und realen Experimenten analysiert. Es wird gezeigt, dass wir mit dynamischen Veränderungen in der Karte umgehen können, die Lokalisierungsgenauigkeit verbessern und in der Lage sind die Anzahl an Punkten in einer Karte zu steuern.

Schlagwörter: digitale Bildverareitung, mobile Roboter, simultanes Lokalisieren und Kartenerstellen, Struktur durch Bewegung, dynamische Umgebungen, Datensatz, SLAM, optisches Lokalisieren und Kartenerstellen, Histogram orientierter Kameras

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

Acknowledgments

It is the commitment of many great people that brought me in the position of being able to finish my PhD studies.

Especially, I want to thank my supervisor Horst Bischof, who guided my research with his experience and inspiring comments. Despite of that he provided a pleasant and exciting working environment at the institute and organized barbecues at his hometown. Further, I wish to thank Konrad Schindler for agreeing to act as my second thesis supervisor and his encouraged and fruitful comments and discussions.

Moreover, I would like to give my most sincere thanks to my dear colleagues at ICG, namely Martin Lenz, Christof Hoppe, Markus Heber, Arnold Irschara, Christian Reinbacher, Michael Maurer and Manfred Klopschitz. Amusing intra- and extra-faculty discussions, collaborative Ü-Menü cooking, one or two bottles of after work beer and their friendship will always remind me of my PhD years.

I am especially indebted to the head of the robot vision lab Matthias Rüther. He mainly guided my research through excessive discussions, helpful suggestions and good thought-provoking impulses. Design and realization of this work would have never been possible without his tireless assistance in the last four years. Thank you!

Finally, I feel greatly indebted to my family and friends for their love and support: Mama, Papa, Lukas, Oma, Opa and Mirli, Friedl, Kathi, Hanna, Niki and Martina. Thanks for your patience and trust. Especially, I would like to accentuate the trust, love and support of my husband Jakob.

This work was supported by the Austrian Research Promotion Agency (FFG) and Federal Ministry of Economics, Family Affair and Youth (BMWFJ) within the Austrian research Studio Machine Vision Meets Mobility.

Contents

List of Figures

List of Tables

1 Introduction

Throughout the last decades robotic research has seen great progress in constructing human-like, self-operating and autonomous robotic systems with many fields of applications. They assist humans in everyday life, perform perpetual tasks in industrial environments or advance into environments which are dangerous or unreachable for human beings. Domestic robots have been constructed to perform basic household tasks such as vacuuming, mowing the lawn, wiping the floor or cleaning the pool. In the field of medicine surgery robots have become more and more popular, where minimal invasive surgery has reduced the tissue trauma and remote surgery has become possible. Mobile robots are found in military applications, space exploration, rescue scenarios, security environments, industry and care of the elderly. Rescue robots are applied in distressed areas dangerous for human beings. In scientific research, robots are used to explore hard reachable areas such as undersea or outer space. Healthcare robots either assist individual humans (e.g. smart wheelchairs) or perform repeatable tasks in pharmacies or hospitals (e.g. delivering drugs). Autonomous mobile robots or cars are able navigate on their own through everyday scenarios. Service robots take over service tasks such as serving coffee or drinks or guiding visitors through museums or exhibitions while performing path planning, obstacle avoidance or grasping. Autonomous driving cars safely navigate through urban traffic scenarios while detecting traffic signs, considering traffic rules or taking care of pedestrians. Figure 1.1 shows some robots whose applications vary from assistance robots to self-driving cars.

The design and development of intelligent, self-acting robots combines scientific achievements from different research areas such as artificial intelligence, robotics, sensor technology, machine learning, mathematics, physics and applied mechanics. However,

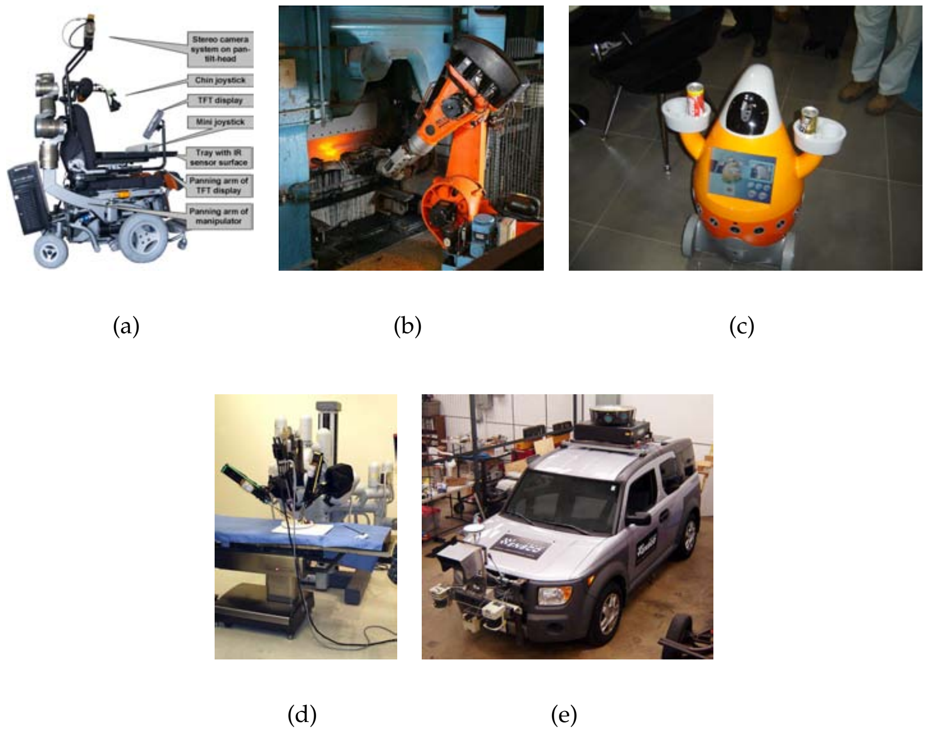

Stereo camera system on pan-tilt-head
Chin joystick
TFT display
Mini joystick
Tray with IR sensor surface
Panning arm of TFT display
Panning arm of manipulator
(a)
(b)
(c)
(d)
(e)

Figure 1.1: Different robot types. (a) Wheelchair with various kind of sensors on it for disabled or high-maintenance persons. (b) Articulated industrial robot. (c) Service robot offering drinks. (d) Multi-arm surgery robot. (e) Autonomously driving car.

to perform higher-level applications such as path planning or grasping a robot has to solve two fundamental key problems: *"Where am I?"* and *"How does the world look like?"*. In literature this problem is known as *Simultaneous Localization and Mapping*, often abbreviated as SLAM. Solving this problem involves constructing a map of a previously unknown environment while simultaneously localizing within this map. It can be seen as a chicken-or-egg problem as depicted in Figure 1.2: A robot equipped with one or more sensors perceiving the outside world is not able to estimate its position without the existence of a map. At the same time the map cannot be constructed from these sensor observations without the knowledge of the vehicle's pose. In simple terms, a robot requires a map to determine its position in the world, whereas knowing its pose is necessary to construct a map.

Generally, the SLAM problem can be visualized as a Bayesian network as depicted in Figure 1.3(a). The variables of interest, namely the sensor poses $\mathbf{P_i}$ and map points $\mathbf{X_j}$, are connected by sensor measurements m_{ij}. Unfortunately, both sensor measurements and robot pose estimates are perturbed by noise. Therefore, most successful solutions

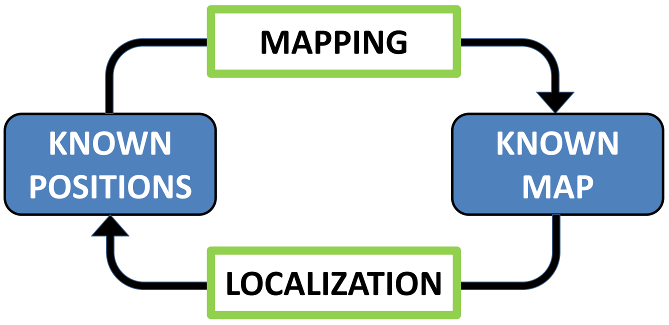
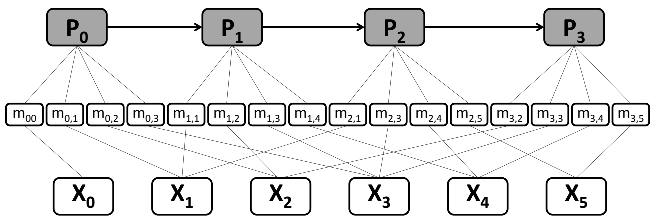


Figure 1.2: Simultaneous localization and mapping can be seen as chicken-and-egg problem. To estimate its pose a robot requires a map while at the same time the map can only be built given an accurate vehicle pose.

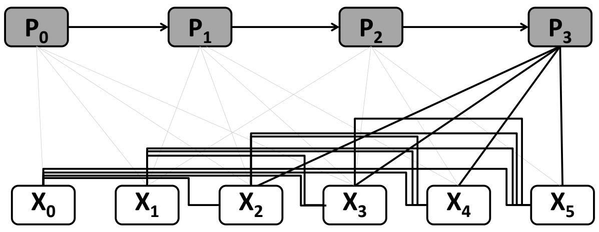
assume a Gaussian distribution for both sensor pose estimate and measurement noise. If one can find correspondences between different measurements (also known as data association) and is able to connect measurements throughout poses as shown in the graph, one can find a maximum likelihood solution to the pose and structure problem.

In the past, lots of methods have been developed to solve the SLAM problem using different kinds of sensors and achieved reasonable results. At the beginning research concentrated on wheeled robots driving in planar scenes and constructing two-dimensional representations of the environment using laser range finders or sonar sensors. Increased computing power and the development of more sophisticated sensor technologies such as cameras, pant-tilt laser scanners or time-of-flight cameras allow to build realistic, large-scale three-dimensional representations of the environment in acceptable time. Simultaneously, the fields of computer vision, robot vision and augmented reality have made great scientific progress regarding image processing, three-dimensional scene reconstruction, object tracking or realistic, real-time scene visualization. Besides, cameras can be found in many electronic devices such as smartphones, tablets or laptops and come along with many advantages: lightweight, widely-used, cheap, small and battery-saving. Furthermore, a single image contains much more information than data delivered from a laser range finder or sonar sensor for which reasons industrial or consumer cameras are becoming the preferred sensor in the robotics community.

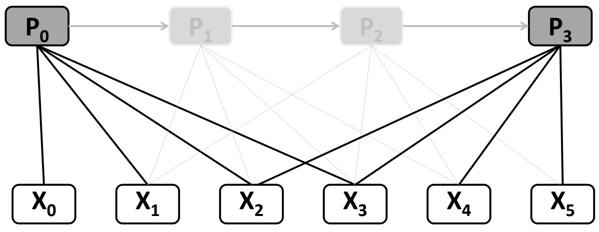
To solve the SLAM problem with vision sensors - also known as *visual SLAM* or *vSLAM* - two methodologies have emerged: batch optimization techniques such as bundle adjustment (BA) [137] minimize some form of image-based error to solve for camera pose parameters and 3D points, whereas recursive filtering techniques [25] update prob-



(a)



(b)



(c)

Figure 1.3: Network graphs for SLAM. (a) Bayesian network design for a SLAM problem. Sensor poses $\mathbf{P_i}$ and map points $\mathbf{X_j}$ need to be estimated. Those are linked through observations m_{ij} delivered by the sensor and form a network graph. (b) When performing EKF based SLAM old poses are marginalized out and their measurements are forwarded to the current state. (c) Marginalization when performing keyframe-based bundle adjustment is done by ignoring some poses and their measurements (courtesy of [128]).

ability distributions over map points and sensor poses. Historically the first research area originates from photogrammetry where structure from motion (SfM) techniques are applied to compute sensor poses and three-dimensional landmarks out of a collection of unordered images. The robotics community instead developed recursive filtering approaches (e.g. Extended Kalman Filters (EKF), Particle Filters) to compute map and pose estimates using an incremental image stream captured by a moving robot. Generally, both approaches pursue the same goal with different algorithmic frameworks.

Both approaches, SfM and EKF, continuously add poses, points, measurements and links to the network graph while exploring the environment. As a consequence the computational effort and storage requirements get larger with every incoming frame and only allow the exploration of a restricted area with a limited number of poses and landmarks. To overcome these limitations recursive filtering techniques marginalize out old poses while broadcasting their measurements. Unfortunately, this leads to a fully connected graph, which, when solved with an EKF causes a huge, intractable state vector and covariance matrix (compare Figure 1.3(b)). SfM techniques instead perform sliding window bundle adjustment over carefully chosen keyframes (compare Figure 1.3(c)), which outperform the EKF approach in terms of accuracy, scalability and speed [128].

1.1 Problem Definition

Most existing solutions to the visual SLAM problem presume the following assumptions:

- The robot is surrounded by a static, well-structured environment containing stationary objects only [63].
- The environment map and robot trajectory are estimated once and used for localization afterwards. Subsequent changes of the surroundings are not taken into account [39, 136].

However, most applications mentioned above (vacuum cleaner, service robot, tour guide robot, autonomous driving car) do not allow a static world assumption. Let's consider a domestic robot equipped with a vision sensor aiding in everyday life by serving drinks to different places in an apartment and vacuuming the entire flat. During operation roommates are moving around distorting its sensor readings throughout several frames. Moreover, humans may change the environment daily by disarranging furniture or adding new objects to the flat. The robot must be able to react on these changes to successfully accomplish its tasks. As a consequence, a static map would not be the method of choice since path planning, obstacle avoidance or grasping may fail. In addition, the SLAM algorithm must be able to process a huge amount of data over several months while executing its instructions online.

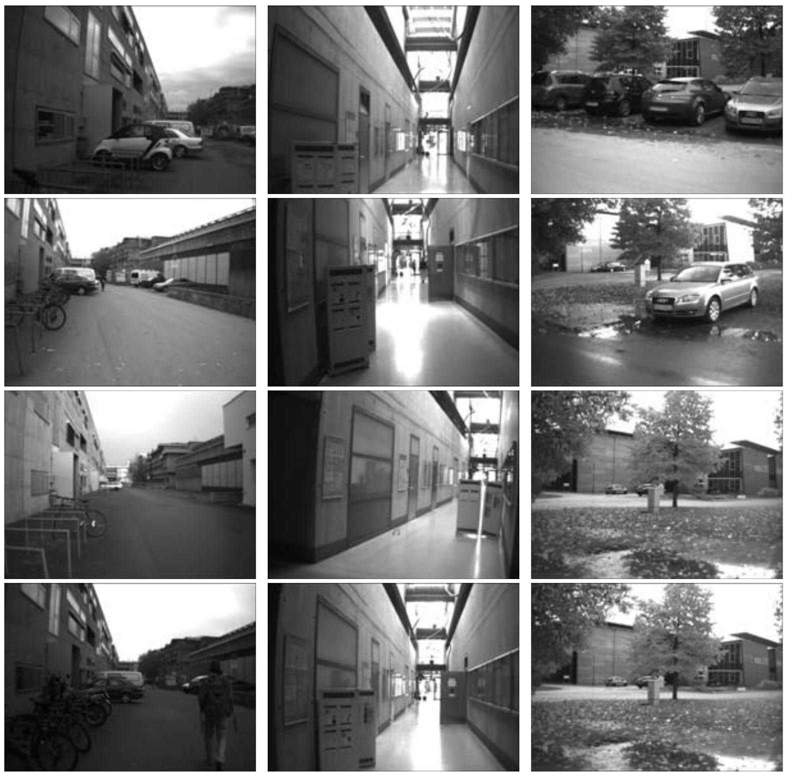


Figure 1.4: Challenging scenarios when performing visual SLAM. Short-term changes such as moving persons or driving cars lead to distorted sensor readings. Different lighting conditions, moved furniture or various weather situations pose a challenge for every camera-only SLAM algorithm.

Unfortunately, the scenario described above is not solvable up to now. While the SLAM problem within a static world is a well studied problem, the long-term localization and mapping task within an ever changing world remains an open problem in the field of robotic research. A few solutions have been presented tackling the problem of short-term noise (i.e. objects/people moving inside the robots field of view) by filtering out these spurious measurements from the sensor data or detecting and tracking of the objects. Other solutions focus on clustering or learning different environment configurations over time but do not provide a solution to update an existing map. Especially, when dealing with sparse local features stemming from visual sensors the change detection and map update problem is not solved at all. The key problem there is that information gained from map features which should be visible from a certain viewpoint, but are actually not observed is completely ignored. Most existing SLAM solutions simply incorporate all incoming sensor data into the map which leads to serious problems like increased storage requirements, inconsistent map representations and as a consequence incorrect data association. In highly dynamic scenes, static SLAM would inevitably fail in the long run.

Looking at Figure 1.4 a complete visual SLAM system operating over weeks or even months faces the following problems:

- Huge amount of incoming data to process.
- Increased computation time and storage requirements.
- Visitation of the same area multiple times.
- Motion blur or frame-loss during data acquisition.
- Various lighting and weather conditions.
- Objects move according to different time scales. Short-term noise originates from driving cars or moving pedestrians. Long-term changes arise from moved furniture or opened/closed doors.

To overcome these problems, a visual SLAM solution has to be designed which is able to robustly navigate through highly dynamic scenes while constructing and updating a three-dimensional map. Different solutions have been proposed to tackle the problem of moving people or driving cars by filtering or tracking these dynamic scene elements [143, 141, 146]. Other solutions focused on the representation of long-term maps to handle low-dynamic scene elements moved outside the robots field of view [4, 71]. However, continuous mapping over a long-period of time remains an open problem in the SLAM community. These considerations together with above mentioned problems lead to the following requirements:

- Permanently perform the localization and mapping task.
- Claim robustness against short-term changes.
- Detect and recover from localization failures.
- Continuously check for loop closures and correct them.

- Continuously check for map consistency and steadily repair and update an existing map.

1.2 Contributions

Compared to state-of-the-art in vision-based localization and mapping, the proposed vSLAM system contains two main contributions which allow a robot to robustly navigate through real-world scenarios containing high and low dynamic entities over a long period of time. We are able to continuously perform localization and mapping whilst permanently checking for loop closures. Short-term changes are filtered out during the localization task and the constructed map implicitly adapts to dynamic changes.

At first, a new three-dimensional feature descriptor called Histogram of Oriented Cameras (HOC) has been developed, which encodes anisotropic spatial visibility information and the importance of a three-dimensional landmark. Each feature holds and updates a histogram of the poses of observing cameras. It is hereby able to estimate its probability of occlusion and importance for localization from a given viewpoint. Through visibility-dependent map filtering we are able to delete vanished landmarks and guarantee an up-to-date map. Hence, short- and long-term dynamics do not affect localization and the constructed map implicitly adapts to dynamic changes during mapping.

Second, keyframes are organized in an undirected, unweighted view-graph, which facilitates sliding window bundle adjustment. The graph structure is further employed to speed up loop closure correction where only a subset of keyframes is determined to be corrected for rotation, translation and scale. As a consequence parameters to be estimated during loop closure correction can be diminished drastically, especially when navigating through large-scale scenes.

In addition, we investigated the difficulties and challenges when trying to benchmark a visual SLAM system in a dynamic environment setting. Considerations lead to the development of a new benchmark dataset denoted *CDBench* containing visual data recorded throughout several days in a mixed indoor/outdoor setting. The dataset comprises moving people, cars, different lighting and weather conditions and long-term changes such as removed and modified objects.

1.3 Outline

The thesis is organized as follows. A short historical overview on the simultaneous localization and mapping problem followed by an extensive review on state-of-the-art methods concerning visual SLAM and methods dealing with dynamic environments is given in Chapter 2. In Chapter 3, the theoretical foundations to understand visual localization and mapping are discussed. This includes topics like projective geometry, the perspective camera model as well as two- and multi-view geometry with special focus on bundle adjustment techniques. The basic concepts to perform visual localization and map building are described in Chapter 4. At the beginning we focus on features used for visual localization and map building followed by a detailed description of state-of-the-art concepts in localization, map building and global pose estimation. Finally, our visual SLAM framework is presented. Chapter 5 contains the theoretical descriptions of the new three-dimensional feature descriptor and the pose graph. We demonstrate how to incorporate them in our visual SLAM framework and highlight their benefits when compared to standard vSLAM systems. The remainder of this thesis includes the generation of a benchmark dataset with special focus on life-long operation and dynamic environments in Chapter 6. Our framework is then evaluated regarding localization and mapping accuracy, scalability and robustness on the presented dataset. We finally give a conclusion and an outlook to future work in Chapter 7.

2 Related Work

2.1 Simultaneous Localization and Mapping

2.1.1 A Bit of History

The first authors who followed a Bayesian formulation for robot pose estimation and showed how successive measurements of objects can be used to improve both robot and object location in a global world coordinate frame were Smith and Cheesman [119]. The survey paper by Durrant-Whyte et al. [46] first established the acronym SLAM and proved important convergence results. From that on several papers were published providing solutions to the mapping problem given known poses. Here, the pioneering work of Moravec and Elfes [88] using occupancy grid maps and Kuipers and Byun [67] performing topological mapping have to be mentioned. Early works in the field of robot localization have been presented by Leonard and Durrant-Whyte [70] or Lu and Milios [76]. Leonard and Durrant-Whyte presented an EKF-based localization algorithm within an existing map composed of geometric beacons. Lu and Milios instead aligned successive laser range scans to estimate the robot's pose.

From a historical point of view the most prominent algorithm performing SLAM using a Kalman Filter (KF) for pose and landmark estimation has been presented by Dissanayake et al. [30] using a millimeter-wave radar. Montemerlo et al. [85] invented the *FastSLAM* algorithm which forms the basis of many state-of-the-art solutions, where the robot pose is estimated with a Particle Filter (PF).

2.1.2 Recent Advances in Visual SLAM

Currently, most existing algorithms follow either a pure optimization based or recursive filtering approach to solve the visual SLAM problem. Filtering approaches update probability distributions over landmarks and sensor poses in a recursive manner, while optimization based approaches, also known as incremental Structure from Motion, use bundle adjustment techniques to estimate sensor poses and landmarks.

Davison and Murray [26] presented the first stereo visual SLAM system based on standard EKF. Later, Davison [25, 27] improved their approach to operate in real-time with a single camera named *MonoSLAM*. However, perspective projection of Euclidean points is a non-linear process and leads to inaccurate filtering results. Therefore, Montiel et al. [87, 17] proposed an inverse depth parametrization for EKF-based SLAM. However, EKF approaches are restricted to a small number of landmarks because of $O(n^2)$ space and computation costs. Consequently, many authors focused on complexity reduction by using an information filter [134] or by exploiting the sparsity structure of the information matrix [29]. Also Chli et al. [16] formed a tree-like hierarchical structure, where 3D point features are grouped into clusters from coarse (all grouped together) to fine (independent) to speed up probabilistic monocular SLAM. To allow monocular approaches to operate in large environments many authors [18, 100, 101] proposed to use submaps. In [18] Clemente et al. used *MonoSLAM* in each submap, where feature correspondences between submaps are established to correct scale drift and to join the individual maps. Paz et al. [100] joined the different submaps in a binary-tree fashion at fixed intervals reducing the computation time from $O(n^2)$ to $O(n)$. Later they improved their approach to work with both monocular and stereo information [101].

Contrarily, Eade and Drummond [31] or Elinas et al. [33] adapted the *FastSLAM* [85] idea to monocular sensor inputs based on a Rao-Blackwellized Particle filter and used a separate EKF for each three-dimensional landmark. To reduce the memory consumption of three-dimensional occupancy grid maps Marks et al. [77] diminished the map to 2D occupancy grids holding the height of a potential obstacle.

Recently, optimization based approaches have become more popular where incremental SfM techniques attracted attention. To achieve constant computation time BA is only performed over a fixed number of carefully selected keyframes also denoted as sliding window or active window [91]. The most prominent work in this field has been implemented by Klein and Murray [63], who split the tracking and mapping process into separate threads allowing for real-time single camera SLAM in small workspaces called *PTAM*. The tracking task (i.e. camera pose estimation relative to a sparse pointcloud) is handled by a simple motion model followed by feature point matching and n-point pose estimation [89]. The map building part generates three-dimensional landmarks through multi-view triangulation followed by bundle adjustment. They also fused their *PTAM* framework [56] with the relative BA approach proposed by Sibley et al. [114] to allow for loop closures. A similar system has been proposed by Clipp et al. [19] using a stereo-camera in a larger workspace. Here, loop closing is performed by applying BA over the whole map and trajectory, which is computationally very demanding when applied to a huge number of poses and points. Mei et al. [82] formed a graph of camera poses and landmarks connected by relative transformations and perform relative bundle adjustment of both structure and motion based on [114]. To efficiently handle loop closures Lim et al. proposed a hybrid map representation [**?**] consisting of locally metric maps containing poses and points and a topological map made of keyframes only. Hereby, local maps are optimized through bundle adjustment techniques while the topological nodes are optimized separately. Contrary, Strasdat et al. [126] suggested to optimize metric and topological maps simultaneously using a double window approach. The inner window is presented as point-to-frame constraints optimized through BA. The outer window is defined by frame-to-frame constraints, which are optimized through their scale drift aware pose graph optimization routine [127]. To limit the parameters to be estimated Konolige and Agrawal [65] merged consecutive poses and associated features in a probabilistic manner. The reduced graph is optimized using the TORO [45] framework.

Taking sparse feature maps and camera poses produced by real-time SfM algorithms as input, Newcombe and Davison [92] computed dense environment models. Hereby, GPU-based optical flow is used to estimate pixel-wise correspondences between keyframes resulting in local dense reconstructions which are merged in a subsequent step. They extended their approach without relying on sparse features and keyframes named *DTAM* [93]. Here, photometric information of several monocular RGB images is fused into a single cost volume to estimate a depth map for selected keyframes through a non-convex optimization framework. Depth values stored in each keyframe are then used to compute a dense representation of the environment. The resulting map allows to track the camera at frame rate by whole image registration.

With the launch of range image devices providing 2.5D data (e.g. Microsofts Kinect) large scale dense reconstruction of indoor environments has been proposed by Henry et al. [54]. An advanced iterative closest point (ICP) variant is used to robustly estimate interframe motion while realistic environment modeling is handled by Surfels

[102]. Contrary, Newcombe et al. [60] fused all incoming depth data into a single global implicit surface model using a truncated signed distance function. They successfully showed a dense representation of a desktop scene. The sensor pose is estimated by registering the depth map relative to the whole surface using a hierarchical ICP algorithm. At the time of writing dense mapping techniques were restricted to small-scale scenes only [92, 93, 60, 43, **?**] or used a pseudo-dense representation [54] to operate in larger areas.

2.2 SLAM in Dynamic Environments

Research tackling the problem of localization and map building within a non-static environment is sparse - especially when using vision sensors. Therefore, we also want to highlight some approaches taking laser or sonar sensor as input. Generally, two lines of research have emerged: Those focusing on short-term changes within crowded environments containing moving people, driving cars or bicycles. Others, concentrating on less frequent changes like moved furniture, opened or closed doors within a long-term mapping context. A tabular summary of available methods is also presented in Table 2.1.

Referring to the first group Wang et al. [145] proposed a SLAM framework combined with an object detection and tracking mechanism to filter out the dynamic parts of the scene, also known as SLAM-DATMO. They are using a motion-based moving object detector [143] and a Bayesian formulation for object tracking. They presented results using a vehicle in a road scenario equipped with two laser range finders. In [144, 141] they improved their framework regarding computation time and accuracy by using two separate probabilistic filters for moving object tracking and the localization and mapping task. Additionally, two object detection algorithms were presented classifying moving and stationary objects in laser scan data where the dynamic scene elements are stored in a local occupancy grid map for each moving object separately. This probabilistic information is fed back to the SLAM process, where a global feature based map is built.

Montemerlo et al. [86] used a particle filter to estimate both the robot pose and the position of persons in a previously mapped environment. A larger particle set is used for person tracking which is conditioned on the estimated robot pose represented by a smaller set of particles. Hence, the number of particles depends on the number of dynamic objects currently observed by the sensor. Without relying on a previously built map Lidoris et al. [72] estimate the position and velocity of all moving entities together with the robot pose using the same approach as [86], where a moving person in 2D laser data is modeled as a cylindrical object. Both algorithms [86, 72] cannot handle arbitrary moving targets since their detection algorithm is restricted to the shape of a person. Contrary, Miller and Campbell [83] used a KF to track moving objects within a map built of stationary objects. To robustly solve the data association problem a particle filter is used whose discrete output is utilized by the tracking task. The proposed factorization allows for an automatic detection of dynamic objects in the map without increasing the particle set as in [86].

A purely monocular approach has been developed by Marzorati et al. [24] who use two separate EKFs for the static and dynamic parts in the scene. They proposed to use uncertain projective geometry to detect dynamic elements. Results are provided within a small office scene using a few features only. Wangsiripitak and Murray [146] are using a 3D object tracker within the *MonoSLAM* framework [27] to detect moving objects and occlusions. Unfortunately, all monocular based approaches are limited to office environments. A stereo-based approach named SLAMMOT was proposed by Lin and Wang [73], where a binary Bayes filter and the inverse depth parametrization are integrated into a decision tree to perform moving object detection. Both robot pose and moving object trajectory are estimated by an EKF. They showed that they outperform monocular SLAMMOT approaches because the stereo-system solves the limited observability and also increases the accuracy of the localization. Similarly, Solá [122] invented BiCam-SLAM with a rule-based moving object detection method where object tracking is done separately and individually by a KF in a robocentric representation.

Many authors built a map out of stationary objects while getting rid of the dynamic elements. The earliest work in this direction has been presented by Fox et al. [40] who developed a probabilistic filter which only uses measurements stemming from stationary objects. Therefore, they enhanced the traditional inverse sensor model [133] to capture the probability a measurement is originating from an obstacle not contained in the map. Their framework has been evaluated during a localization task in highly populated environments using a prebuilt map. Hähnel et al. [48] incorporated the results of a people tracker into a probabilistic filter to get rid of these spurious elements in laser sensor data. Without incorporating these dynamic elements data association, scan alignment, localization and map building becomes more accurate. They evaluated their algorithm in a large-scale environment where several people were moving near the robot. They further improved moving people detection by developing an Expectation-Maximization (EM) algorithm [49]. The filtered laser data is again used to create an occupancy grid of the static scenery. Both methods do not update the map adaptively and only stationary objects are present in the map. Huang et al. [58] combined sonar and vision data to differentiate between dynamic and stationary objects. The temporal difference between subsequent sonar readings and a statistical background subtraction technique applied to camera images allowed them to detect moving objects and exclude them from the mapping process.

Baig et al. [7] incorporate information stemming from an occupancy grid made of stationary data to detect moving objects in the scene. If the ray of laser data ends in a grid cell previously seen as free space then the beam must correspond to a non-stationary object which is then tracked by a global nearest neighbor method and stored in a separate map. Rays ending in a filled grid are considered to be static and used to update the map. One drawback of their method is that a local grid map containing stationary objects is required to allow moving object detection.

The second group concentrates on long-term localization and mapping, where many authors focused on learning the robustness and strength of different landmarks [2, 55] or to robustly discriminate between static and dynamic scene elements [4, 5]. Andrade-Cetto and Sanfeliu [2] combined appearance properties and strength states to learn the robustness of each landmark. This measure of importance is used within an EKF for state estimation. Furthermore, this quality measure allows them to delete unreliable features from the map. Hochdorfer and Schlegel [55] addressed the problem of ever growing number of landmarks within a feature based map, especially in life-long operations. To avoid extensive growing of the EKF state-vector, they limit the number of allowed landmarks in a two stage process: First, k-means clustering combined points which are observed from neighboring robot poses. Second, landmarks with the lowest localization benefit within each cluster, estimated out of their covariances, are removed. To differentiate between static and dynamic objects Anguelov et al. [4] used an EM algorithm to learn a model of non-stationary objects from a sequence of occupancy grid maps stemming from different points in time. Unfortunately, their algorithm requires three prerequisites: Objects must be uniquely identifiable by their shape, they should not move during mapping and should be spaced enough from each other. To overcome these problems they proposed to combine an omni-directional camera with a laser range finder [5] to capture color, motion, appearance and shape of stationary (walls) and non-stationary (doors) objects. They showed that the algorithm greatly benefits from the supplemental information stemming from the vision sensor. Stachniss and Burgard

[124] instead tried to learn different configurations of a grid map stored in several local sub maps using a laser range finder. To estimate the configurations of dynamic areas they performed clustering of local grid maps assuming that a limited number of clusters is sufficient to model the scene configurations (e.g. two clusters would model an opened or closed door).

To capture the environmental changes over a longer period of time many researchers incorporate additional information into traditional two-dimensional occupancy grids. Arbuckle at al.[6] invented temporal occupancy grids, where each grid cell captures the probabilities at different timescales. The stored probabilities are exploited to differentiate between fixed obstacles (those having high values at all timescales) and moving obstacles (those having high values on short timescales only). Similarly, Mitsou and Tzafestas [84] store the whole history of sensor readings in each cell using the so called TimeIndex [35], which is organized as a B-tree for fast access operations. To check for dynamic objects they exploit the standard deviation of the probabilties stored in each grid cell, where a small deviation indicates a static cell, while a high variance represents a dynamic object. However, both grid structures are only used to identify low- or high-dynamic objects but do not adapt the resulting grid map. More recently, Levinson and Thrun [71] proposed to store the variances of the sensor data in each grid cell. As a consequence they are able to identify dynamic scene elements during localization and prefer those map parts more likely to be stationary. They showed successful navigation superior to GPS with an autonomous driving vehicle during rush-hour traffic. Instead of storing additional information in the occupancy grid structure many authors use two or more traditional grids. Wolf and Sukhatme [150] maintain one grid map for the static and the dynamic map components each. In order to verify which laser sensor reading can be regarded as dynamic, they make use of a recursive Bayes filter dependent on the previous grid map states.

The most complete systems operating over a long period of time and trying to adapt an existing map to the most recent environmental changes have been presented by [8, 28, 66]. Biber and Duckett [8] create occupancy grid maps from laser data at different timescales to incorporate new elements while preserving the old and stable ones. Hereby, each incoming laser reading is compared to all timescales and the one best fitting the data is used. They evaluated their method regarding localization accuracy over several weeks. Dayoub and Duckett [28] focused on map adaption within an existing topological map made of omni-directional images. They are using the concept of long-term and short-term memory, where persistent features extracted from the images are added to the map whereas older ones are removed. A rehearsal algorithm is presented to select stable features moved to the long-term memory while a recall algorithm is responsible for feature deletion. They evaluated their algorithm in a long-term experiment over nine weeks monitoring a canteen scene and showed improvements in localization against static approaches. Unfortunately, the topology of the initial map is never changed and remains fixed. Similarly, Konolige and Bowman [66] adapted *FrameSLAM* [65] to update a given map in case of new or removed features and to recover from localization failure. They first build a connectivity graph between keyframes, based on the number of successful SIFT features, and delete those keyframes with a very high SIFT matching percentage to the neighbors. They evaluated their system in a dynamic indoor environment of about $50 \times 50m^2$, including moving people and various lighting conditions. They successfully managed to update a map after removed and added furniture and kept the number of keyframes relatively small.

2.3 Conclusion

This chapter provided a short historical overview of the SLAM problem followed by an extensive review of state-of-the-art algorithms especially in vision based SLAM. The main focus lied on localization and mapping techniques dealing with dynamic environments in a life-long mapping context. Here, many authors are using two-dimensional range scanners. Those algorithms who make use of a vision sensor are restricted to operate in small-scale areas. All vision based methods either use object detection and tracking to separate moving objects from the static map or perform spatial clustering in combination with heuristics to discard weak features. Few of them have the ability to repair, change or update the previously constructed map. Instead of that many authors augment a map with an additional timing variable or try to learn different environment configurations.

Table 2.1 provides a compact comparison of SLAM algorithms operating in dynamic environments by means of handling high or low dynamic scene elements or being able to update and repair an existing map. We also highlighted our solutions to the SLAM problem being capable to deal with all challenges when traveling in highly dynamic scenes over a long period of time.

Approach	continuous SLAM	high dynamics	low dynamics	removing data	adding data	map required
Anguelov et al. [4]	×	×	✓	×	×	✓
Montemerlo et al. [86]	×	✓	×	×	×	✓
Wang et al. [145, 141, 73], Lidoris et al. [72], Miller et al. [83],Hähnel et al. [48]	×	✓	×	×	✓	×
Daniele et al. [24]	×	✓	✓	×	✓	×
Baig et al. [7]	×	✓	✓	×	✓	✓
Arbuckle et al. [6], Levinson et al. [71], Biber and Duckett [8]	✓	✓	✓	×	✓	×
Andrade-Cetto et al. [2]	✓	×	✓	✓	✓	×
Dayoub et al. [28]	✓	✓	✓	×	✓	✓
Konolige and Bowman [66]	✓	✓	✓	✓	✓	×
Ours [103, 104]	✓	✓	✓	✓	✓	×

Table 2.1: Comparison of SLAM algorithms operating in dynamic environments. We checked if they are able to handle high (e.g. moving people, driving cars) and low (e.g. moved or added furniture, plants) scene dynamics as well as their ability to react on environmental changes and to update the existing map by removing vanished objects or add new landmarks to the map. Some algorithms need to build a map out of static components only to detect dynamic entities later on (map required). In contrast to previous approaches our solution handles both static and dynamic objects as well as map updates in a life-long mapping context without relying on a previously built map.

3 Theory and Background

A visual SLAM algorithm operating on a robot equipped with one or more cameras should create a three-dimensional map of its unknown surrounding while moving and localizing within this map using visual sensor information only. Therefore, we require knowledge about the imaging process, the extraction of three-dimensional information out of image data and the geometric relationship between two or more cameras and the three-dimensional world. This chapter provides the theoretical background and a mathematical framework in order to understand the visual localization and map building tasks.

First, the imaging process and its involved geometric entities are reviewed in Section 3.1. This involves the description of the two- and three-dimensional projective space and their geometric transformations. We also explain different rotation representations because of their importance in parameter estimation techniques later on. Finally, the imaging process and its geometric properties by means of the general projective camera is described. Section 3.2 concentrates on the geometry between two or more images including a detailed description of the geometric relations between two views and the estimation of three-dimensional structure out of them. In large part we concentrate on structure from motion (SfM) and its mathematical framework, which describes the process of three-dimensional data estimation out of multiple views.

3.1 Projective Geometry

The imaging process, namely the mapping of a three-dimensional scenery to two-dimensional images, is described elegantly by projective geometry. We focus on the geometry and its entities in the two- and three-dimensional projective space $\mathbb{P}^2$ and $\mathbb{P}^3$, since the imaging model is described by a mapping from $\mathbb{P}^3$ to $\mathbb{P}^2$. Especially when performing visual SLAM one has to understand how images are built and the geometric properties between the world and the image representing it. More attention is given to different rotation representations, since they play an important role in parameter estimation techniques such as bundle adjustment. Finally, the anatomy of the general projective camera with its geometric entities is described. For a more detailed description on projective geometry, their main geometric ideas and various camera models we refer the reader to [53], [34] and [153].

3.1.1 The Two-dimensional Projective Space

A point (x, y) in the Euclidean plane $\mathbb{R}^2$ is represented as homogeneous point $(X, Y, 1)$ in the projective space $\mathbb{P}^2$, by simply adding a 1 as a third coordinate. Since overall scaling is irrelevant (WX, WY, W) is the same point as $(X, Y, 1)$. The transformation from projective points to Euclidean ones is given by $(X/W, Y/W) = (x, y)$. Similarly, a line is defined by

$$ax + by + c = 0 \qquad \in \mathbb{R}^2 \tag{3.1}$$

$$\tag{3.2}$$

where $\mathbf{u} = (a, b, c)$ denotes the line and $\mathbf{p} = (X, Y, W)$ a point on the line. An important property is, that the roles of points and lines can be interchanged - the so called duality principle. As a consequence the intersection of two lines or the line through two points are both defined by the vector cross product. All mentioned properties are summarized in Table 3.1.

Property	Algebraic formulation
Homogeneous point	$\mathbf{x} = (X, Y, W)$
Homogeneous line	$\mathbf{l} = (a, b, c)$
Euclidean point	$\tilde{\mathbf{x}} = (X/W, Y/W)$
Intersection of two lines	$\mathbf{x} = l_a \times l_b$
Line defined by two points	$\mathbf{l} = x_a \times x_b$
Line normal	$\mathbf{n} = (a, b)^T$

Table 3.1: Algebraic properties of the two-dimensional projective space $\mathbb{P}^2$.

3.1.2 The Three-dimensional Projective Space

All concepts of $\mathbb{P}^2$ can be extended to the projective space $\mathbb{P}^3$. Points and planes, as their brothers points and lines in $\mathbb{P}^2$, are represented as four-vectors (X, Y, Z, W) and (a, b, c, d). Also their intersection, plane equation and back-transformation to the Euclidean space $\mathbb{R}^3$ are defined the same way as in $\mathbb{P}^2$ and summarized in Table 3.2.

Property	Algebraic formulation
Homogeneous point	$\mathbf{x} = (X, Y, Z, W)$
Homogeneous plane	$\mathbf{l} = (a, b, c, d)$
Euclidean point	$\tilde{\mathbf{x}} = (X/W, Y/W, Z/W)$
Intersection of three planes	$\begin{bmatrix} \mathbf{l_a}^T \\ \mathbf{l_b}^T \\ \mathbf{l_c}^T \end{bmatrix} \mathbf{x} = 0$
Plane defined by three points	$\begin{bmatrix} \mathbf{x_a}^T \\ \mathbf{x_b}^T \\ \mathbf{x_c}^T \end{bmatrix} \mathbf{l} = 0$
Plane normal	$\mathbf{n} = (a, b, c)^T$

Table 3.2: Algebraic properties of the three-dimensional projective space $\mathbb{P}^3$.

3.1.3 Geometric Transformations

Generally, a projective transformation describes an invertible, linear mapping $h : \mathbb{P}^n \to \mathbb{P}^n$. Since it preserves the collinearity of points it is also called a collineation. Acting on a homogeneous point $\mathbf{x} \in \mathbb{P}^n$, a projective transformation can be written as simple matrix left-multiplication by a non-singular $(n + 1) \times (n + 1)$ matrix H, called Homography $\mathbf{x}' = H\,\mathbf{x}$.

The most important specializations of projective transformations are isometries, similarity transformations and affine transformations, which are briefly described in the following sections. In fact, projective transformations are applied in many fields of computer vision and 3D reconstruction: mapping between image planes (e.g. removing lens distortion), transformation of points and lines, camera projection or multi-view geometry (e.g. trifocal tensors, frame transformations). As shown in Figure 3.1 they form a hierarchy within the projective group. We also describe their composition and invariant

	Euclidean	similarity	affine	projective
Transformations				
rotation	X	X	X	X
translation	X	X	X	X
uniform scaling		X	X	X
non-uniform scaling			X	X
shear			X	X
perspective projection				X
Invariants				
length	X			
angle	X	X		
ratio of lengths	X	X		
parallelism	X	X	X	
incidence	X	X	X	X
cross ratio	X	X	X	X

Table 3.3: Hierarchy of transformations. We show their allowed geometric transformations and the measures that remain invariant in every group.

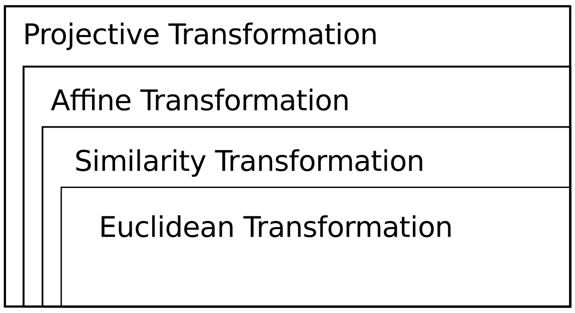


Figure 3.1: Hierarchy of transformations. Geometric transformations form a hierarchy within the projective group dependent on their inner-group invariants and allowed transformations.

properties which are summarized in Table 3.3. All transformations and their matrix representation are given in Table 3.4.

Isometries

Isometries (metric transformations, Euclidean transformations) are composed of a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$. These transformations preserve Euclidean distances and model rigid body motions (relative movements in $\mathbb{P}^3$). A planar isometry in $\mathbb{P}^2$ has 3 degrees of freedom (DOF); one for rotation and two for translation. In $\mathbb{P}^3$ a metric transformation has 6 DOF.

The two-dimensional rotation matrix is defined the following way:

$$R(\phi) = \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix}. \tag{3.3}$$

A general three-dimensional rotation is composed of the three orthogonal rotation matrices around the coordinates axes:

$$\begin{aligned} R(\phi,\theta,\psi) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{pmatrix} \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix} \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} c(\theta)c(\psi) & -c(\phi)s(\psi) + s(\phi)s(\theta)c(\psi)) & s(\phi)s(\psi) + c(\phi)s(\theta)c(\psi) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\phi)s(\theta)s(\psi) & -s(\phi)c(\psi) + c(\phi)s(\theta)s(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{pmatrix} \end{aligned} \tag{3.4}$$

where $s(.)$ and $c(.)$ denote the sine and cosine function. A more detailed discussion on rotations matrices and their different parameterizations is given in Section 3.1.4.

Similarity Transformation

A similarity transformation extends the metric transformation with an additional isotropic scaling factor. The scaling factor adds one additional degree of freedom in $\mathbb{P}^2$ (4 DOF) and $\mathbb{P}^3$ (7 DOF). It is also known as *equi-form* transformation because of its shape preserving nature.

Affine Transformation

The affine transformation is composed of a non-singular linear transformation **A** followed by a translation **t**. Hereby, the affine matrix is represented by rotations and non-isotropic scaling D

$$\mathbf{A} = R(\theta)R(-\phi)DR(\phi) \tag{3.5}$$

$$D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \text{ or } \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix} \tag{3.6}$$

leading to a total of 6 DOF (two for both rotations, two for translation and two for non-isotropic scaling) in $\mathbb{P}^2$ or 12 DOF in $\mathbb{P}^3$.

Projective Transformation

The projective transformations are more general than their affine counterpart since the fourth matrix row does not contain the null-vector. In contrast to affine transformations, it behaves non-linear when operating on inhomogenous points. The additional non-null vector leads to a total of 8 DOF (in $\mathbb{P}^2$) or 15 DOF (in $\mathbb{P}^3$).

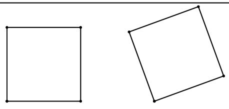
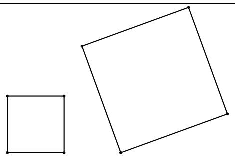
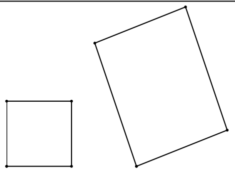
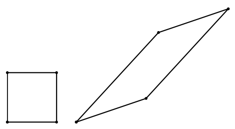
Group	Matrix	Distortion	$\mathbb{P}^2$	$\mathbb{P}^3$
Isometry	$\begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}$		3DOF length, area, angle	6DOF length, volume
Similarity	$\begin{pmatrix} sR & t \\ 0^T & 1 \end{pmatrix}$		4 DOF ratio of length and angle	7 DOF absolute conic
Affinity	$\begin{pmatrix} A & t \\ 0^T & 1 \end{pmatrix}$		6 DOF parallel lines, ratio of line segments and area	12 DOF parallel planes, ratio of volumes, cetroids
Projective	$\begin{pmatrix} A & t \\ v^T & s \end{pmatrix}$		8 DOF collinearity, cross ratio of collinear points, concurrency	15 DOF intersection and tangency of surfaces

Table 3.4: Geometric transformations. The number of degrees of freedom (DOF) as well as their invariants are described for $\mathbb{P}^2$ and $\mathbb{P}^3$. The matrix representation of the appropriate transformation H is also given [53].

3.1.4 Rotation Representations

As shown in Equation 3.4 a rotation matrix in $\mathbb{P}^3$ is represented by a 3×3 orthogonal matrix composed of three rotations around the principal axes, where the three parameters define the rotation around each axis. A rotation matrix R comes along with orthogonality and norm constraints:

$$\begin{aligned} det(R) &= 1 \\ R^{-1} &= R^T \\ \|Rv\| &= \|v\|. \end{aligned} \tag{3.7}$$

While the nine parameter encoding is useful for coordinate and frame transformations, it has disadvantages in many parameter estimation applications like bundle adjustment. Therefore, we focus on simpler representations consisting of three or four parameters like Euler angles, quaternions and the axis-angle representation.

Euler Angles

Any rotation from one orthogonal coordinate system to another can be described by three successive rotations parametrized by the Euler angles (θ, ϕ, ψ). Each angle describes a rotation about a single coordinate axis, where we focus on the most popular out of 12 rotation orderings called ZYZ or Y convention. Hereby, an initial coordinate system xyz is transformed the following way:

1. Rotation by angle ψ around the z-axis resulting in $x'y'z'$.
2. Rotation by angle ϕ around the y′-axis resulting in $x''y''z''$.
3. Rotation by angle θ around the z″-axis.

The final rotation matrix R is computed by matrix multiplication of three matrices $R_z(\psi)$, $R_{y'}(\phi)$ and $R_{z''}(\theta)$ each describing a rotation about a single axis.

$$\begin{aligned} R &= R_z(\psi) \cdot R_{y'}(\phi) \cdot R_{z''}(\theta) \\ &= \begin{pmatrix} c(\psi)c(\phi)c(\theta) - s(\psi)s(\theta) & -c(\psi)c(\phi)s(\theta) - c(\theta)s(\psi) & c(\psi)s(\phi) \\ s(\psi)c(\phi)c(\theta) - c(\psi)s(\theta) & -s(\psi)c(\phi)s(\theta) + c(\psi)c(\theta) & s(\psi)s(\phi) \\ -s(\phi)c(\theta) & s(\phi)s(\theta) & c(\phi) \end{pmatrix}. \end{aligned} \tag{3.8}$$

The inverse mapping is given by

$$\begin{pmatrix} \psi \\ \phi \\ \theta \end{pmatrix} = \begin{pmatrix} arctan\left(\frac{r_{23}}{r_{13}}\right) \\ arctan\left(\frac{\sqrt{r_{13}^2+r_{23}^2}}{r_{33}}\right) \\ arctan\left(-\frac{r_{32}}{r_{31}}\right) \end{pmatrix}, \tag{3.9}$$

where r_{ij} denotes the matrix element or R in the i-th row and j-th column. If $\phi = 0$ or $\phi = k\pi$ a double rotation about identical z-axes would occur, which is also known as gimbal lock. Figure 3.2(a) shows the three axis rotations evolved in a ZYZ convention.

Unit Quaternions

A unit quaternion is a four parameter vector $q = (q_1, q_2, q_3, q_4)$ obeying the constraint that

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1. \tag{3.10}$$

The four parameters are defined as

$$\begin{aligned} q_1 &= e_x sin(\frac{\theta}{2}) \\ q_2 &= e_y sin(\frac{\theta}{2}) \\ q_3 &= e_z sin(\frac{\theta}{2}) \\ q_4 &= cos(\frac{\theta}{2}), \end{aligned} \tag{3.11}$$

where e_x, e_y, e_z denote the Euler axes and θ the rotation angle. The fourth parameter is also called the scalar term, which can be calculated from q_1, q_2, q_3 through Equation 3.10.

The conversions between a rotation matrix R and quaternions are as follows:

$$R = \begin{bmatrix} 2(q_4^2 + q_1^2) - 1 & 2(q_1q_2 - q_4q_3) & 2(q_1q_3 + q_4q_2) \\ 2(q_1q_2 + q_4q_3) & 2(q_4^2 + q_2^2) - 1 & 2(q_2q_3 - q_4q_1) \\ 2q_1q_3 - q_4q_2 & 2(q_2q_3 + q_4q_1) & 2(q_4^2 + q_3^2) - 1 \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} sgn(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ sgn(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ sgn(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \\ \sqrt{r_{11} + r_{22} + r_{33} + 1} \end{bmatrix}, \tag{3.13}$$

where r_{ij} denotes the matrix element of R in the i-th row and j-th column. Their usage in computer vision and robotics is very popular because of their compact representation, the absence of discontinuous jumps and singularities and the reduced computational cost.

Axis-Angle Representation

In the axis-angle representation, also called exponential coordinates, a rotation is expressed by two parameters: the rotation angle θ and a unit vector $v \in \mathbb{R}^3$ indicating the axis of rotation. This four parameter representation can also be converted to a three-parameter case where rotation axis and angle are encoded by a non-normalized vector, where the angle is given by its magnitude. The conversion between rotation matrix R to its axis-angle counterpart (v, θ) with $\|v\| = 1$ is given by the Rodrigues formula (also called the exponential map):

$$R(v, \theta) = I + sin(\theta)\,[v]_\times + (1 - cos(\theta))\,[v]_\times^2, \tag{3.14}$$

where $[v]_\times$ denotes the antisymmetric matrix of v. Its conversion (also denoted as logarithmic mapping) is given by

$$\theta = arccos\left(\frac{trace(R) - 1}{2}\right) \tag{3.15}$$

$$v = \frac{1}{2sin(\theta)} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}. \tag{3.16}$$

This type of rotation representation is sketched in Figure 3.2(b).

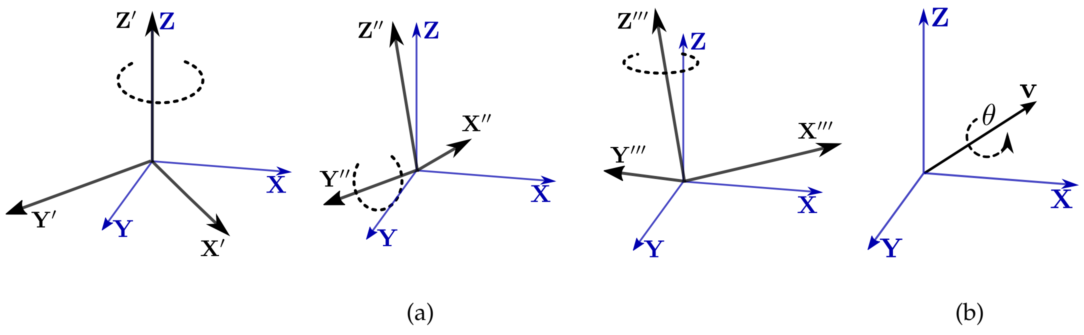


Figure 3.2: Rotation representations. (a) Three rotations around one of the main coordinate axes parameterized by the Euler Angles. Here the ZYZ or Y-convention is shown. (b) The axis-angle representation consists of a unit vector v denoting the axis of rotation and the angle θ its amount.

3.1.5 Perspective Camera Model

A camera defines a mapping from a three-dimensional point in space onto the two-dimensional image plane, which can be easily formulated by projective geometry. The camera coordinate system with its origin in the camera center C is defined as follows: The viewing direction of the camera is the positive z-axis, also known as principal axis or optical axis. The principal plane intersects with the xy-axis whereas the image plane is located parallel to it at distance f. Assuming a basic pinhole camera model, an image point $\mathbf{x} = (u, v)$ is defined by the intersection of the ray through a point in space $\mathbf{X} = (X, Y, Z)^T$ and the projection center C with the image plane.

Looking at Figure 3.3 this can be expressed as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix} \tag{3.17}$$

using similar triangles (central perspective projection). Taking homogeneous coordinates into account Equation 3.17 can be rewritten as simple matrix multiplication

$$\begin{aligned}
\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} &= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \\
&= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times 3} | 0_{3\times 1} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} . \qquad (3.18)
\end{aligned}$$

In practice the origin of the image frame is not at the principal point $\mathbf{p} = (p_x, p_y)$ (the intersection of the optical axis and the image plane) as assumed in Equation 3.18, but translated to the upper left pixel for example. This leads to the principal point offset (p_x, p_y) expressed in the following Equation:

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times 3} | 0_{3\times 1} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} . \qquad (3.19)$$

The first 3×3 matrix is called camera calibration matrix $\mathbf{K}$, which holds the intrinsic camera parameters. Together with the 3×4 matrix $[I_{3\times 3} | 0_{3\times 1}]$ this results in the homogeneous camera projection matrix $\mathbf{P}$, which describes the mapping from a homogeneous world point $\mathbf{X}$ to a homogeneous image point $\mathbf{x}$ by matrix multiplication

$$\begin{aligned}
\mathbf{x} &= \mathbf{K}\ [I_{3\times 3} | 0_{3\times 1}]\ \mathbf{X} \\
&= \mathbf{P}\ \mathbf{X}. \qquad (3.20)
\end{aligned}$$

The above defined camera calibration matrix $\mathbf{K}$ assumes quadratic pixels. In the rare case of sheared pixels an additional skew parameter s may be added leading to the general form of a camera calibration matrix

$$\mathbf{K} = \begin{pmatrix} f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{pmatrix} . \tag{3.21}$$

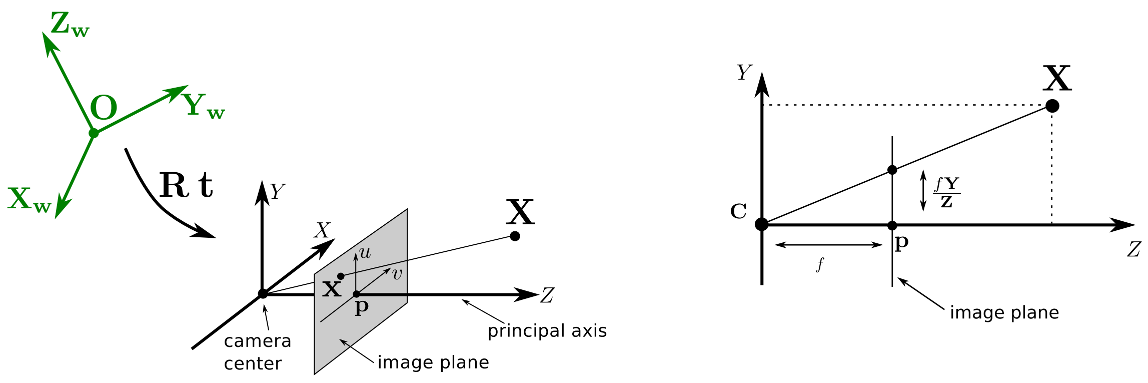


Figure 3.3: Perspective camera model. (left) A camera coordinate frame lying in a common world coordinate frame **O**. The extrinsic parameters **R** and **t** (representing an Euclidean transformation) describe pose and orientation of the camera in the world coordinate frame. (right) Pinhole camera geometry, where C is the camera center and **p** the principal point. Focal length f denotes the perpendicular distance between image plane and projection center C. The mapping from a world point X onto the image plane can be described by similar triangles.

Generally, three-dimensional points are defined within a common (Euclidean) world coordinate frame. The camera model defined so far is called *central perspective camera*, which assumes that the camera coordinate frame coincides with the world coordinate frame. In order to describe the orientation of the camera coordinate frame at a different pose an Euclidean transformation is used. Equation 3.20 can be extended to

$$\begin{aligned} \mathbf{x} &= \mathbf{K}\,[I_{3\times3}|0_{3\times1}] \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} \mathbf{X} \\ &= \mathbf{K}[\mathbf{R}|\mathbf{t}]\,\mathbf{X} \end{aligned} \tag{3.22}$$

using **R** and **t** to transform a world point **X** to the camera coordinate frame before projection takes place. Since **R** and **t** describe the exterior orientation of a camera they are called the *external camera parameters*. The inverse of the Euclidean transformation can be used to compute the camera center $C = (0,0,0,1)^T$ in the world coordinate frame as follows:

$$C_w = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix}^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = -\mathbf{R}^{\mathbf{T}}\,\mathbf{t}. \tag{3.23}$$

Putting all together the general projection matrix **P** defines a linear mapping from the projective space $\mathbb{P}^3$ to the image plane $\mathbb{P}^2$ with eleven DOF (3 for R, 3 for t and 5 for K)

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]\,. \tag{3.24}$$

The above described imaging model is assumed to be noise free. In practice many devices come along with lens distortion effects due to manufacturing errors in the lenses (i.e. straight lines in the scene are not imaged straight). The most common types are radial and tangential distortion, where radial distortion is the most significant one [138, 154]. In order to correct a distorted pixel $x_d = (u_d, v_d)$ to an undistorted pixel $x_u = (u_u, v_u)$, one has to define a radial distortion function $L(r)$ which is modeled as

$$x_u = \begin{pmatrix} u_u \\ v_u \end{pmatrix} = \begin{bmatrix} p_x + L(r)(u_d - p_x) \\ p_y + L(r)(v_d - p_y) \end{bmatrix}, \tag{3.25}$$

where p_x and p_y denote the center of radial distortion (typically the principal point). The distortion function $L(r)$ is approximated by a Taylor-series

$$\begin{aligned} L(r) &= 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + ... \\ r &= \sqrt{(u_d - p_x)^2 + (v_d + p_y)^2}, \end{aligned} \tag{3.26}$$

where r denotes the Euclidean distance between the undistorted pixel and the center of distortion. The distortion parameters $\{\kappa_1, \kappa_2, ...\}$ are usually determined together with the intrinsic camera parameters during a calibration process. For detailed description on internal camera calibration and distortion estimation we refer the reader to [53, 138, 154]. A useful tool to perform intrinsic calibration together with distortion estimation is provided by Bouguet et al. [11].

3.2 Multiple View Reconstruction

Since visual SLAM algorithms produce three-dimensional information out of multiple images taken from different locations, this Section outlines the geometry between two or more images of the same scene. Generally, this involves several steps like intrinsic camera calibration, computation of their relative orientation followed by the detection of point correspondences used for triangulation. While intrinsic calibration has been discussed in Section 3.1 we review epipolar geometry in Section 3.2.1, which describes the algebraic relations between two images. Furthermore, the construction of three-dimensional information out of two or more images through triangulation is discussed. We also introduce the concept of bundle adjustment in Section 3.2.2, which exploits the problem of estimating or refining camera parameters and three-dimensional structure simultaneously through an iterative optimization algorithm minimizing some kind of image based error.

3.2.1 Multi-View Geometry

Multi-view geometry deals with the relationship between two or more images and how depth information can be recovered. The different geometric properties and relations of 3D points observed from two calibrated views and their image projections are described by the epipolar geometry. Based on the pinhole camera model depicted in Section 3.1.5 two basic concepts are reviewed in this chapter: First, fundamental and essential matrices are derived describing the algebraic relation between corresponding image points in two precalibrated views. Second, we introduce structure estimation out of two or more views by triangulating projective rays passing through point correspondences.

Epipolar Geometry

Given a scene point $\mathbf{X}$ and its image projections x and x' using two different projection matrices $\mathbf{P}$ and $\mathbf{P}'$, the relation between both image points is encoded in the fundamental matrix $\mathbf{F}$.

Figure 3.4 sketches this relationship between image points x and x': One can easily see that image points, the triangulated 3D point $\mathbf{X}$ and both camera centers $\mathbf{C}$ and $\mathbf{C}'$ define a plane which is called *epipolar plane*. The *epipoles* e and e' are the points of intersection of the line joining the two camera centers with the image planes. They can also be interpreted as the projection of the camera center of one view in the second one. The *baseline* describes the line joining the two camera centers. Intersecting the epipolar plane with the image planes results in the *epipolar lines* l and l'. As shown in Figure 3.4 the epipolar line in one view can be interpreted as the projection of the ray from camera center to world point $\mathbf{X}$ in the second view.

Above described entities allow us to derive a relationship between both image points. Generally, each point x in one image corresponds to an epipolar line l' in the second image. This projective mapping from points to lines is encoded in the fundamental matrix $\mathbf{F}$:

$$l' = [e]_\times \, x' = [e']_\times \, H_\pi \, x = \mathbf{F}x, \tag{3.27}$$

with $l' = e' \times x' = [e']_\times x'$. Equation 3.27 can be interpreted as the transfer of image point x via the epipolar plane to a corresponding point x' into the second view written as $x' = H_\pi \, x$ using a 2D homography H_π. Finally, we define the fundamental matrix

$$\mathbf{F} = [e']_\times \, H_\pi. \tag{3.28}$$

Up to now we described a correlation between points and lines. Equation 3.27 can be used to derive a geometric relation between corresponding image points x and x' the following way:

$$\begin{aligned} x'^T \, l' &= 0 \\ x'^T \, \mathbf{F} \, x &= 0, \end{aligned} \tag{3.29}$$

which is also known as the *correspondence condition*, using the condition that x' lies on the epipolar line l'.

The fundamental matrix is a uniquely defined, homogeneous 3×3 matrix of rank 2 and has seven DOF. Therefore, at least seven point correspondences are necessary to compute $\mathbf{F}$. For a more detailed derivation of the fundamental matrix and its computation we refer the reader to [53].

The essential matrix $\mathbf{E}$ can be seen as a specialization of the fundamental matrix operating on normalized image coordinates

$$\hat{x}'^T \, \mathbf{E} \, \hat{x} = 0, \tag{3.30}$$

where $\hat{x} = K^{-1} \, x$ and $\hat{x}' = K'^{-1} \, x$. To understand the relation between $\mathbf{F}$ and $\mathbf{E}$ we

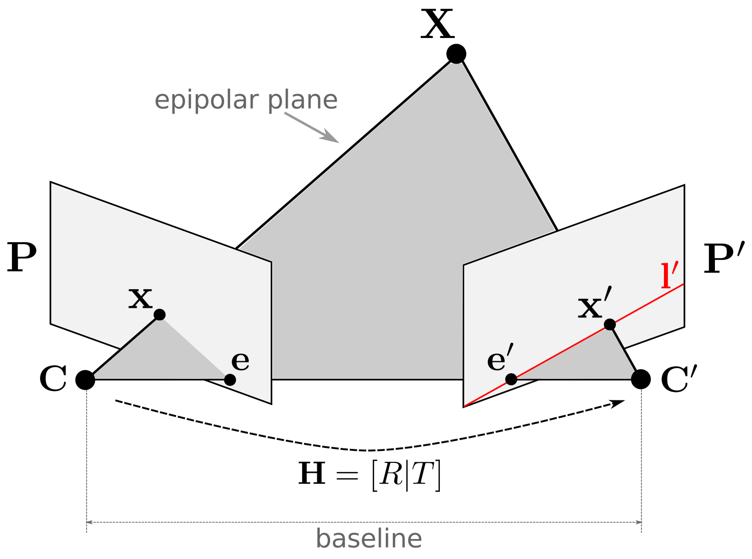


Figure 3.4: Two-view geometry. Two cameras marked by their projection centers **C** and **C**′ are shown. The image projections x and x' of the scene point **X** together with both camera centers form the epipolar plane. The epipolar line l' is defined by the intersection of the epipolar plane and the image plane. The line joining both camera centers is called baseline. Intersecting the baseline with both image planes results in the epipoles e and e'. The epipolar line l' can also be seen as the image of the ray in space defined by **C** and **X** in the second view. So the projection x' of **X** in the second view must lie on l'.

substitute for $\hat{x}'$ and $\hat{x}$ resulting in the following equations

$$\begin{aligned}\left(K'^{-1}\,x'\right)^{T}\,\mathbf{E}\,K^{-1}\,x &= 0\\ x'^{T}K'^{-T}\,\mathbf{E}\,K^{-1}\,x &= 0\\ K'^{-T}\,\mathbf{E}\,K^{-1} &= \mathbf{F}\\ K'^{T}\,\mathbf{F}\,K &= \mathbf{E}\end{aligned} \tag{3.31}$$

Multi-View Triangulation

The projection matrix explained in Section 3.1.5 defines a mapping from three-dimensional world coordinates to two-dimensional image coordinates. Triangulation from two or more views describes the inverse problem, where a world point can be computed from two corresponding image positions and the projection matrices from those views. Assuming noise-free projection matrices and correspondences, the reconstructed point is defined by the intersection of the two back-projected rays through the image points. In practice these two measurements are subject to noise and their rays will not intersect (compare Figure 3.5). Therefore, an estimated solution needs to be computed. The most popular method makes use of the projection equation $x = \mathbf{P}\,\mathbf{X}$. Given an image point in its homogeneous form $x = w(u, v, 1)$ and splitting the known camera matrix $\mathbf{P}$ into rows the mapping from world coordinates to image coordinates can be rewritten as follows:

$$\begin{aligned} wu &= P_1^T\mathbf{X} \\ wv &= P_2^T\mathbf{X} \\ w &= P_3^T\mathbf{X}, \end{aligned} \tag{3.32}$$

where P_i^T denotes the i-th row of the general projection matrix $\mathbf{P}$. Replacing w in the first two equations results in a linear equation system

$$\begin{aligned} P_3^T\mathbf{X}u - P_1^T\mathbf{X} = 0 \\ P_3^T\mathbf{X}v - P_2^T\mathbf{X} = 0 \end{aligned} \tag{3.33}$$

for a single homogeneous image point x. Since a three-dimensional world point $\mathbf{X}$ is described by a 4-vector at least two corresponding image points $x \leftrightarrow x'$ and the appropriate projection matrices $\mathbf{P}$ and $\mathbf{P}'$ are necessary. The required solution for $\mathbf{X}$ is given by the eigenvector corresponding to the smallest eigenvalue of the matrix $\mathbf{A}$, which is defined as follows:

$$\begin{pmatrix} P_3^T u - P_1^T \\ P_3^T v - P_2^T \\ (P')_3^T u' - (P')_1^T \\ (P')_3^T v' - (P')_2^T \end{pmatrix} \mathbf{X} = \mathbf{A}\,\mathbf{X} = 0. \tag{3.34}$$

Equation 3.34 describing two-view triangulation can be easily extended to multiple views, where $\mathbf{A}$ is a $2n \times 3$ matrix and n the number of views where $\mathbf{X}$ is visible in.

3.2.2 Structure and Motion Estimation

In the previous section we described the geometric relationship between two views in terms of fundamental and essential matrix. Furthermore, we show how corresponding points between calibrated image pairs can be used to compute three-dimensional information. Generally, triangulation cannot be performed exactly since most correspondence algorithms deliver noisy point correspondences which do not satisfy the epipolar constraint. Looking at Figure 3.5 the correct image points x and x' are close to the measured ones $\hat{x}$ and $\hat{x}'$, where d and d' denotes the Euclidean distance between them. Moreover, camera poses used for triangulation cannot be estimated exactly, which further decreases reconstruction accuracy. Hence, we seek to find a 3D point **X** and camera parameters **P** and **P**′ which minimize the sum of squared errors $err = d^2 + d'^2$ between measured and predicted image points, also referred to as reprojection error. Usually, an iterative non-linear minimization algorithm such as Levenberg-Marquardt [78] is used to estimate both structure and motion. These considerations can also be extended to an arbitrary number of views and points, also referred to as bundle adjustment (BA) or structure from motion (SfM).

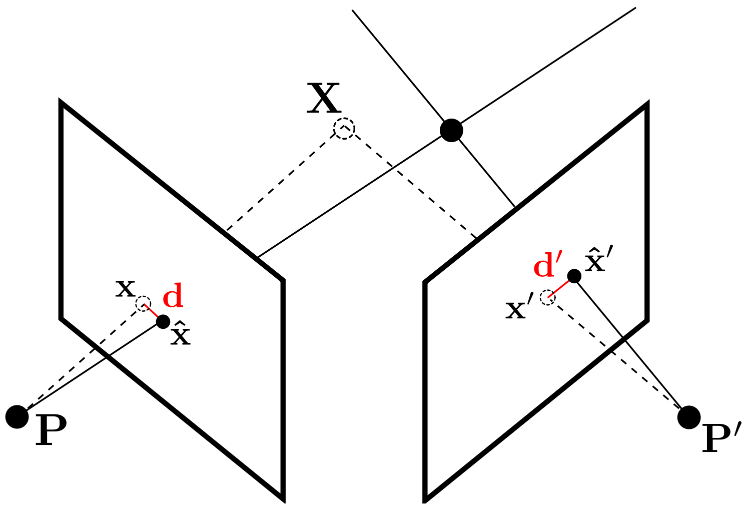


Figure 3.5: Triangulation error. Given projection matrices **P** and **P**′ a three-dimensional point **X** is computed from two corresponding image points x and x'. Because of measurement noise triangulation cannot be performed exactly resulting in reprojection errors d and d' between measured $(\hat{x}, \hat{x}')$ and predicted (x, x') image points.

Basically, there exist two methods performing structure and motion estimation from multiple views: Sequential or batch based methods.

Sequential Methods Starting from a two-view reconstruction subsequent image views are inserted one at a time. Hereby, each view is registered to an already existent pointcloud. An image view with known intrinsics can be registered to an already existing pointcloud through at least three $2D \leftrightarrow 3D$ correspondences resulting in four solutions [53]. Recovering both intrinsic and extrinsic camera parameters can be done by at least six correspondences. This form of sequential structure from motion has been used from many authors like [21, **?**, 106]. The reconstruction is then extended by adding three-dimensional points through standard triangulation methods using two or more views.

As an alternative one can merge partial reconstructions from two views into a single consistent 3D model using $3D \leftrightarrow 3D$ correspondences. At least four non-collinear points are necessary to compute a similarity transformation registering both pointsets. Partial reconstructions are merged sequentially or hierarchically as described by [37, 64, 42].

Alternatively, one can exploit epipolar geometry to relate an incoming image to the previous one. Given eight image point correspondences one can use the Eight-Point-algorithm [52] to compute the fundamental matrix. Given the internal calibration matrices we can recover the relative camera motion between two views from the essential matrix as described in [53]. The resulting projection matrices can be used for structure estimation through triangulation as described in Section 3.2.1. Recently Snavely et al. [121] followed this approach.

Batch-based Methods Batch-based methods use all captured images simultaneously to recover structure and motion within an optimization routine. In contrast to sequential methods there exists no error propagation from previous reconstructions since the overall error can be distributed equally. Starting from a coarse initial solution structure and motion is improved through standard bundle adjustment routines allowing for fast convergence [135, 79, 20].

Nearly all described methods perform some kind of structure and motion refinement after initial reconstruction minimizing an image based error, which is also known as **bundle adjustment** (BA). In the following, estimation of camera parameters and three-dimensional structure through an iterative least-squares solver is explained. Furthermore, robust cost functions are introduced to deal with data association outliers.

Problem Definition

Given an initial estimate of camera poses $\mathbf{P_j}$ and 3D landmarks $\mathbf{X_i}$ bundle adjustment refines both parameter sets simultaneously by minimizing the reprojection error

$$\epsilon(\mathbf{P_j}, \mathbf{X_i}) = m_{ij} - \mathbf{P_j} \cdot \mathbf{X_i}, \tag{3.35}$$

where m_{ij} denotes the image point of $\mathbf{X_i}$ captured by camera $\mathbf{P_j}$. Throughout this chapter m_{ij} are denoted as measurements or observations, which are kept fixed during the whole optimization procedure. Usually, a non-linear least-squares solver is used to minimize the sum of squared residuals, which can be summarized as

$$\min_{\mathbf{X_i},\mathbf{P_j}} \sum_{ij} \| m_{ij} - \mathbf{P_j} \cdot \mathbf{X_i} \|^2, \tag{3.36}$$

using the standard projection function.

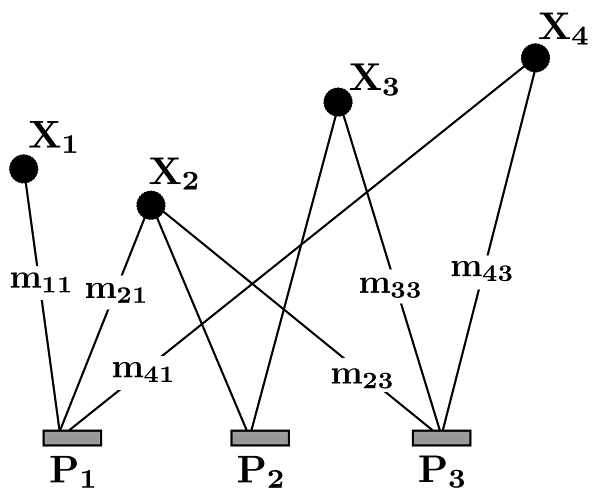


Figure 3.6: Bundle adjustment problem. Four scene points $\mathbf{X_1}, ..., \mathbf{X_4}$ are observed by three cameras $\mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}$. The lines visualize which scene point produces a measurement in a specific camera. Hereby, m_{ij} denotes the measurement of map point $\mathbf{X_i}$ in camera $\mathbf{P_j}$.

Figure 3.6 shows a bundle adjustment problem consisting of three camera poses and four map points to be optimized. The lines visualize which map point is associated to a measurement in a specific view.

Parameter Estimation

The cost function depicted in Equation 3.35 describes a non-linear relationship between parameters and measurements. Further, the number of parameters to be optimized is much smaller than the number of measurements. So we have to deal with an overdetermined non-linear system of equations, which can be solved by a non-linear least-squares optimization routine as described in Appendix A.

The projection matrices $\mathbf{P_j}$ and world points $\mathbf{X_i}$ can be rearranged in a single parameter vector $\mathcal{P} = \{P^T, X^T\}$, where $P = \{\mathbf{P_1}, ..., \mathbf{P_m}\}$ contains the camera parameters and $X = \{\mathbf{X_1}, ..., \mathbf{X_n}\}$ the map point parameters. Similarly, the image measurements denoted as $\mathcal{M}$ can be divided into $\mathcal{M} = \{M_1, ..., M_n\}$, where each M_i contains the measurements available for map point $\mathbf{X_i}$. Each M_i is written as $M_i = \{m_{i1}, m_{i2}, ..., m_{im}\}$, where m_{ij} describes the projection of the i-th world point into the j-th image. The measurements are visualized in Figure 3.6 as lines connecting map point and camera pose. Assuming that each map point is visible in each image results in a maximum of $m \cdot n$ measurements. We further recognize that image point m_{ij} only depends on the parameters of the j-th camera and the i-th point, which leads to $\frac{\partial m_{ij}}{\partial P_k} \neq 0$ only if $j = k$ and $\frac{\partial m_{ij}}{\partial X_k} \neq 0$ only if $i = k$.

Because of the non-linearity of the cost function and the sparse relationship between measurements and parameters a sparse Levenberg-Marquardt algorithm as described in Appendix A.3 is applied. The Jacobian $J = \frac{\partial \mathcal{P}}{\partial \mathcal{M}}$ shown in Figure 3.7(a) exhibits this sparse block structure. In practice a map point is not visible in all views as shown in Figure 3.6, which results in the appropriate Jacobian sketched in Figure 3.7(b). Referring to the Jacobian J used in Appendix A.3

$$J = \begin{pmatrix} A_1 & B_1 & & & \\ A_2 & & B_2 & & \\ \vdots & & & \ddots & \\ A_n & & & & B_n \end{pmatrix} \tag{3.37}$$

the blocks A_i contain the derivatives with respect to the camera parameters $P = \{\mathbf{P_1}, ..., \mathbf{P_m}\}$ and are structured like this:

$$A_i = \begin{pmatrix} A_{i1} & & \\ & \ddots & \\ & & A_{im} \end{pmatrix}, \tag{3.38}$$

where $A_{ij} = \frac{\partial m_{ij}}{\partial P_j}$. The blocks $B_i = \frac{\partial M_i}{\partial X_i}$ contain the derivatives with respect to the point parameters $X = \{\mathbf{X_1}, ..., \mathbf{X_n}\}$, where $B_{ij} = \frac{\partial m_{ij}}{X_i}$

$$B_i = \begin{pmatrix} B_{i1} \\ \vdots \\ B_{im} \end{pmatrix}. \tag{3.39}$$

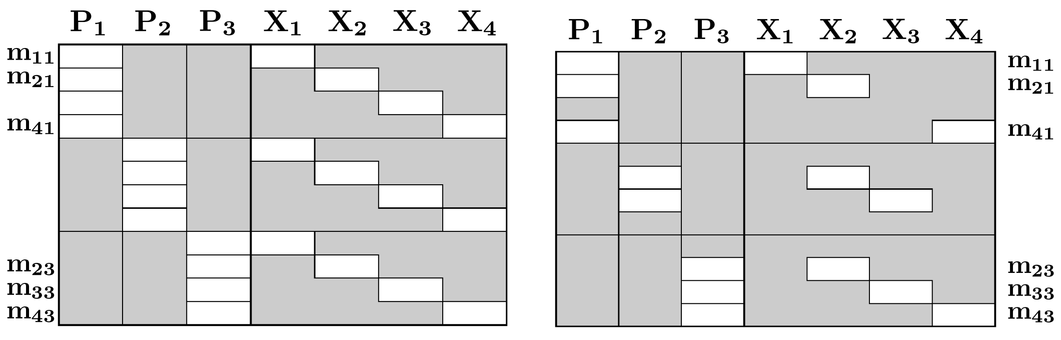


Figure 3.7: Bundle adjustment Jacobian structure. (a) If every map point is observed from every camera we get this kind of block structure in the Jacobian. The gray boxes remain empty. (b) Here we show the Jacobian structure for the bundle adjustment problem depicted in Figure 3.6. The blocks corresponding to those map points not producing a measurement in a certain camera are set to zero.

These matrices together with the error function in Equation 3.36 are used in the sparse Levenberg-Marquardt algorithm to optimize both structure X and motion P. For a detailed derivation of the bundle adjustment problem and some implementation hints utilizing the block structure as well as the sparsity of the Jacobian we refer the reader to the excellent reviews given in [53] and [137].

Robust Cost Functions

Looking at the cost function in Equation 3.36 the traditional bundle adjustment problem optimizes the Euclidean reprojection error in a least-squares manner. If measurements are free of outliers, the least-squares cost function would lead to the Maximum Likelihood estimate of the parameters. In practice measurements cannot be estimated robustly due to data association failures (e.g. wrong feature point matching). This would result in large reprojection errors having a strong influence on the squared cost function to be optimized. Figure 3.8 demonstrates the effect of a single outlier (red) to a least-squares fit of a line to some data points.

To be more robust against outlier measurements the least-squares cost function is replaced by an M-Estimator. To understand their behavior we take at look at the general

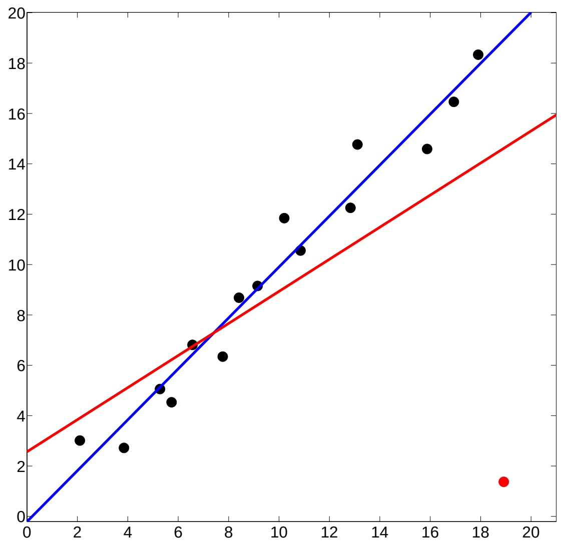


Figure 3.8: Least-squares line fitting. The least-squares estimate of the line through the data points is shown in blue. A single outlier data point (red) greatly influences the result of the optimum least-squares line fit depicted in red.

formulation of a bundle adjustment problem

$$min \sum_i C(\epsilon_i), \tag{3.40}$$

where ϵ_i denotes the residuals and $C(.) : \mathbb{R} \rightarrow \mathbb{R}$ a cost function operating on the residuals which should be less increasing than square. In traditional bundle adjustment $C(.)$ is written in a least-squares sense $C(\epsilon) = \frac{1}{2}\epsilon^2$ resulting in Equation 3.36 with ϵ defined as the standard reprojection error from Equation 3.35. A minimum occurs if the derivative of Equation 3.40 becomes zero

$$\sum_i \psi(\epsilon_i) \frac{\partial \epsilon_i}{\partial P} = 0, \tag{3.41}$$

where P denotes the parameters to be estimated and

$$\psi(\epsilon_i) = \frac{\partial C(\epsilon_i)}{\partial P} \tag{3.42}$$

is called the influence function. Let us define a weighting function operating on the influence function $w(x) = \frac{\psi(x)}{x}$, then Equation 3.41 becomes

$$\sum_i w(\epsilon_i)\, \epsilon_i \, \frac{\partial \epsilon_i}{\partial P} = 0, \tag{3.43}$$

which is an equation system containing weighted gradients. As shown in [**?**] Equation 3.43 results from solving an iterated reweighted least-squares problem, which can be solved by iterative numerical algorithms like Gauss-Newton or Levenberg-Marquardt. Since the weighting function depends on the residuals $w(\epsilon_i)$, it must be recomputed every iteration.

Table 3.5 describes different cost functions $C(x)$ (including the standard squared cost function), which could be used if outliers are present. Even their influence function $\psi(x)$ as well as the weighting term $w(x)$ are presented. Their graphical representation is shown in Table 3.6.

M-Estimator	$C(x)$	$\psi(x)$	$w(x)$
Squared	$\frac{1}{2}x^2$	x	1
L_1	$\|x\|$	$sgn(x)$	$\frac{1}{\|x\|}$
Huber $\begin{cases} \|x\| \leq k \\ \|x\| \geq k \end{cases}$	$\begin{cases} \frac{1}{2}x^2 \\ k(\|x\| - \frac{k}{2}) \end{cases}$	$\begin{cases} x \\ ksgn(x) \end{cases}$	$\begin{cases} 1 \\ \frac{k}{\|x\|} \end{cases}$
Tukey $\begin{cases} \|x\| \leq k \\ \|x\| \geq k \end{cases}$	$\begin{cases} \frac{k^2}{2}\{1 - exp\{-\left(\frac{x}{k}\right)^2\}\} \\ \left(\frac{k^2}{6}\right) \end{cases}$	$\begin{cases} x\left[1 - \left(\frac{x}{k}\right)^2\right]^2 \\ 0 \end{cases}$	$\begin{cases} \left[1 - \left(\frac{x}{k}\right)^2\right]^2 \\ 0 \end{cases}$

Table 3.5: Various M-estimators. We summarize the cost function $C(x)$, the influence function $\psi(x)$ and the resulting weight $w(x)$ of four M-Estimators.

Squared error As mentioned above the squared error is not robust against outliers since their influence is not bounded. Looking at the weighting function in Table 3.6, each residual is weighted equally.

L_1 It measures the absolute value of the residuals, where outliers are given less weight. Of course they reduce the influence of large errors but they are still given low

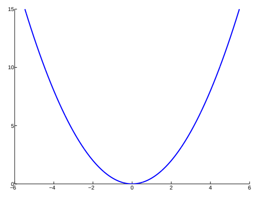
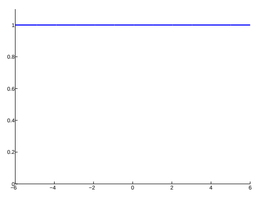
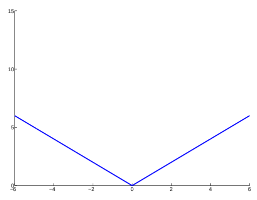
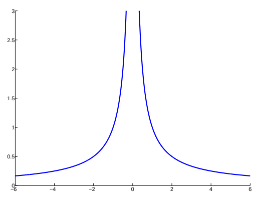
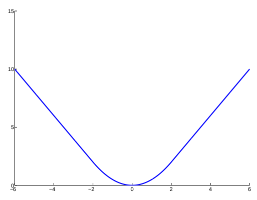
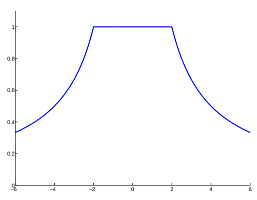
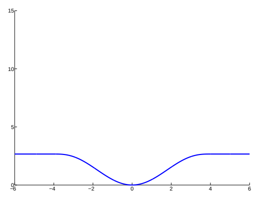
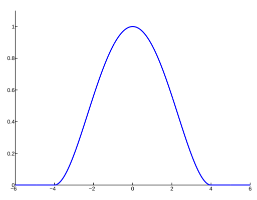
M-Estimator	$C(x)$	$w(x)$
Squared		
L_1		
Huber		
Tukey		

Table 3.6: Cost and weighting functions of different M-Estimators. Looking at the weight $w(x)$ one can see that the squared cost function assigns equal weight to each residual. L_1, Huber and Tukey instead reduce the influence of high residuals. Tukey is a so called cut-off estimator since measurements resulting in high residuals are regarded as outlier.

weight since there is no cut-off point.

Huber Introduced by Huber et al.[59], it combines the standard squared and the L_1 error, where residuals smaller than k get constant weight and higher residuals are downweighted. The weighting function shown in Table 3.6 reflects this behaviour.

Tukey In contrast to the others, Tukey [139] is a cut-off estimator, where residuals above a certain threshold are depicted as outliers. This behavior is reflected in the weighting function in Table 3.6.

Huber and Tukey both depend on a tuning constant k, which is used for outlier identification and weight computation, where smaller values for k results in more resistance to outliers. Since an iterative optimization procedure is used, the residuals change in every iteration. Therefore, the value of k is a critical factor throughout the optimization process which should be recomputed every iteration based on the distribution of the residuals. Generally $k = 1.345\sigma$ for Huber and $k = 4.685\sigma$ for Tukey with σ denoting the standard deviation of the residuals. A robust measure for σ is given by the median absolute deviation (MAD) of the residuals

$$\sigma \approx 1.4826\, MAD(\epsilon_1, ..., \epsilon_n) \tag{3.44}$$

$$MAD(\{X_1, ..., X_n\}) = median_i\left(|X_i - median_j(X_j)|\right). \tag{3.45}$$

For a detailed insight on robust regression and data analysis the reader is referred to Mosteller and Tukey [90].

3.3 Conclusion

In this chapter we provided the preliminaries for this work. The principles of projective geometry, especially the perspective camera model, form the basic prerequisite for every visual SLAM algorithm. The estimation of three-dimensional points out of two or more views are used especially in the map building process. Moreover, the understanding of the imaging process and parameter estimation techniques such as bundle adjustment is required to form a basic visual SLAM procedure.

4 Visual Localization and Map Building in Static Environments

Throughout this chapter our own visual SLAM algorithm operating within a static environment is derived, whose building blocks are sketched in Figure 4.1. Before describing the proposed algorithm in detail, we provide an extensive overview on current state-of-the-art techniques to perform visual localization and map building. After a brief problem description in Section 4.1 we first describe various visual features, their parameterizations, field of application and properties in Section 4.2. Localization and map building - although performed simultaneously - are examined separately. Section 4.3 focuses on different map representations and visual map building techniques given accurate sensor poses. The localization process using vision sensors assuming a previously constructed point map is described in Section 4.4. Finally, current loop closing techniques together with fast and robust place recognition algorithms are examined Section 4.5.

In the last Section 4.6 our visual SLAM algorithm is explained in detail. Based on the information gained from state-of-the-art review in the preceding theoretical background sections, we decided to use incremental structure from motion techniques to build a three-dimensional pointcloud map where the robots trajectory is represented by keyframes.

4.1 Introduction

Generally, every visual SLAM algorithm comprises two main parts: After reading and processing the sensor data, a map from a previously unknown environment is built and localization within this map is performed. Optionally, loop closure detection and correction can be added to handle large scale environments. Figure 4.1 outlines the main tasks to be solved when performing localization and mapping.

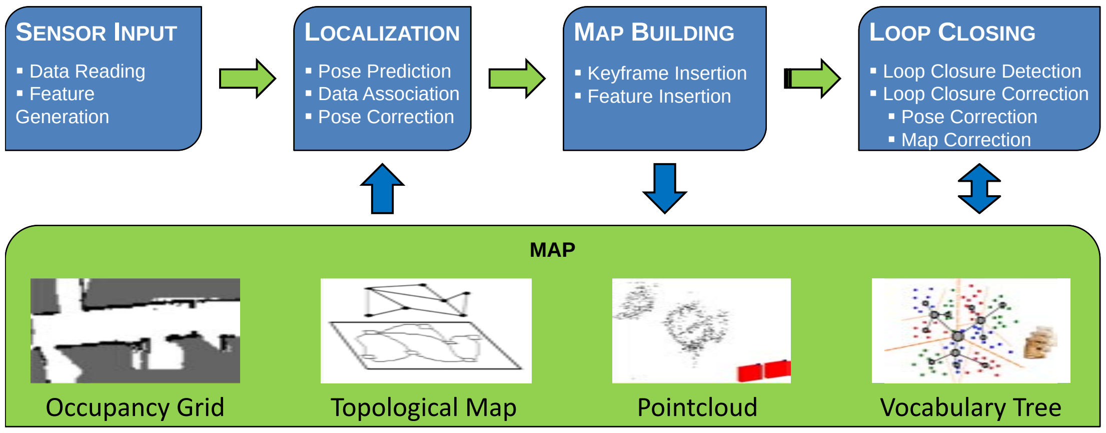


Figure 4.1: Simultaneous Localization and Map Building. Despite data processing, a SLAM algorithm consists of two building blocks: Localization and Map Building. Localization is the process of determining the robot/sensor pose within a given environment. Map building instead assumes accurate pose estimates while constructing a previously unknown surrounding. The loop closing process, consisting of loop closure detection and correction, can be decoupled from the whole SLAM process.

Localization describes the problem of determining the robot/sensor pose relative to a given map. Using a vision sensor this process is also known as *position tracking* or *visual pose estimation*, since the robot keeps track of its pose while moving within the environment.

The *mapping* process instead describes the problem of building an accurate representation of the surrounding given accurate sensor poses. In many cases a three-dimensional representation is constructed when using visual sensor information.

When performing simultaneous localization and mapping one has to think about the type of map to be built, the appropriate mapping algorithm as well as how localization is performed within the map:

The environment representation must be chosen carefully taking into account different aspects like sensor information, size and structure of the environment the robot is traveling in and subsequent higher-level tasks. While occupancy grid maps are more suitable for range sensor devices like time-of-flight cameras, laser or sonar [44, 32, 47, 105], feature based maps are preferred for monocular camera or stereo mapping

[63, 25, 65]. A robot equipped with a laser sensor moving in the plane constructs two-dimensional occupancy grids. Contrary, an autonomous vehicle traveling in a multi-floor environment capturing the surrounding with a stereo-camera would construct a three-dimensional pointcloud associated with some distinctive visual features (e.g SIFTs, SURFs or Harris corners). Size, structure, appearance and complexity of the environment to be constructed often restrict the developer to a specific map representation. When dealing with a large outdoor scenery containing lots of distinctive features, sparse point based maps are preferred because of storage requirements [102]. A more detailed representation like textured dense surface information or three-dimensional occupancy grids could be used for describing a small, restricted desktop environment [93]. If the reconstructed map should be used for any higher-level robotic tasks such as grasping or path planning, occupancy grid maps are the method of choice since they contain information about free, unseen or occupied space as well as occlusion information allowing a robot to compute safe pathways to reach a desired location.

The choice of the mapping algorithm mostly depends on the given sensor information and its noise characteristics. Range sensing devices are often integrated into probabilistic occupancy grids [47], where and inverse sensor model is defined to deal with noisy sensor readings. When dealing with cameras only, salient image features can be extracted and sparse three-dimensional pointclouds are estimated. Inaccurate feature extraction methods or wrong data association are common error sources, which are usually handled by a robust optimization routine during mapping [63].

The localization procedure deals with the representation and estimation of the sensor pose. Here, poses can be estimated in the two- or three-dimensional domain. While a laser range finder restricts a localization algorithm to two dimensions, a range image device allows a 6DOF pose estimate. In both cases there exist a huge variety of different parameterizations like mean and covariance estimated by a Kalman Filter [25], a set of particles [107] or exterior orientation composed of rotation and translation computed through a non-linear least-squares optimizer [63].

4.2 Visual Features for Localization and Map Building

To gather knowledge about the environment an autonomous system perceives the outside world through various sensors and extracts meaningful information out of them. A robot can be equipped with a huge variety of sensors (compare Figure 4.2) like sonar, wheel encoders, bumpers, laser range finders, stereo-rigs, infrared sensors or time-of-flight cameras. All of them provide different measures like range and bearing readings, a single image, the robots velocity, the robots global position or simply the existence of an obstacle.

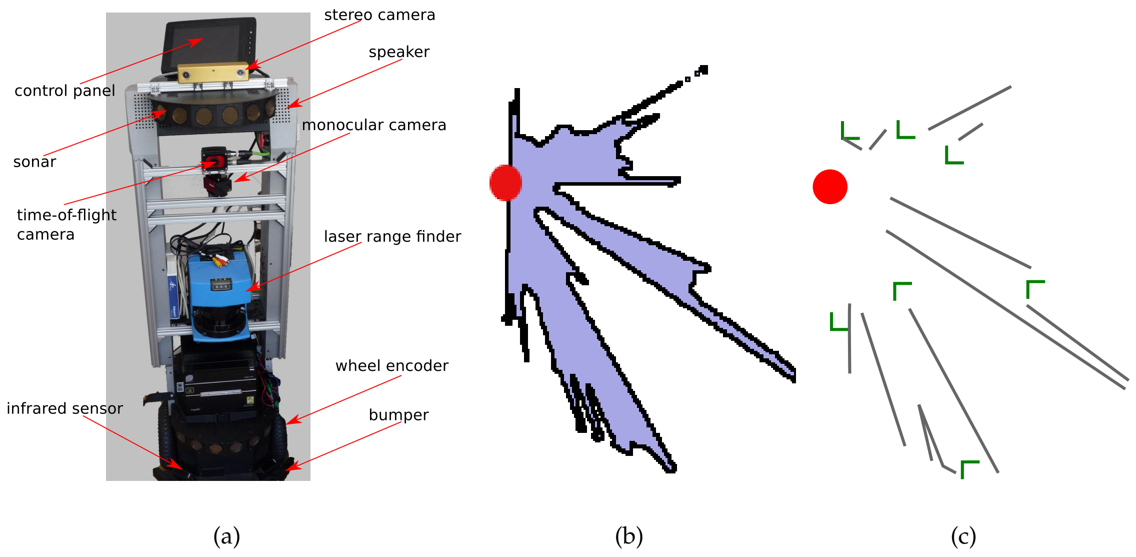


Figure 4.2: Sensors and their features. (a) Robot equipped with various sensors perceiving outside world. (b) Raw laser measurement (360 data points, high information content, computationally not expensive). (c) Corner (green) and line (gray) features extracted out of the raw laser reading (7 data points, low information content, computationally expensive).

In order to describe the surrounding world there exist two main strategies: First, one could use the raw sensor output as input for a SLAM process (e.g. whole image, depth map). Second, only specific information described by some higher-level feature can be extracted out of one or more raw sensor readings (also called feature extraction). In visual SLAM, a *feature* refers to an interesting part in the image domain or camera coordinate frame, ranging from simple structures such as points, edges, curves or lines to more complex descriptions extracted out of a larger image subregion. The most desirable property of an image feature is its stability under local and global image deformations such as scale, rotation and translation changes or illumination and brightness variations. Furthermore, it should be easily recognizable, fast to compute and exhibit a high degree of reproducibility. The process to locate visual features consists of two main steps: feature detection followed by feature extraction. A *feature detection* algorithm decides for every pixel whether there is a specific image feature or not. *Feature extraction* extracts some relevant information around a certain neighborhood of the detected feature, which results in a so called *feature descriptor* of a specific size.

Whether to use raw sensor data or some extracted features highly depends on the underlying application, environment and computational power available: Raw sensor measurements may be used for occupancy grid mapping or obstacle avoidance, while geometric features like lines or polygons are useful for topological or landmark based map building. Even the surrounding environment plays an important role when performing feature extraction. Line, plane or corner features (also known as low-level features) are preferred to represent well-structured indoor environments. Using raw sensor data maximizes the information content and requires no computationally expensive feature extraction algorithms but at the same time comes along with high storage requirements and low distinctiveness. Feature extraction methods instead are usually computationally very expensive leading to serious delays in data processing, which should be considered during the design of a SLAM algorithm. On the other hand features require less storage and are more distinctive than raw sensor readings. To visualize a features information content, memory consumption and distinctiveness Figure 4.2 shows a raw laser reading as well as line and corner features extracted out of it.

In contrast to laser range finders, sonar or odometry sensors, vision based sensors provide an enormous amount of information about the surrounding world and are becoming more and more popular. We present three main types of visual features, assuming that the reader is familiar with CCD/CMOS technology and transport protocols like Firewire (IEEE 1394) or USB. Otherwise, we refer the reader to an excellent overview given by Siegwart and Nourbakhsh [115]. We distinguish between sparse visual features (Section **??**) referring to single image pixels, dense features (Section 4.2.2) describing a larger image portion and appearance based features (Section 4.2.3) corresponding to the information found in one ore more images.

4.2.1 Sparse Visual Features

Sparse visual features are computed out of a subset of image pixels only and are located at a single image pixel, which corresponds to a specific location in the physical world. As mentioned at the beginning a two-step procedure consisting of feature detection and feature extraction is necessary to fully describe a sparse visual feature.

The simplest one is called an image patch, where a small region of a predefined size is cropped around an image point. The image point itself is found by an interest point detector. The most popular interest points used in the robotics community are corner detectors such as Harris [51], FAST [109] or SUSAN [120] often found by searching for two dominant, different edge directions. Figure 4.3(a) shows quadratic image patches extracted around Harris corners. Image patches are fast to compute but are not robust against viewpoint and illumination changes. Since the information content of a single image patch is not very high, they are not very descriptive and reliable.

Contrary, a blob detector, such as maximally stable extremal regions (MSER) proposed by Matas et al. [81], defines image regions by summarizing pixels with similar intensity values. MSERs are invariant under affine transformations and useful for wide-baseline matching. Unfortunately, they are sensitive to lighting effects such as change of daylight or shadows. Figure 4.3(b) demonstrates three regions found by MSER.

Both, image patches and MSER regions, are not invariant to viewpoint and lighting changes, they are not robust against high-frequency noise and have a weak discriminative property. To overcome these limitations one can use the *Scale Invariant Feature Transform* (SIFT) descriptor [74] which combines a Difference of Gaussian (DOG) based keypoint detector with a descriptor based on gradient histograms. After keypoint detection the 16×16 neighborhood is subdivided into 4×4 image patches where a gradient histogram with 8 bins is computed. This results in 16 gradient histograms of length 8 from whom a feature descriptor of length 128 is built. In contrast to beforehand mentioned features it is robust against rotation, scale, lighting and viewpoint changes, which are desirable properties when identifying corresponding features during localization and mapping. Figure 4.3(d) shows SIFT descriptors extracted around DOG keypoints, where the size of the circle corresponds to the estimated scale and the line its gradient magnitude orientation.

Sparse visual features are often used in landmark maps, where the environment is described in two- or three-dimensions. Depending on the used sensor different parameterizations exist leading from a two-dimensional image feature to its three-dimensional representation in the camera coordinate frame:

In case of a monocular imaging device a three-dimensional observation cannot be represented by its coordinates but only by the viewing ray leading from the camera center through the image point. This leads to various representations like unified inverse depth [87] as shown in Figure 4.4(a), anchored homogeneous points [123], inverse scaling [80] or parallax angle [155]. For a well structured overview and a comparison between these different types of landmarks in the context of Kalman Filter-based SLAM we refer the reader to [15]. Contrary, three-dimensional points are also estimated out of more singular views through multi-view triangulation as described in Section 3.2.1.

When dealing with a stereo-setup two-view triangulation can be performed to di-

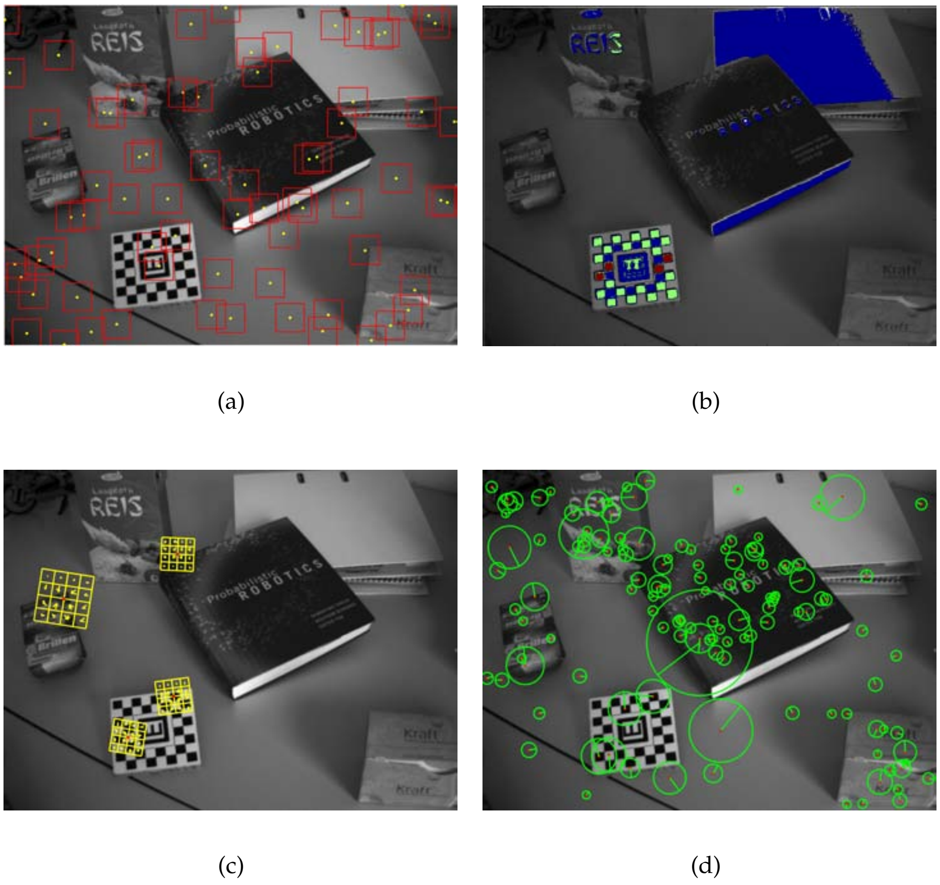
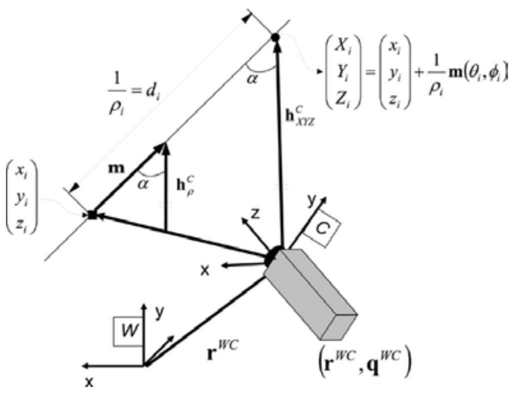
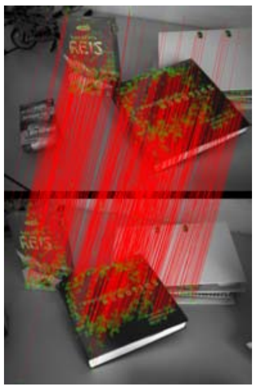


Figure 4.3: Image features. (a) Local image patches extracted around Harris cornerer. (b) MSER features describing a larger, connected image region. (c) Gradient histograms of SIFT features (yellow) around a DOG keypoint (red). (c) SIFT features around DOG keypoints (red) where the size of the circle depicts the scale and the line the magnitude orientation.

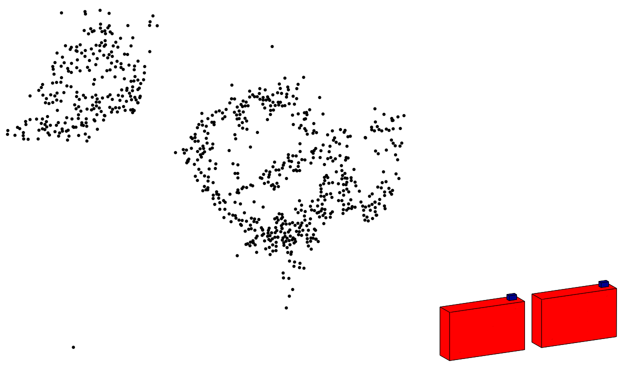
rectly estimate a features depth. After intrinsic and extrinsic camera calibration, corresponding points in both images have to be found through correspondence algorithms. For details we refer the reader to Scharstein and Szeliski [112] for a survey on various techniques. Given associated image points one can use two-view triangulation as described in Section 3.2.1 to estimate the depth of points. Matched point correspondences and the resulting sparse pointcloud are presented in Figures 4.4(b) and 4.4(c). The accuracy of the estimated depth depends on the following parameters: First, the more inaccurate the preceding intrinsic and extrinsic calibration process the more uncertain the resulting depth estimates. Second, point correspondence estimation is a notable error source. One can easily verify that little pixel deviations may lead to large uncertainties in


$\frac{1}{\rho_i} = d_i$
$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i)$
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$
α
$\mathbf{m}$
$\mathbf{h}^C_{XYZ}$
$\mathbf{h}^C_\rho$
z
y
C
x
W
y
x
$\mathbf{r}^{WC}$
$(\mathbf{r}^{WC}, \mathbf{q}^{WC})$

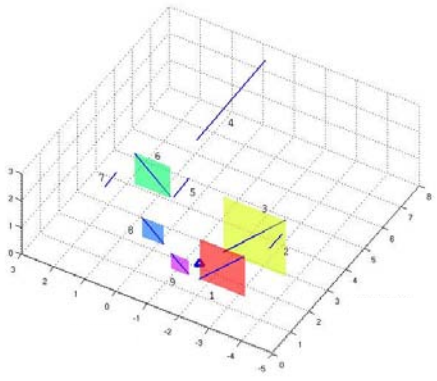
(a)



(b)



(c)



(d)

Figure 4.4: Sparse three-dimensional feature representations. (a) In the monocular case the inverse depth parametrization allows a three-dimensional feature representation (courtesy of [17]). (b) SIFT features extracted from stereo images are matched and produced corresponding image points. (c) The three-dimensional pointcloud is computed through two-view triangulation. (d) Planes estimated out of line segments (courtesy of [95]).

z-direction. Third, there is a direct relationship between the depth of the scene point to be estimated and the estimated z-value. Far away objects are triangulated less accurate than near ones. Fourth, a large baseline allows for a more accurate reconstruction of the three-dimensional point. Last, corresponding image points far away from the principal point lead to more uncertainty in the reconstruction, because of manufacturing errors of the chip and distortion effects which increase at image boundaries. Sparse pointclouds are used both indoors and outdoors and are suitable for any unstructured environments.

Compared to monocular features they are computationally more expensive because of double feature extraction and features matching, but more accurate because of accurate triangulation.

Instead of single three-dimensional point features one could also fit lines, surface patches or other geometric primitives to the pointcloud data (compare Figure 4.4(d)). To estimate the parameters of a line in range data one can use approaches like split and merge, RANSAC, Hough-Transform or linear regression [96]. These types of features are often used in well-structured indoor environments like flats, offices or corridors and require less storage in contrast to sparse pointcloud data. However, feature extraction requires more computation time.

4.2.2 Dense Visual Features

Dense visual features allow a more detailed representation of the environment and also provide occlusion information which can be used for higher level robotic tasks such as path planning or grasping. Dense pointclouds, planes and polygonal surface meshes refer to this group.

In order to calculate a dense three-dimensional pointcloud one could estimate per pixel correspondences between two or more images through an optical flow [149] algorithm as shown in Figure 4.5(a). Although resulting in very realistic environment maps their high computational effort and huge memory requirement restrict their operation area to small desktop scenes or single objects models.

A more sophisticated environment representation consists of a dense surface model computed out of depth maps or range image data as presented in [92, 60]. Unfortunately, these mesh based techniques are very time and memory consuming and therefore only applicable in restricted environments such as small desktop scenes.

Plane parameters can be derived from 3D lines [95] or one can do a least-squares-fitting to pointcloud data [147, 108] as shown in Figure 4.5(b). In contrast to surface meshes these are less memory consuming and applied in large, well-structured indoor environments.

4.2.3 Appearance based Features

Appearance based features (global image features) are functions over the entire image area and therefore correspond to a larger world area. A histogram of the grayvalues or colorvalues can be used to describe a specific location (see Figure 4.6(a)). An alternative

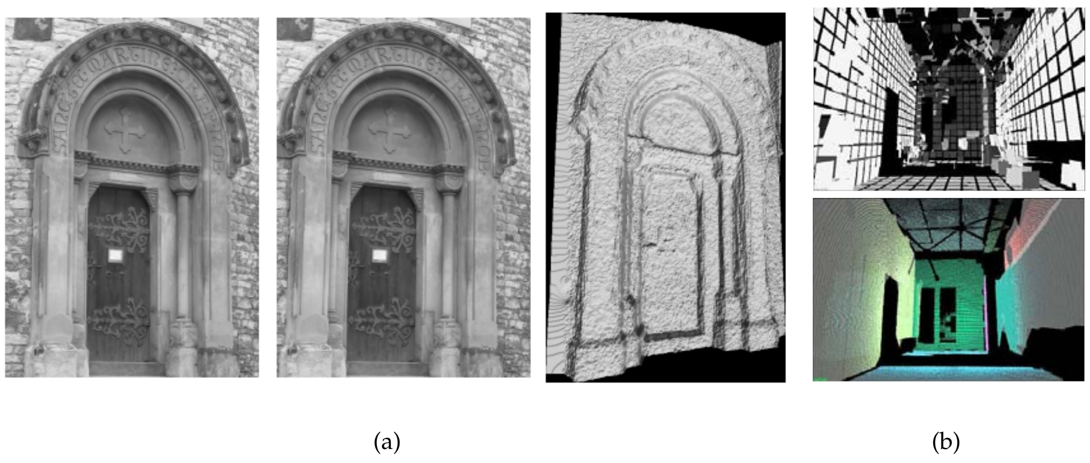


Figure 4.5: Dense visual features. (a) Dense surface information (right) calculated out of stereo images. (b) Planes estimated by a least-squares method (courtesy of [147]).

is to use fingerprints, where a bunch of local features is detected and summarized in a higher-level description [69]. Figure 4.6(b) shows a fingerprint composed of vertical lines and sixteen hues of color. Global image features are the method of choice when dealing with a topological map.

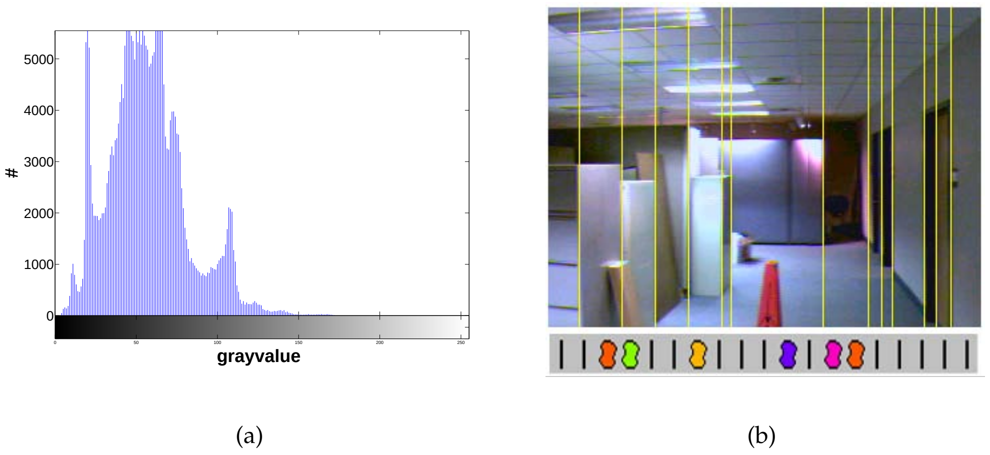


Figure 4.6: Appearance based features. (a) Grayvalue histogram of a single image. (c) Fingerprints are a collection of various image measurements like vertical lines and color information in this case (courtesy of [69]).

4.3 Map Building

Map building describes the process of constructing a map of the environment using the data of the sensors a robot is equipped with. Hereby, the robot poses are assumed to be known. Figure 4.7 shows the graphical model of the mapping process, where gray shaded variables are unknown. Given the sensor poses at different points in time $\mathbf{P_i}$ and their associated sensor readings m_{ij}, the map building process aims at recovering the environment composed of landmarks $\mathbf{X_j}$. Optionally, some kind of relative motion information u_i is provided to compute the absolute poses $\mathbf{P_i}$.

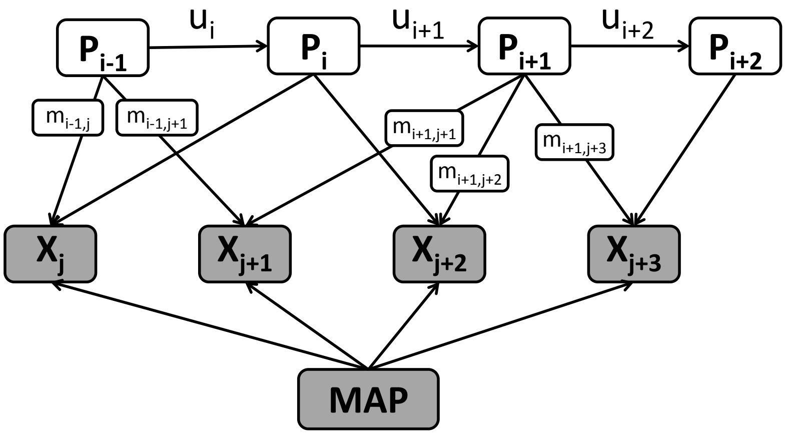


Figure 4.7: Mapping network. The graphical model of the mapping process where gray shaded values (landmark locations) need to be estimated. The sensor poses at different timestamps $\mathbf{P_i}$ and their associated data m_{ij} are used to construct the landmark locations $\mathbf{X_j}$.

The representation of the environment forms the basis of every map building system and therefore a reasoned design is a basic prerequisite for a successful operation of the algorithm. If we could design our "ideal" environment configuration, it should meet the following requirements: Low memory consumption is a basic prerequisite, especially when mapping large scale environments such as houses, campus areas or even cities. At the same time we want to store as much information as possible - for example occlusion information or even semantic information. Furthermore, the representation should be usable for many robotic tasks such as localization, mapping, obstacle avoidance, path planning or grasping. To reduce the computational effort when building a map the complexity of the representation should be kept as simple ass possible. Last but not least, we want the environment to be visualizable and readable for human beings.

The huge variety of applications and requirements lead to many different environment designs, especially when performing visual SLAM. In the following the most important map representations are described together with a short discussion of their properties and field of application. We also depict various map building algorithms since they are strongly linked to the underlying map representation.

4.3.1 Landmark Maps

As stated in Section 4.2 a feature m is represented in the robots coordinate frame or image plane. A *landmark* instead is defined by the features location in the global world coordinate frame through its two- or three-dimensional coordinate $\mathbf{X}$, which is often computed using the robot pose the feature is associated with. Optionally, a landmark uncertainty and some form of signature d is stored too. The signature can be seen as a unique characterization of the landmark and often comes along with the feature extraction algorithm (e.g. keypoint descriptor, grayvalue histogram of an image patch, length of the extracted line, etc.), which is used for data association during map building.

Robotic sensors deliver high-dimensional raw data like 2D or 3D laser range readings, high-resolution images or dense depth maps. In landmark maps only a small number of features are extracted. Local features are then converted to landmarks lying in the global world coordinate frame. The type of landmarks used for map building, data association and localization depends on the sensors used and the environment the robot is traveling in. Figure 4.8 shows some two- and three-dimensional landmark based maps using various geometric primitives.

Generally, a landmark for map building should fulfill the following requirements:

- It should be easily reobservable which allows us to detect it from different robot positions.
- A landmark should be unique among others, so that it can be easily identified over and over again. More specifically, if the sensor returns to an already visited place it should be an easy task to determine that the landmarks have been seen before.
- The parametrization or representation of the landmark should be chosen according to the environment to be mapped. For example points should be preferred to lines when mapping unstructured outdoor environments.
- Only stationary landmarks should be added to the map. Using moving landmarks like persons or cars the robot might estimate a wrong pose when doing localization with the constructed map.

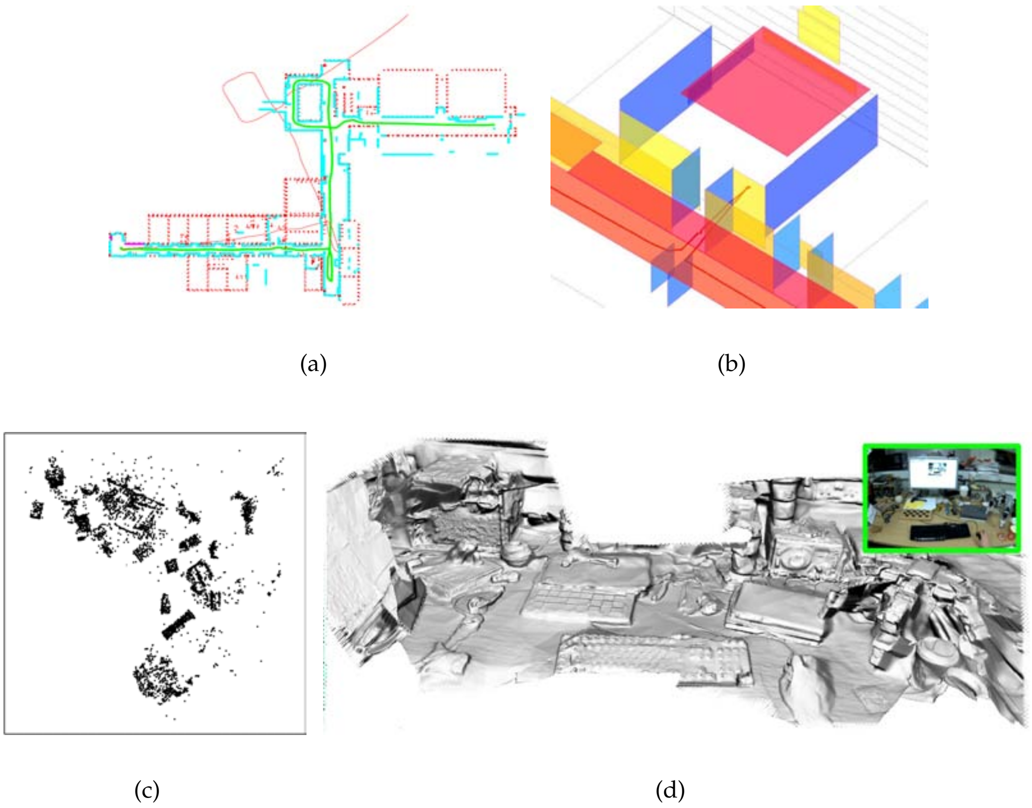


Figure 4.8: Landmark maps. (a) Line features (cyan) are used to represent a planar indoor environment where groundtruth data is shown in red (courtesy of [94]). (b) Mapping of an office room using orthogonal planes constructed out of line segments fitted to laser range data (courtesy of [95]). (c) Pointcloud map of a desktop environment constructed out of single camera poses. (d) Dense surface based reconstruction of an office environment using poses of a single camera. (courtesy of [92])

The general map building workflow is sketched in Figure 4.9 and consists of four main parts:

1. Local features (keypoints, corners, lines, etc.) are extracted from the raw sensor data as explained in Section 4.2.
2. They are transformed to landmarks in the global world coordinate frame using the given sensor poses.
3. Provided the signature from the feature extraction algorithm one has to perform data association in order to distinguish between new landmarks and those already contained in the map. Data association can be performed through nearest neighbor search in the feature space [1, 74] or template matching [63].

4. Finally, unobserved landmarks are used to extend the current map. Features of reobserved landmarks are used to refine the landmark position of those already in the map. This refinement process can be done through recursive filtering techniques (e.g. (Extended) Kalman Filter [25]) or least-squares optimization routine (e.g. bundle adjustment [63]).

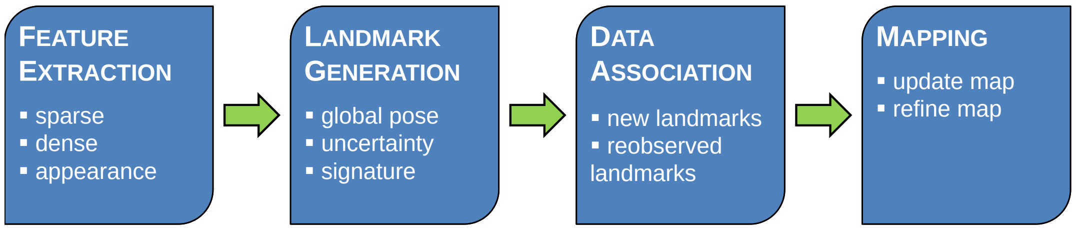
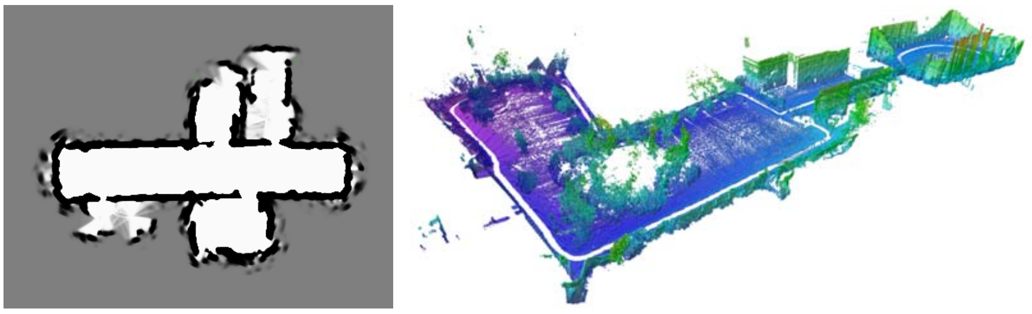


Figure 4.9: Landmark based map building workflow. Building an environment out of raw sensor data can be decomposed into four main tasks: First, features have to extracted. Second, using the provided robot pose they are transformed to landmarks in the global word coordinate frame. Third, using the distinctive feature signature, data association is performed to distinguish between new and reobserved landmarks. Last, new landmarks are added to the map while revisited ones are used to refine existing landmarks.

4.3.2 Occupancy Grid Maps

Occupancy grid maps have been first introduced by Moravec and Elfes [88] in the context of sonar based map building. Hereby, the environment is discretized into an equally sized grid, where each grid cell $\mathbf{X_j}$ holds a probability p_j for being occupied. Each cell (2D) or voxel (3D) can be regarded as free space (low probability), part of an obstacle (high probability) or unseen (unitialized probability). This type of map representation is often used when a robot is equipped with a range sensing device like sonar, laser or range cameras. The range and bearing values together with the robot position can be directly used to verify if a cell is empty space (i.e. the ray strikes through the cell) or occupied (i.e. the cell has been hit by a range measurement).

The goal of an occupcancy grid mapping algorithm is to calculate the posterior probability over maps $p(\mathbf{X}|\mathbf{P_{1:t}}, m_{1:t})$ provided accurate sensor poses $\mathbf{P_{1:t}}$ and the sensor data $m_{1:t}$ up to time t. One key assumption is that grid cells are independent from each other



(a) (b)

Figure 4.10: Occupancy grids. (a) Two-dimensional occupancy grid map made of sonar data (courtesy of [133]). (b) Three-dimensional occupancy grid created out of 3D laser scan data with color coded height (courtesy of [152]).

and their state can be estimated separately

$$p(\mathbf{X}|\mathbf{P_{1:t}}, m_{1:t}) = \prod_j p(\mathbf{X_j}|\mathbf{P_{1:t}}, m_{1:t}). \tag{4.1}$$

To avoid numerical instabilities during map creation each cell stores its probability p_j^t at time t in its log-odd form l_j^t

$$l_j^t = log\left(\frac{p(\mathbf{X_j}|\mathbf{P_{1:t}}, m_{1:t})}{1 - p(\mathbf{X_j}|\mathbf{P_{1:t}}, m_{1:t})}\right), \tag{4.2}$$

which allows for fast and additive map updates through

$$l_j^t = l_j^{t-1} + sensor\_model(\mathbf{X_i}|\mathbf{P_t}, m_t) - l_j^0 \tag{4.3}$$

$$sensor\_model(\mathbf{X_j}|\mathbf{P_t}, m_t) = log\left(\frac{p(\mathbf{X_j}|\mathbf{P_t}, m_t)}{1 - p(\mathbf{X_j}|\mathbf{P_t}, m_t)}\right), \tag{4.4}$$

where l_j^0 denotes the initial belief of the cell which is usually set to 0 [133]. The function $sensor\_model(\mathbf{X_j}|\mathbf{P_t}, m_t)$ returns the probability for the cell $\mathbf{X_j}$ being occupied given the robot pose and sensor data at time t. The sensor model used to interpret every incoming measurement is usually a combination of a linear function and a Gaussian whose reliability depends on the noise characteristics of the used sensor. For a detailed discussion on sensor modeling the reader is referred to [132] or [133]. Usually, all cells lying within the perceptual field of a sensor are considered for an update. The probabilities p_j^t can be easily recovered through

$$p_j^t = 1 - \frac{1}{1 + exp\{l_j^t\}}. \tag{4.5}$$

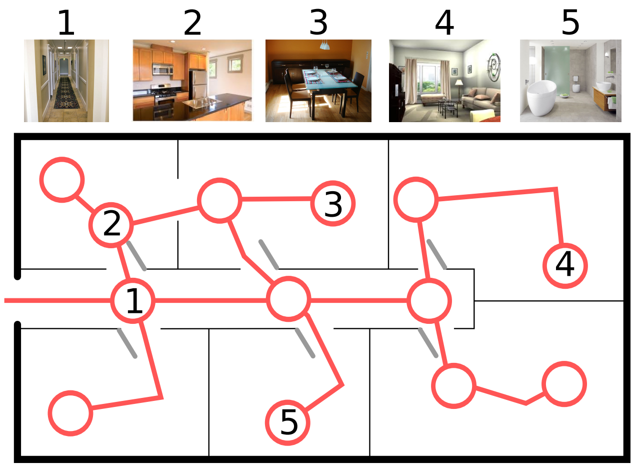
Figure 4.10 presents a two and three-dimensional occupancy grid created with sonar and 3D laser scan data, respectively.

4.3.3 Topological or Appearance Based Maps

In contrast to occupancy grid maps, topological maps dissect the environment into distinct places, where the most relevant image information is concentrated. More specifically, the environment is represented as an undirected, unweighted graph consisting of nodes and edges. The nodes represent a characteristic area or point in the map that is easily recognizable later on. The edges connecting the nodes represent adjacency, which can be seen as safe navigation paths between the specific locations. Figure 4.11(a) shows a topological map of a flat, where the nodes are set at very discriminative places like doorways, corners or specific rooms. Alternatively, also artificial landmarks can be added as a node in a topological map as used by [98] shown in Figure 4.11(b).

The nodes can store a lot of information in order to encode a specific place, e.g. single image feature, a collection of geometric features (e.g. points, lines, planes, corners etc.) or even one or more images characterizing the surrounding of the node. Ulrich et al. [140] attach color histograms to each node while others [110, 111, 3] build the map out of local image descriptors like SIFT or SURF features. To create a more distinctive representation Tapus et al. [131] combine edges from panoramic images, color information and corners extracted from a laser range finder to represent a single node in the map. Many topological map building algorithms [41, 23] use image data only which are stored in a database for fast place recognition later on. Hereby, an image is represented by a bag of visual words where each word is a local image region described by a patch or descriptor. For speedup and storage reasons these visual words are quantized and organized in a tree structure, also known as vocabulary tree.

One challenge when building a topological map is the decision when to add a new node to the graph. Most of the time a new node is inserted when the robot has covered a certain distance (e.g. $3m$ or $4°$) based on relative motion estimates u_i. Another method is to compare the extracted feature information of the currently processed frame to the nodes already existing in the map. If they do not match to any stored region, features are describing an unknown region and a new node is created.



(a) (b)

Figure 4.11: Topological maps. (a) A topological map of an indoor environment where nodes define some distinctive points or areas in the map. The edges connecting the nodes represent pathways which can be traversed safely from the robot. In visual maps often one or more images are representatives for each location. (b) Artificial landmarks like the pink label or the marked docking station [98] can also be used to represent a node in the map.

When building a topological map the edges are added incrementally. In its simplest form an edge just represents a valid path between two nodes which is an essential information for subsequent tasks like navigation or path planning. In metric map building an edge can also store the relative movement u_i of the robot provided by odometry sensors, inertial measurement units or some kind of visual odometry algorithm.

4.3.4 Conclusion

Table 4.1 summarizes above mentioned map types. We compare their ability to handle different sensor data, the resolution/accuracy of the estimated map as well as their memory consumption and computation time. Hereby, computation time includes data preprocessing, possible feature extraction methods and the virtual map building algorithm.

Landmark maps handle sparse visual features only and are applied in localization and mapping tasks. Since they do not provide occlusion information or estimates about free or occupied space they cannot be used for obstacle avoidance, grasping or path planning as occupancy grid maps do. Contrary, they do not require that much memory

Map Type	Sensor Data	Resolution	Memory consumption	Computation time
Landmark Map	2D/3D Features	high	low	medium
Occupancy Grid Map	Raw sensor data	medium	medium-high	medium
Topological Map	Raw sensor data/Features	low	low	high

Table 4.1: Map building techniques. The three different map types are compared regarding their sensor data used for map building, the resolution and memory consumption of the resulting map and the computation time. Hereby, computation time includes sensor data preprocessing, feature extraction and map building.

and result in high accurate environment maps.

The main disadvantage of occupancy grid maps is that memory consumption grows to the third power with the size of the environment explored. Especially the degree of discretization (i.e. the cell resolution) affects the accuracy of the resulting map and its storage requirements. Therefore, many authors [152] use hierarchical data structures like quadtrees or octrees to reduce memory consumption and to speed up access time to specific grid cells. In contrast to landmark based maps they come along with various advantages: First, no feature extraction is necessary since raw sensor data can be used directly. Second, no data association is required. Third, the resulting maps can be used for grasping or path planning. Finally, the accuracy of the resulting map can be controlled by setting the desired resolution.

A topological map can be seen as the most compressed and most abstract representation of the environment. The huge amount of data encoded in each image is broken down to a simplified representation containing the most relevant data for a specific area. Therefore, its main applications are fast and robust place recognition or loop closure detection. Because of the relative motion information encoded in the maps edges, topological maps are often used for higher level robotic tasks like path planning or navigation. In contrast to landmark based maps or occupancy grid maps this form of map representation provides no human readable visualization through geometric features. Since no structure estimation is necessary the computational effort is very low, while on the other hand many image features need to be stored leading to large storage requirements.

4.4 Incremental Localization Techniques

Mobile robot localization describes the process of determining the pose of the robot within a known environment. A graphical depiction is shown in Figure 4.12, where the gray shaded variables are unknowns. Given the map $\mathbf{X_j}$ and sensor measurements m_{ij} associated with some landmarks, the localization process aims at recovering the sensor pose $\mathbf{P_i}$. Unfortunately, measurements are subject to noise and therefore only the most likely pose can be estimated. Consider a robot moving using noisy odometry information u_i only; it might get lost due to high uncertain motion data delivered by the sensors. Thus the robot has to localize itself relative to a given map using additional information stemming from sensors perceiving the outside world.

When doing localization one also has to consider if the robot moves in a well-structured static environment or if it is surrounded by dynamic elements like persons or cars. Short-term changes affect only a single sensor reading and are usually filtered out or treated as noise. More persistent changes like moved furniture, opened or closed doors or lighting changes require special treatment.

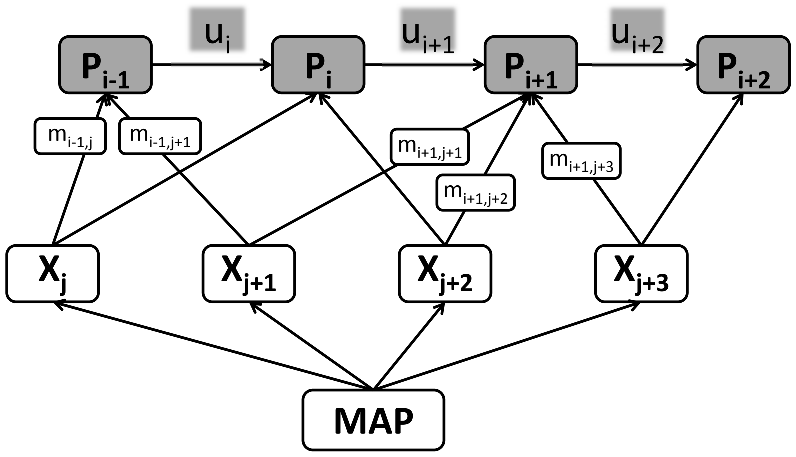


Figure 4.12: Localization network. The graphical model of a localization process, where gray shaded values (sensor poses and interframe motion u_i) are unknown. The observed landmarks $\mathbf{X_j}$ and their associated features m_{ij} are used to estimate the sensor pose $\mathbf{P_i}$.

Generally, localization techniques can be subdivided into *local* or *global* ones. Local localization techniques (also known as tracking or incremental localization) assume an initial pose estimate and the robot keeps track of its pose during navigation. Given the initial sensor pose, sensor data is compared to the existing map resulting in a refined pose. This chapter provides an overview on local localization techniques. We shortly outline their functionality, characteristics and field of application with special focus on visual localization. Global localization is discussed in Section 4.5.

Incremental or local localization assumes a rough prior on the robot pose and no recovery is possible if the algorithm loses its pose. Incremental localization is designed to compensate odometric errors stemming from any odometry device or algorithm. Generally, the incremental localization process can be divided into two separate phases - prediction and correction - which are shown in Figure 4.13:

Prediction Internal motion sensors like wheel encoder, global positioning systems (GPS), inertial measurement units (IMU) or some kind of visual odometry u_i are used to predict a new sensor position $\mathbf{\hat{P}_i}$ from the previous one $\mathbf{P_{i-1}}$. Assuming noisy motion information u_i this increases the uncertainty of the predicted pose which needs to be corrected in the subsequent step by incorporating sensor data. How visual sensors can be used to provide interframe motion estimates is examined in Section 4.4.1.

Correction Exteroceptive sensor readings m_{ij} at time i are used to update the predicted pose, resulting in a more accurate sensor position $\mathbf{P_i}$. Hereby, features provided by the sensor data are associated to landmarks in the map. The difference between the measurements stemming from the map to those from the sensor is examined to refine the prediction $\mathbf{\hat{P}_i}$. This can either be handled by a recursive filtering or pure optimization based approach, which are described in Sections 4.4.2 and 4.4.3, respectively.

Throughout this chapter we distinguish between recursive filtering and optimization based correction methods using vision sensors only. The filtering versions are derived from the Bayes filter, where the robot pose is modeled as a Gaussian distribution (EKF, Unscented Kalman Filter (UKF)), as an equally spaced grid (metric, topological) or as discrete samples (Monte Carlo techniques, particle filter). Optimization based techniques are split into purely appearance-based or point-based methods. An appearance-based method operates in the image space allowing approximate sensor pose estimates only, while point-based algorithms try to compute a pose relative to a given pointcloud allowing for accurate localization.

In the following we shortly describe the generation of odometry information u_i, which is used to predict a preliminary robot pose. Afterwards, update methods of both probabilistic and geometric categories are presented and we highlight their positive and negative aspects in context to purely vision-based localization.

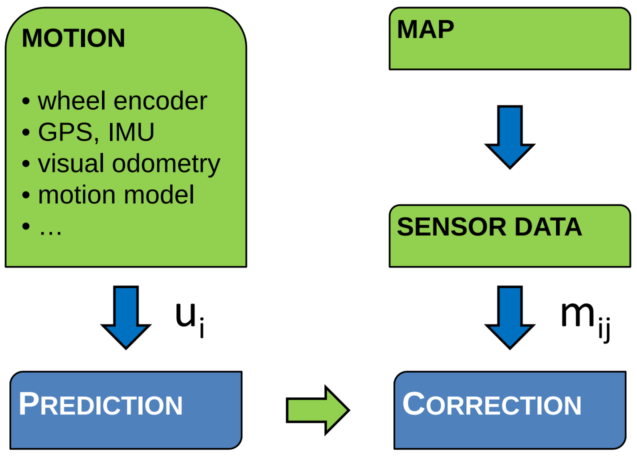


Figure 4.13: Incremental localization workflow. Localization, given a map of the environment, is subdivided into two phases: First, the robots pose is predicted using some noisy sensor information from interoceptive sensors. The underlying map and readings from exteroceptive sensors are used to compute a more accurate pose in the correction phase.

4.4.1 Pose Prediction

The first part of every localization algorithm requires an odometry input u_i, which estimates the change in position between time $i-1$ and i to predict a new robot pose $\hat{\mathbf{P}}_\mathbf{i}$ at time i.

Visual odometry uses sequential camera images to compute interframe motion u_i, which can be estimated in many different ways. In the following the most prominent ones are shortly described. After image acquisition and distortion correction, the computation of inter-frame motion at time i involves the following steps:

1. Feature extraction and matching: Salient image features are detected in the current frame and matched against one or more preceding frames producing corresponding image points.

2. Camera motion estimation can be done in several ways using different imaging devices and motion recovery techniques:

   - Given monocular images and corresponding feature points between three or more images, Nister et al. [?] propose to use the five-point algorithm together with RANSAC to estimate the relative pose between frames. These initial estimates are used to perform multi-view triangulation, resulting in a three-dimensional pointcloud valid up to scale. The pose of every successive frame is estimated relative to the previously constructed pointcloud by a three-point algorithm [50]. Again new 3D points are triangulated using corresponding image features and the pose of the next frame is estimated again.

- Given depth data from a stereo rig, relative motion between a set of corresponding 3D points can be computed by an iterative optimization routine assuming Gaussian landmark noise [99].
- Instead of modeling the error in the three-dimensional space other authors [130] exploit an image-based error. Hereby, the relative motion between two stereo-rigs is computed by minimizing the reprojection error of the pointcloud stemming from the current frame in the previous one.
- Given corresponding 3D points $\mathbf{X_{i-1}} \leftrightarrow \mathbf{X_i}$ provided by two stereo-camera pairs $\mathbf{P_{i-1}}$ and $\mathbf{P_i}$ at time $i-1$ and i, the relative motion is defined by a rigid body transformation $u_i = [R, t]$, which relates both pointsets to each other $R\mathbf{X_{i-1}} + t = \mathbf{X_i}$.

All described methods are locally very accurate and smooth but tend to drift over a long period of time. Hence, using visual odometry information only would result in wrong pose estimates in the long run. Therefore, additional sensor information coupled with the information stemming from a pre-built map are used to correct odometric drift, which is summarized in the update step of every localization algorithm. Various methods tackling this problem are described in the two following sections.

4.4.2 Recursive Filtering Techniques for Pose Correction

In filtering based localization one can distinguish between Markov Localization techniques like EKF or UKF, grid based techniques using a histogram filter or Monte Carlo localization (PF). All are using different approximations to represent the sensor pose and are restricted to operate on specific map representations.

Markov Localization

In EKF-based localization the robot pose is represented by a Gaussian distribution composed of mean and covariance. Hereby, a motion model operating on $\mathbf{P_{i-1}}$ and u_i is used to predict a new pose $\mathbf{\hat{P}_i}$. The measurement model instead computes the prediction of measurements $\hat{m}_{ij}$ given the estimated pose $\mathbf{\hat{P}_i}$ and the precomputed map $\mathbf{X_j}$. The difference between observed m_{ij} and predicted $\hat{m}_{ij}$ measurements is used to compute a new pose $\mathbf{P_i}$. Both motion and measurement model are linearized throughout the process. When dealing with high uncertainties the linearization incurs higher approximation errors of the underlying distribution leading to wrong pose estimates.

The UKF [62] represents the Gaussian as sigma points leading to a more accurate approximation when passed through a non-linear function. Therefore, it gives better estimates than the EKF when dealing with non-linear motion and measurement models or high sensor noise.

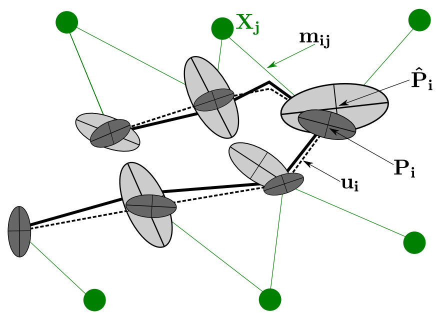
For a more detailed description we refer the reader to [148]. Kalman filter based localization using precomputed landmarks (green) is shown in Figure 4.14(a). The groundtruth path is represented as solid line where the estimated trajectory is marked as dotted line. Applying odometry information u_i to $\mathbf{P_{i-1}}$ results in a predicted pose $\mathbf{\hat{P}_i}$, with a large uncertainty ellipsoid shaded light gray. Incorporating measurement information m_{ij} results in a refined pose $\mathbf{P_i}$, whose uncertainty is decreased (dark shaded ellipsoid).

Histogram Filter

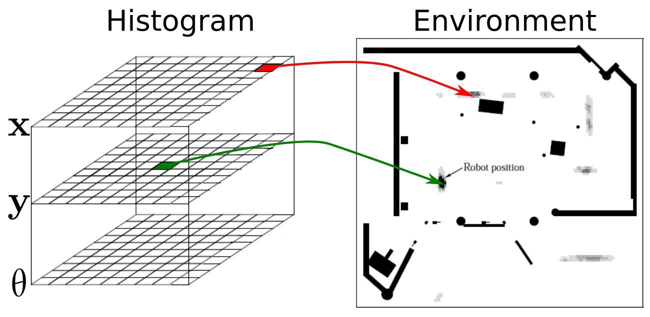
When using histogram filters, poses are distributed over a regular grid, where each occupied cell represents a possible sensor pose [13, 12]. The resolution of the grid can be chosen by the user. Figure 4.14(b) shows a histogram filter for a robot moving in the plane. In case of a very coarse grid resolution this leads to localization in a topological environment, where each grid cell corresponds to a different place or area. They are not computationally expensive but provide less accurate pose estimates. A finer resolution (e.g. 15cm for x and y and 5° for the robots orientation) is called metric localization and leads to accurate pose estimates but at the expense of more computation time and memory consumption. Figure 4.14(b) demonstrates the usage of a histogram filter representing planar robot motion within occupancy grid-based localization.

Monte Carlo Localization

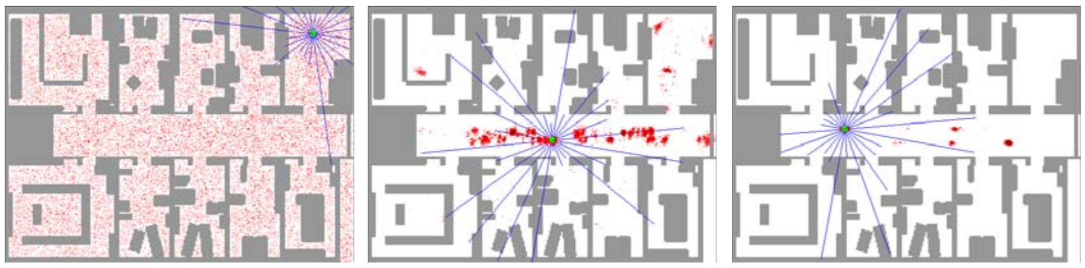
The youngest and most popular localization technique is Monte Carlo localization (also known as particle filtering), where the robot poses are represented by a set of discrete points (particles). Each of them is carrying a weight factor encoding its importance. At the beginning, particles are evenly distributed over the whole map with equal weight. After moving each particle using the odometry information u_i perturbed by noise, the importance weights are updated. As in EKF localization, sensor measurements are pre-



(a)



(b)



(c)

Figure 4.14: Filtering based localization techniques. (a) Kalman filter based localization given some landmarks (green) observed from the robot. Starting from an initial pose the predicted (light gray) and corrected (dark gray) poses together with their uncertainty ellipsoids are shown. The groundtruth path is shown as solid line. (b) A histogram filter operating in the plane together with its map is shown. Each cell combination corresponds to a specific place in the map, where probabilities stored in the cells represent the confidence of the pose. Darker gray values in the map correspond to more likely robot poses (courtesy of [133]). (c) Particle filter based localization using sonar readings (blue), where the most likely pose (highest importance weight) is marked green. The red dots describe all pose hypothesis contained in the particle filter (courtesy of [133]). Any distribution can be modeled using either a histogram or particle filter technique.

dicted using the new particle poses and compared to the measurements stemming from the sensor reading. The more the measurement predictions coincide with the sensor data, the higher the particles are weighted. Finally, resampling of the particles takes place, where those with higher weight are duplicated and those with a low weight are discarded. This leads to a new set of particles with uniform weight concentrated at the most likely map positions. Figure 4.14(c) demonstrates three steps during particle filter based localization using a sonar sensor given an executive drawing of an indoor environment. Starting with pose hypothesis (red dots) distributed over the whole area, pose estimates are further refined by incorporating sensor measurements which are compared against the map. Naturally, the more particles are used the more accurate the resulting pose estimates but the more expensive are the computational costs.

4.4.3 Optimization based Techniques for Pose Correction

The basis of optimization based localization is an iterative optimization routine minimizing a geometric error metric. The sensor pose to be estimated is modeled by its orientation (rotation matrix) and position (translation vector) and therefore three (2D) or six (3D) parameters need to be estimated. Since there is no explicit categorization, we differ between image-based and point-based algorithms.

Point based algorithms require a pre-built two- or three-dimensional landmark map and their associated signatures d as input. After extracting a set of features from the image to be localized, $2D \leftrightarrow 3D$ matches are established between the query image and the existing map. A 6DOF pose relative to the map can be computed by a perspective n-Point algorithm given a set of n correspondences between 3D map points and their 2D image projections [89, 75] or least-squares optimization techniques minimizing the reprojection error [63, 127, 117].

When dealing with range image devices many authors perform data association between 3D range image features and 3D map points leading to $3D \leftrightarrow 3D$ correspondences. In order to be robust against wrong data associations a RANSAC [36] algorithm is used to compute a least-squares solution of the rigid motion between at least 3 point correspondences. Alternatively, one can use a three-dimensional variant of the iterative closest point algorithm to register the pointcloud stemming from the imaging device to the map.

Image based algorithms operate on topological maps and often solve the global positioning task which is described in more detail in Section 4.5.

4.4.4 Conclusion

An overview of localization methods is given in Table 4.2, where we summarized characteristic like sensor data used, type of the underlying map, pose estimation accuracy, robustness to wrong data associations and computation time. Hereby, the computation time includes image processing, possible feature extraction methods as well as the localization algorithm itself.

EKF and UKF localization can only solve the position tracking problem because of the uni-modal representation of the robot pose. Moreover, outliers during data association may cause the filters to fail. Both filters are only applicable to landmark based maps which require feature extraction and landmark generation. One further characteristic is, that they cannot process negative data association information (i.e. if a landmark is represented in the map but not observed by the sensors).

In contrast to EKF and UKF, histogram filters can solve the global positioning problem because of the non-parametric representation. Furthermore, they can handle all kinds of maps and sensor data. Even more, also negative data association can be processed. Their accuracy highly depends on the degree of discretization of the underlying grid, where a finer resolution results in higher computation time.

Particle filters can handle both occupancy grid maps as well as landmark based maps [61]. Additionally, they are very robust against wrong data associations, where random particles can be added. Their accuracy grows with the number of particles but at the expense of a higher computation time.

Optimization based localization algorithms either operate on landmark based maps or topological maps. Naturally, point based algorithms produce more accurate pose estimates and are also resistant to wrong data associations when robust estimators are used. Image based techniques operate on topological maps only providing rough pose estimates but are fast to compute and also solve the global localization task.

4.5 Global Localization Techniques

In contrast to its incremental counterpart, global localization techniques do not use any a-priori knowledge about the pose to be estimated and therefore global localization is much more difficult than tracking. Global localization techniques can be divided into two sub-problems: *wake-up problem* and *kidnapped robot problem*. At wake-up the robot is initially placed somewhere in the map without any estimate about its current pose. Transferring the robot during tracking abruptly to some other unknown localization describes the kidnapped robot problem. In contrast to the wake-up problem where the robot is aware of its unknown pose, the robot assumes a wrong pose estimate at the kidnapped robot problem. Of course, one might argue, that nobody ever kidnaps a robot, but it does often occur indirectly in practice: just assume a pose tracking algorithm which fails during operation.

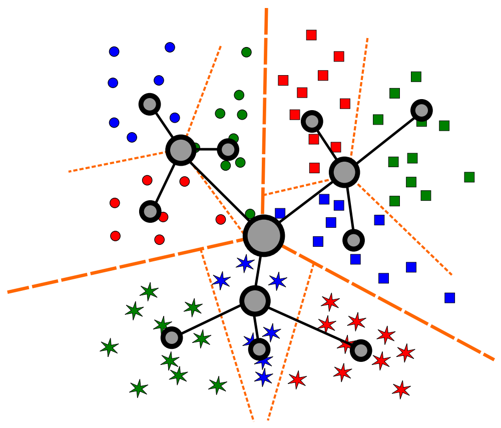
Most image based algorithms are operating on topological maps only consisting of geo-tagged images, an image database or a collection of image features representing

Recursive Filtering Localization	Sensor Data	Map Type	Accuracy	Robustness	Computation time	Global Localization
EKF	Features	Landmark Map	medium	low	fast	no
UKF	Features	Landmark Map	medium	low	fast	no
Topological grid	Features	Topological map	low	medium	medium	yes
Metric grid	Raw data, Features	Occupancy Grid, Landmark map	medium	high	slow	yes
MCL	Raw data, Features	Occupancy Grid, Landmark map	high	high	medium	yes
Optimization based Localization						
Point based	Features	Landmark Map	high	high	medium	no
Image based	Raw data, Features	Topological Map	low	high	fast	yes

Table 4.2: Localization techniques. We distinguish between recursive filtering and optimization based localization approaches. We summarized some characteristics like sensor data used, the underlying map representation and pose estimation accuracy. The computation time includes feature extraction, data association and pose estimation. The last column indicates the algorithms ability to solve the global localization task.

a specific place (vocabulary tree). Consider a topological map composed out of 1.000 nodes where each node is represented by a collection of 50.000 features. Given a query image each feature has to be compared against 50.000.000 descriptors of the map in order to find the most similar node in the graph. This would lead to problems storing the map (e.g. $50.000 \cdot 1.000 \cdot 128 = 6.400.000.000\,bytes \approx 6.4GB$ in case of SIFT features) and is computationally infeasible when doing exhaustive search. As a consequence, an effective data structure is needed to search a large database (nodes) in a high-dimensional space (descriptor length). In order to solve this problem most existing approaches are inspired by the bag-of-words approach for object or place recognition [116, 97] denoted as *vocabulary tree*.

The vocabulary tree is defined by k clusters at L levels, where k defines the branching and cluster factor and L the number of levels. The clusters are built during an unsupervised offline training phase using a large set of arbitrary image descriptor vectors. First, k-means clustering is performed on the whole set resulting in k cluster centers defining k nodes at the first level. Second, each descriptor is assigned to the closest cluster center resulting in k groups. The k-means procedure is then applied recursively to each group again. This results in a total of k^L leaf nodes denoted as visual words. The final tree can be interpreted as nested Voronoi cells as shown in Figure 4.15(a).


(a)

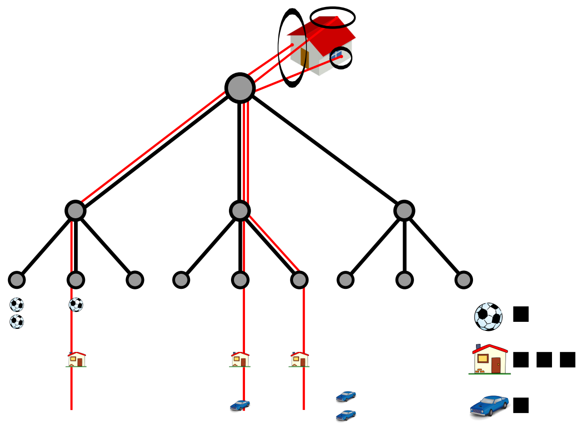

(b)

Figure 4.15: Vocabulary tree. (a) Vocabulary tree with branching factor $k = 3$ and two levels $L = 2$ visualized as Voronoi diagram. (b) Vocabulary tree where three images (ball, house, car) with three descriptors each have been added. The query image (house containing three features) is passed down the tree and the paths (red) are compared resulting in three votes for the house and one correlation with ball and car, respectively.

During operation the trained tree is used to compute a single integer for every input descriptor: Hereby, the descriptor is compared to the first k cluster centers at the first level and the closest one is determined by a simple dot product. This procedure is applied recursively down the tree. The path down the tree is stored as a single integer which is used for scoring later on. Furthermore, each node in the tree stores the features from whom it has been visited. This so called inverted file structure speeds up the query process later on. In order to determine the similarity of a query image to those already in the database, all features extracted out of the query image are propagated down the tree and their feature paths are compared to those already in the tree. The image with the most similar paths is reported as a match. Figure 4.15(b) shows a vocabulary tree, where three images (ball, house, car) have already been inserted. For a given query image (house) the most similar one has to be found. Therefore, three SIFT descriptors are propagated down the tree (red lines) ending up in three different leaf nodes, where the house is visited most (black scoring dots on the right). The inverted file structure speeds up similarity computation since only images stored in the passed nodes have to be taken into account. The similarity itself is computed by different scoring functions such as Jacard Scoring or term frequency-inverse document frequency (tf-idf). For a detailed description we refer the reader to [116, 97].

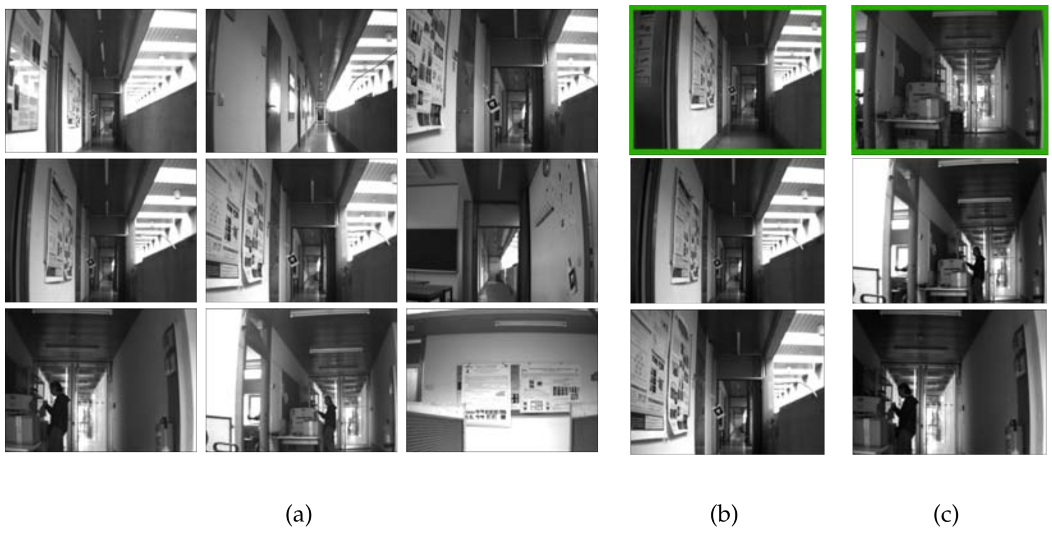


Figure 4.16: Vocabulary tree queries. (a) Nine images are already in the database. (b)(c) Query images are marked green and their two most similar images from the database are presented.

Regarding memory consumption and computation time we employ the same example as above. Assuming SIFT descriptors a vocabulary tree only requires $128 \cdot k^L$ bytes of storage (e.g. 128 MB for $k = 10, L = 6$). A query for a single descriptor takes only $k \cdot L$ dot products plus some effort for scoring. Figure 4.16 shows the two best matches for the query images depicted on the top right, where the images on the left have already been in the database.

In literature Sivic and Zisserman [116] or Nister and Stewenius [97] only concentrated on small image datasets for object recognition. Later on Schindler et al. [113] use the most informative features only to boost the retrieval performance in a topological map consisting of more than 30000 streetside images. More recently Cummins et al. [22, 23] also capture the inference *between* visual words by constructing a Chow-Lui tree resulting in a more robust place recognition system especially when dealing with a huge amount of data (103256 images) and repetitive structures over a long trajectory (1000 km).

All mentioned algorithms return a probability for each image/place already in the map, describing the similarity to the query image. Hence, they do not return a specific pose in the map but probabilities for the most likely places. Because of the low memory requirement and the fast computation time, a vocabulary tree is quite often used for loop closure detection. Even if pose estimation during pose tracking fails, a database query provides a new starting point to continue tracking again.

4.6 A Complete Visual SLAM Algorithm

This section describes our complete framework performing localization and map building using image data only. As investigated by Strasdat et al. [128] sliding window bundle adjustment should be preferred over recursive filtering techniques in monocular SLAM. Therefore, we decided to implement an incremental SfM algorithm operating on keyframes and building a sparse three-dimensional pointcloud map. The different building blocks of our visual SLAM framework are depicted in Figure 4.17 and described in more detail in the subsequent sections.

From all beforehand mentioned approaches performing localization and mapping we use those best suitable for a visual SLAM system, which should meet the following requirements:

Continuous localization and map building A visual SLAM algorithm should perform continuous, incremental localization and map building over a long period of time in large scale environments. We propose to use keyframes (the pose of an image at a specific point in time used for map expansion) and a sparse three-dimensional pointcloud together with sliding window bundle adjustment because of the following reasons: First, only heuristically selected keyframes are used for map building and pose estimation, since subsequent frames contain a lot of redundant information not necessary during bundle adjustment optimization. As a consequence repeated optimization over a sliding window of a predefined size becomes practicable and results in accurate pose estimates (compare [128]). Assuming the robot moves with constant, moderate velocity the localization process is invoked more frequently than the map building process. Second, to model an unstructured environment a sparse pointcloud map (see Section 4.3.1) is the best choice since we do not want to make any assumptions about the structure of the surroundings (e.g. planar, orthogonal indoor buildings). Although, planes or lines would consume less memory they are not able to model arbitrary surroundings.

Robust data association Each landmark is a highly distinctive description attached to facilitate data association during pose tracking. We decided to use the rotation and scale invariant SIFT descriptors [74]. Although features like image patches are faster and easier to compute they are less distinctive and not as reliable as SIFT descriptors (see Section 4.2). Moreover, existing GPU implementations [151] allow fast SIFT descriptor computation and matching.

Recovery and relocalization In case of tracking failures or when reentering the map at a different point in time, a robust method for image based place recognition is necessary. We borrowed the concepts used in topological SLAM and organize extracted features in a vocabulary tree, as described in Section 4.5, since global localization does not need any a-priori information about the pose to be estimated. Furthermore, a vocabulary tree provides an elegant way to store a huge amount of data (e.g. many images) and allows for fast database queries resulting in rough pose estimates, which can be used as a new starting point for position tracking.

Loop closure detection and correction The system should continuously check for loop closures and automatically correct the drift induced by visual odometry. Loop closure detection is performed by querying the vocabulary tree everytime a keyframe is established. After a positive loop closure detection structure and motion optimization over the whole map and trajectory is performed.

The workflow of our visual SLAM system using SfM for trajectory and landmark estimation is visualized in Figure 4.17. After initializing the map from two images, a

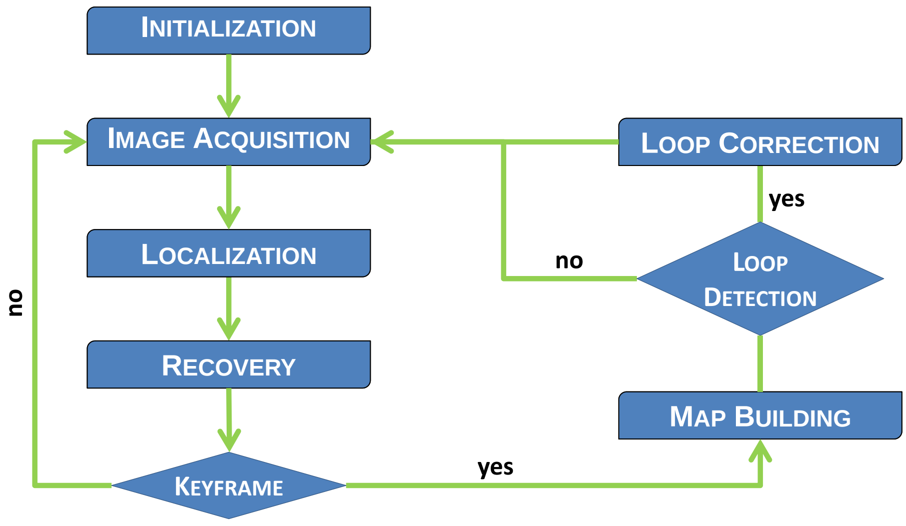


Figure 4.17: The proposed visual SLAM workflow. After map initialization we grab a new image from the camera device and perform pose tracking. If tracking failed a recovery routine re-localizes the current frame. If the sensor has moved a certain amount we start exploring new terrain, establish a new keyframe and extend the map, otherwise the next image is processed. If we visit a previously seen place (loop detection) structure and motion refinement over the whole trajectory is performed.

new frame is processed and localization with respect to the given map is performed. In case of localization failure a recovery routine is started performing global localization. If there is no need for a new keyframe the next frame is processed. Otherwise, the map is expanded by new landmarks. In case of a loop closure the whole map and trajectory has to be corrected. In the following we introduce the representation of our environment and its notation. Finally, the building blocks as shown in Figure 4.17 are described more detailed.

4.6.1 Environment Model

Our environment is represented by a set of *landmarks* (also called map points) $\mathbf{X_j}$ and *keyframes* (also called camera poses) $\mathbf{P_i}$, located in a global world coordinate frame. The j-th map point is represented by its three-dimensional coordinates $\mathbf{X_j} = (X_j, Y_j, Z_j, 1)^T$ in homogeneous form. Each map point is associated with a *descriptor* d_j used for data association. Additionally, a set of image *measurements* $\mathbf{m_{ij}}$ related to the j-th landmark is stored. Every single measurement (also called observation) describes a local interest point descriptor (SIFT) d_{ij} as well as its two-dimensional image coordinates $x_{ij} = (u, v, 1)^T_{ij}$ in keyframe $\mathbf{P_i}$. Every map point comes along with at least two image measurements, from whom it has been triangulated. The i-th camera pose is parametrized by a standard projection matrix

$$\mathbf{P_i} = K\,[I_{3\times3}|0_{3\times1}]\begin{bmatrix} R_i & t_i \\ 0^T & 1 \end{bmatrix} \tag{4.6}$$

composed of a 3×3 rotation matrix R_i and 3×1 translation vector t_i defining the pose and orientation of the camera in the global world coordinate frame. The internal calibration matrix K is determined by a preceding calibration process and therefore assumed to be known in advance.

Map point $\mathbf{X_j}$ is projected onto the image plane of keyframe $\mathbf{P_i}$ with the projection equation

$$\hat{x}_{ij} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_{ij} = \left(\mathbf{P_i}\,\mathbf{X_j}\right)_n \tag{4.7}$$

where $\hat{x}_{ij}$ is called the *prediction* and $(.)_n$ denotes the conversion from homogeneous to Euclidean coordinates. Additionally, all features extracted from the keyframes are stored in a vocabulary tree $\mathbf{V}$ as described in Section 4.5 for fast loop closure detection and relocalization. The parametrization of our environment is summarized in Table 4.3.

$\mathbf{X_j}$	$(X_j, Y_j, Z_j, 1)^T$	homogeneous, three-dimensional landmark coordinates
	d_j	descriptor associated with $\mathbf{X_j}$ used for data association
$\mathbf{m_{ij}}$	$(u, v, 1)^T_{ij}$	set of image measurements related do $\mathbf{X_j}$ described by homogeneous image coordinates in keyframe $\mathbf{P_i}$
$\mathbf{P_i}$	$K\,[I_{3\times3}\|0_{3\times1}]\begin{bmatrix} R_i & t_i \\ 0^T & 1 \end{bmatrix}$	intrinsic K and extrinsics R_i, t_i parameters define the projection matrix for a keyframe
$\mathbf{V}$		vocabulary tree organizing the image features of every keyframe

Table 4.3: Environment representation. Our map is composed of map points $\mathbf{X_j}$ and keyframes $\mathbf{P_i}$ located in a global world coordinate frame. Each map point is associated with two or more measurements $\mathbf{m_{ij}}$ in image $\mathbf{P_i}$.

Modeling the environment as a set of sparsely distributed three-dimensional points is more memory efficient than dense occupancy grid maps or triangle meshes and allows the modeling of arbitrary environments in contrast to planes, lines or any other geometric primitive. Moreover, we are not restricted to some specific structure of the surrounding (e.g. orthogonal planes) and sparse pointclouds fit well into the SfM process. The main issues using sparse map points are: How many points should be triangulated to guarantee robust localization and where should we add new points to expand the map.

Using images as input one could add every incoming frame to the traveled trajectory, but in practice subsequent images along a robot path contain lot of redundant information. Hence, the concept of using keyframes is much more memory efficient and still allows robust localization and map expansion as shown by Klein and Murray [63]. The difficulties lie in when or where to add a new keyframe to the map to guarantee enough image overlap to preceding frames and robust triangulation of new landmarks. If too many camera poses are established memory consumption increases. In case of insufficient keyframes visual connectivity is lost resulting in localization failures and distorted maps.

4.6.2 Visual Localization and Mapping Framework

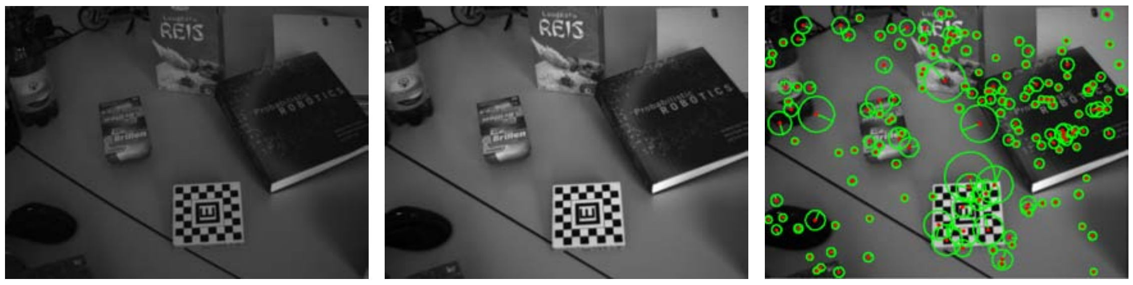
Based on the environmental representation, localization and map building is handled by an incremental SfM framework together with a vocabulary tree for loop closure detection and relocalization. In the following the different steps involved in our visual SLAM process as shown in Figure 4.17 are described.

4.6.2.1 Image Acquisition

Image acquisition involves grabbing a new frame $\mathbf{P_i}$ from the camera device followed by undistortion using the coefficients determined in an offline calibration process. Afterwards, SIFT [74] keypoints x_i and their descriptors d_i are extracted using a fast GPU implementation [151].

4.6.2.2 Initialization

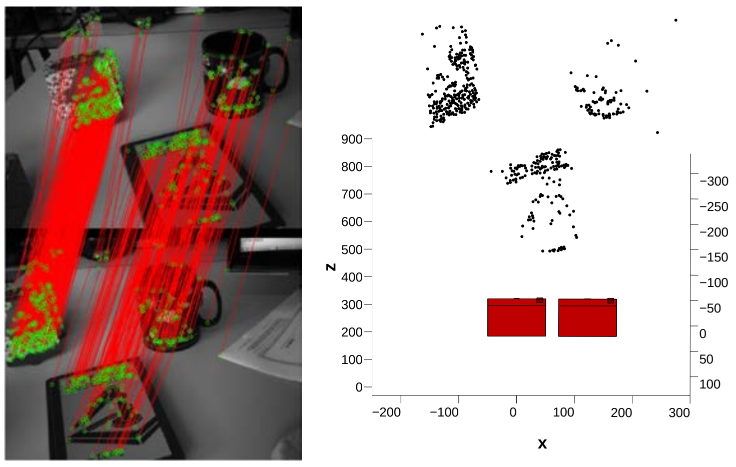
To initialize we enforce the user to acquire two images exhibiting enough baseline for accurate two-view triangulation later on. Using the previously extracted keypoints, SIFT descriptor matching results in $2D \leftrightarrow 2D$ correspondences. The eight-point algorithm [**?**]



(a) (b) (c)

Figure 4.18: Image acquisition and processing. (a) Image grabbed from the camera device. (b) After distortion correction using the parameters from an offline calibration process. (c) Extracted SIFT keypoints (red) and their descriptors (green), which are used for data association and map point generation.

together with a RANSAC routine is used to estimate the exterior orientation of both cameras, where the world coordinate frame origin resides with the first camera. Both sensor poses form the initial set of keyframes in the map and landmarks are established through two-view triangulation.



(a) (b)

Figure 4.19: Map and keyframe initialization. (a) Two input image chosen by the user and their corresponding image points. (b) Using the eight-point-algorithm this results in map points (black) and camera poses (red), which define the initial environment. The origin of the whole reconstruction resides in the first camera.

In case of a stereo image device no initialization process is necessary since three-dimensional information is provided directly by the sensor.

4.6.2.3 Incremental Localization

Assuming a sparse pointcloud $\mathbf{X_j}$ and the pose $\mathbf{P_{i-1}}$ of the latest keyframe, incremental localization aims at computing the current pose estimate $\mathbf{P_i}$ given a single, undistorted image and its extracted keypoints x_i and descriptors d_i. As described in Section 4.4 this comprises a prediction and correction step:

Prediction To predict a pose $\mathbf{\hat{P}_i}$ interframe motion has to be estimated. Given a range-image device camera motion is estimated by using a three-point pose estimation method [50] embedded in a RANSAC [36] algorithm. Hereby, $3D \leftrightarrow 3D$ point correspondences established through feature matching are used to compute a rigid body transformation describing the camera motion between two consecutive frames.

When dealing with monocular input some kind of motion model needs to be defined approximating the sensor motion. Here, we simply ignore interframe motion and set $\mathbf{\hat{P}_i} = \mathbf{P_{i-1}}$. Since images are captured at high framerates this is a sufficient initialization for the subsequent correction step.

Correction To refine the estimate $\mathbf{\hat{P}_i}$ to be correctly aligned with the map two steps are necessary: First, all map points within the view frustum of $\mathbf{\hat{P}_i}$ are determined. Therefore, each landmark $\mathbf{X_j}$ is projected onto the image using Equation 4.7 and each landmark whose projection resides within the image borders forms the potentially visible set $\mathbf{\tilde{X}_j}$. Their associated descriptors $\tilde{d}_j$ are compared against those extracted from $\mathbf{P_i}$ defining the $2D \leftrightarrow 3D$ correspondences $x_i \leftrightarrow \mathbf{\tilde{X}_j}$. Second, given at least three correspondences the camera pose can be computed through a least-squares optimization routine by minimizing the reprojection error:

$$\mathbf{P_i} = \arg\min_{\hat{P}_i} \; d\left(x_i, \mathbf{\hat{P}_i}\,\mathbf{\tilde{X}_j}\right), \tag{4.8}$$

where $d(.)$ denotes the Euclidean distance. To be more robust against matching outliers a robust error norm (compare Section 3.2.2) within a reweighted least-squares solver is used resulting in a corrected pose $\mathbf{P_i}$ after at most 10 iterations.

Incremental localization is done as long there is no need for another keyframe.

4.6.2.4 Map Building

Map building involves the decision when to add a new keyframe to the map, the construction of new map points and the refinement of both structure and motion through bundle adjustment.

For simplicity the tracked pose $\mathbf{P_i}$ becomes a new keyframe $\mathbf{P_k}$, if the camera has moved a certain amount (e.g. $12cm$ or 5°), which has been used by many authors before in localization and mapping [63]. This rule guarantees enough overlap to previous keyframes, which is an essential prerequisite for subsequent bundle adjustment and the construction of new map points.

In order to construct new map points each keypoint x_i lacking a positive match during tracking is considered for map expansion. To establish corresponding keypoints of the current frame are matched against the descriptors in the last N keyframes followed by a geometric verification using epipolar geometry. To establish new landmarks multi-view triangulation is performed. The keyframe, newly created map points, their descriptors and newly found measurements in the last N frames are added to the environment. All image descriptors stemming from $\mathbf{P_k}$ are inserted into the vocabulary tree $\mathbf{V}$ representing a new place.

Since optimizing the complete map is computationally demanding, only structure and motion over the last N keyframes are refined by applying sparse bundle adjustment as described in Section 3.2.2. The keyframes are parametrized by three Euler angles (see Section 3.1.4) and the 3×1 translation vector resulting in 6 parameters to be optimized per keyframe. The map point is represented by its three Euclidean coordinates.

As long as no loop closure is detected, incremental localization and map building are performed in an alternating manner.

4.6.2.5 Global Localization

The global localization task consists of a loop closure check for every keyframe, a recovery task in case of a tracking failure or when reentering a known map. The descriptors of every keyframe are inserted into the pre-trained vocabulary tree $\mathbf{V}$ as described in Section 4.5. One can use the approach proposed by Nister et al. [97] or the FAB-MAP algorithm proposed by Cummins et al. [23]. Both implementations are able to detect revisited places over long time differences and are robust against lighting changes. The descriptors d_i of the currently processed frame $\mathbf{P_i}$ are propagated down the tree returning the M most similar keyframes. Clearly, the result always contains recently added keyframes, as well as keyframes established a long time before. While the former ones can always be safely rejected, the loop closures frames may be polluted by outliers due to the quantization effects of the vocabulary tree. They are geometrically verified by establishing $2D \leftrightarrow 2D$ correspondences between the current frame and the potential candidate. A RANSAC routine is used to estimate the fundamental matrix F. The candidate producing the most inliers while estimating F is regarded as the potential loop closing frame $\mathbf{P}_l$.

The query results for the 8 most similar frames in a loop are shown in Figure 4.20. The x-axis denotes the query images and the y-axis denotes the indices of the 8 most similar keyframes. Up to keyframe 240 the query results are mostly spatial neighbors of the current keyframe (e.g. diagonal elements). After frame 240, a second parallel line emerges indicating a loop closure. Figure 4.21 shows the results of loop closure detection for the candidate frames marked blue in Figure 4.20. Hereby, the candidate with the highest number of inliers reported from the RANSAC routine is assigned to be the loop closing frame $\mathbf{P}_l$ (framed red).

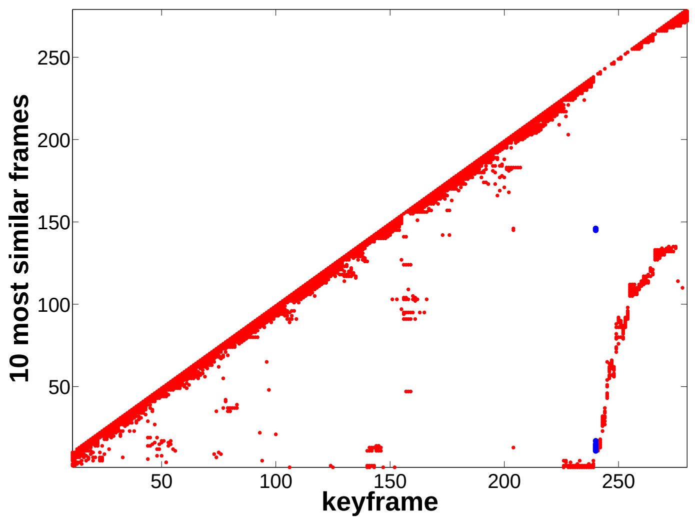


Figure 4.20: Vocabulary tree query results. The x-axis denotes the current query frame while the y-axis shows the keyframe indices of the 8 most similar neighbors. Recently added keyframes (diagonal elements) are ignored during loop closure detection since they are in the spatial neighborhood of the query frame. The remaining candidates are geometrically verified.

In case of relocalization or recovery, the pose is estimated by a non-linear least-squares optimization routine minimizing the reprojection error, initialized with the best

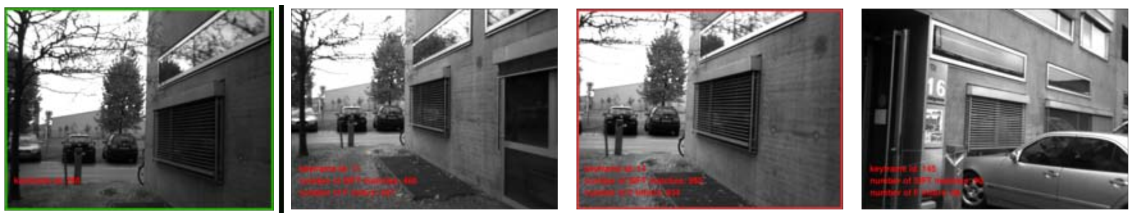


Figure 4.21: Geometric verification of the loop closure candidates for the query image framed green. Wrong voctree responses are filtered out through descriptor matching followed by fundamental matrix estimation. The keyframe producing the most RANSAC inliers is reported to be the loop closing frame $\mathbf{P_l}$ (framed red).

matching keyframe $\mathbf{P_l}$:

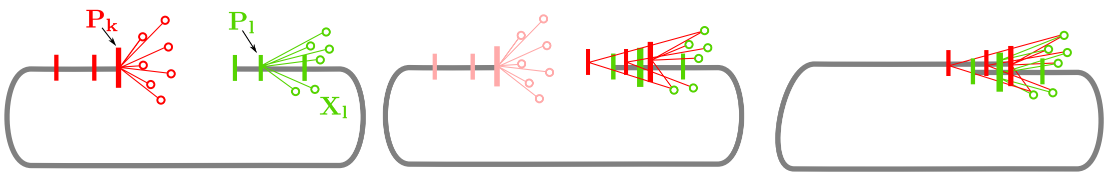
$$\mathbf{P_k} = \arg\min_{\mathbf{P_l}} \; d\left(x_k, \mathbf{P_l}\,\mathbf{X_l}\right). \tag{4.9}$$

After successful recovery incremental localization is continued.

In contrast to relocalization, loop closures require special treatment. To close the loop and to guarantee convergence of the subsequent bundle adjustment routine, as many correspondences as possible $x_k \leftrightarrow \mathbf{X_l}$ between the current frame $\mathbf{P_k}$ and map points visible from $\mathbf{P_l}$ have to established. Therefore, descriptors associated with the set of map points $\mathbf{X^*{}_k}$ visible from $\mathbf{P_k}$ are compared to the set of map points $\mathbf{X^*{}_l}$ visible by the loop closing frame $\mathbf{P_l}$. If there exists a positive match between two points $\mathbf{X_k}$ and $\mathbf{X_l}$, all measurements m_{kk} related to $\mathbf{X_k}$ are added as additional measurements for map point $\mathbf{X_l}$ and the duplicate landmark $\mathbf{X_k}$ is deleted. A simplified sketch of the loop closing process is shown in Figure 4.22. Once enough correspondences are detected, sparse bundle adjustment is applied over the entire map and keyframe poses. In contrast to iterative map optimization we use the least-squares cost function since the newly added measurements result in high reprojection errors. Hence, they would be treated as outliers by a robust cost function. The results of the loop closing process are shown in Figure 4.23, where the top-view of map and trajectory before and after loop closure detection and correction is shown.

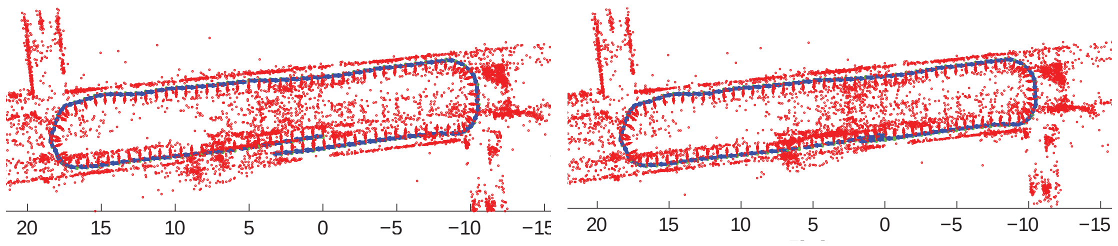
4.7 Conclusion

Early in this chapter we presented three typical visual features used in robotics. Then we provided an intensive state-of-the-art overview on different environment models, map building algorithms as well as localization techniques within the map. We put special



(a) (b) (c)

Figure 4.22: Schematic loop closing process. (a) Before loop closing the green points are visible at the beginning and the red points at the end of the loop. (b) After determining the potential loop closing frame $\mathbf{P_l}$, $2D \leftrightarrow 3D$ point correspondences between $\mathbf{P_k}$ and map points visible from $\mathbf{P_l}$ are established. (c) Sparse bundle adjustment over the whole loop is performed to correct all sensor poses and landmarks.

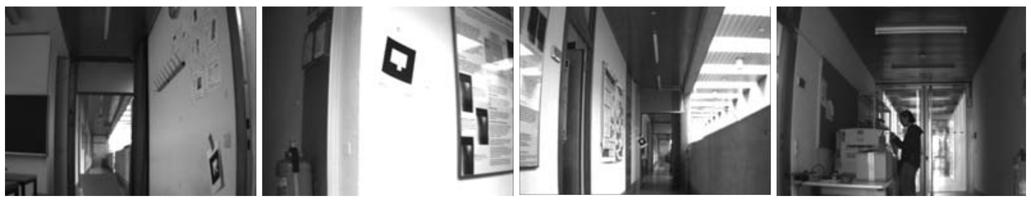


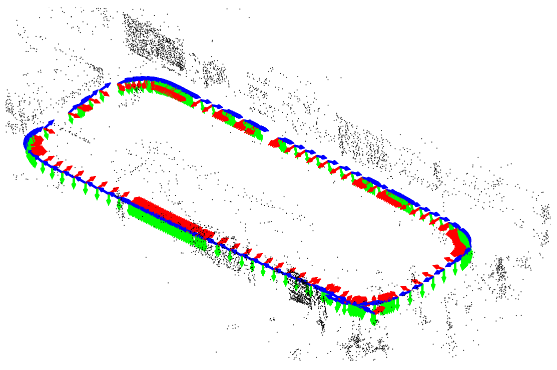
(a) (b)

Figure 4.23: Loop closure correction. (a) Top view of a three-dimensional map (red) and a $70m$ trajectory (blue) before loop closing took place, where a small displacement occurs. (b) Corrected map and trajectory after applying bundle adjustment over the whole environment.

emphasis on global localization techniques using a vision sensor only, since it is a basic prerequisite for every SLAM algorithm. Based on these considerations we implemented our own framework capable of doing simultaneous localization and map building using either a monocular camera or a rang8e image device. We also presented techniques to recover from localization failures and for robust loop closure detection and correction.

The proposed visual SLAM algorithm is able to perform localization and map building within static environments resulting in a set of camera poses and a sparse three-dimensional pointcloud map of the environment. Figure 4.24 shows the resulting environment and estimated trajectory of a typical indoor scenery, where we used stereo-data as input and traveled a closed loop. Even short-term noise caused by moving people is handled implicitly by the robust cost function during bundle adjustment. If the proposed algorithm should perform visual SLAM over a long period of time in a large scale environment containing dynamic scene elements several modifications would be necessary: First, the current heuristic used for keyframe selection and landmark creation leads to a steadily increasing map size. This involves high memory consumption and computation time. Hence, a more sophisticated rule for keyframe selection would be necessary and map points should be added at unexplored regions only. Second, the presented loop closing routine becomes computationally infeasible when traveling in larger environments since a huge amount of parameters would be optimized. Finally, during map building we assumed a static environment. This assumption only holds for laboratory conditions or single run reconstructions but not for many real-world applications. A robot traveling in everyday scenarios is confronted with an ever changing world consisting of moved furniture, driving cars or lighting variations. A mapping algorithm operating in a dynamic world should react on these changes and offer the possibility to update and repair an existing map. The following section focuses on localization and map building in dynamic environments and comprehends some solutions to tackle these problems.


(a)


(b)

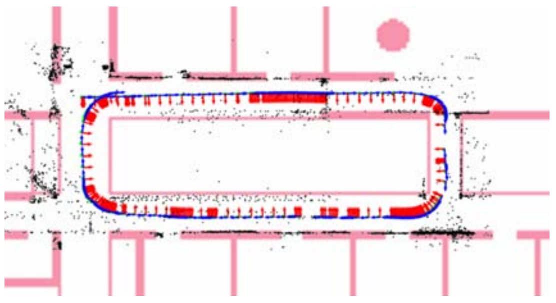

(c)

Figure 4.24: Results of an indoor environment. (a) Images recorded in a typical office building. (b) The resulting three-dimensional map and trajectory. (b) Top view of map and trajectory overlaid on a floor plan.

5 Visual Localization and Map Building in Dynamic Environments

In the previous chapter we described the different blocks necessary to implement a SLAM algorithm using vision sensors. Assuming a static environment the proposed algorithm is able to construct a three-dimensional representation of the environment while simultaneously localizing within this world. In visual SLAM, as presented in Section 4.6, map construction is done in a single run only, without considering any environmental changes. Simply adding all incoming sensor data would involve increased memory consumption, inconsistent maps and data association problems - especially in high dynamic scenes. Consequently, SLAM would inevitably fail in the long run.

This chapter addresses exactly these problems. At the beginning we try to see our environment from a robots point of view and highlight the challenges and problems when performing visual SLAM in a real-world scenario. We propose two main contributions to tackle the problems arising in dynamic surroundings: First, a novel three-dimensional landmark descriptor is developed in Section 5.2, which allows us to react on dynamic entities. The descriptor stores feature visibility information and implicitly filters out-of-date map regions. Second, a graph-based keyframe organization is presented in Section 5.3 to gain efficiency and scalability in loop closing and sliding window bundle adjustment. We show how to embed them into our visual SLAM algorithm depicted in Section 4.6 and highlight their benefits.

5.1 Introduction and Motivation

5.1.1 What is a Dynamic World?

Most state-of-the-art works on visual SLAM assume a well-structured environment containing stationary objects only and input data is captured in a single run. For a robot moving in real-world environments over long periods of time this assumption does not hold anymore. People are moving around resulting in noisy sensor readings. Simultaneously they also modify the surrounding by adding, moving or deleting objects such as furniture, packages, plants, posters, desks or chairs leading to undefined changes in the robots sensor readings.

In Figure 5.1 we define the different states of a realistic, everyday environment a robot is moving in, where we can distinguish five categories: There exists a lot of static objects, which do not change their position and appearance over time. When thinking of a robust SLAM system the map should be made mainly out of stationary objects since these are reliable landmarks for localization. Buildings, walls, windows, floors or locked doors belong to this group. Dynamic scene elements instead are grouped into noisy, radiometric, short-term and long-term dynamics, which undergo different time scales. Noisy dynamics are moving randomly inside the robots field of view and are characterized through unpredictable movements. Leaves moved by the wind or clouds belong to this group. Consequently, landmarks stemming from these objects should not be present in the map or at least not used for localization. Short-term dynamic elements are moving quickly and flighty inside the robots field of view and can be found at many different places in the map. For example pedestrians or cars are moving in front of the robot in real-world traffic scenes. Even in indoor office scenes or museums steadily walking people distort the robots sensor readings. However, as mentioned in Section 2.2, short-term dynamic changes can be robustly filtered out during the SLAM process since they are not important for pose tracking. Static objects probably changing their appearance over time outside the robots field of view are called radiometric dynamics. Appearance changes are mainly caused by seasonal changes (summer, winter, day, night) or varying lighting conditions (artificial lighting, sunshine, shadows). Here, seasonal changes occur at regular time intervals while varying lighting conditions are temporally unpredictable. Objects with shiny surfaces like metallic doors, floors or lights belong to this group. Since these objects do not change their position they represent important landmarks for localization. Long-term dynamic scene elements instead pose a greater challenge for the localization and mapping task. Usually, they move in-

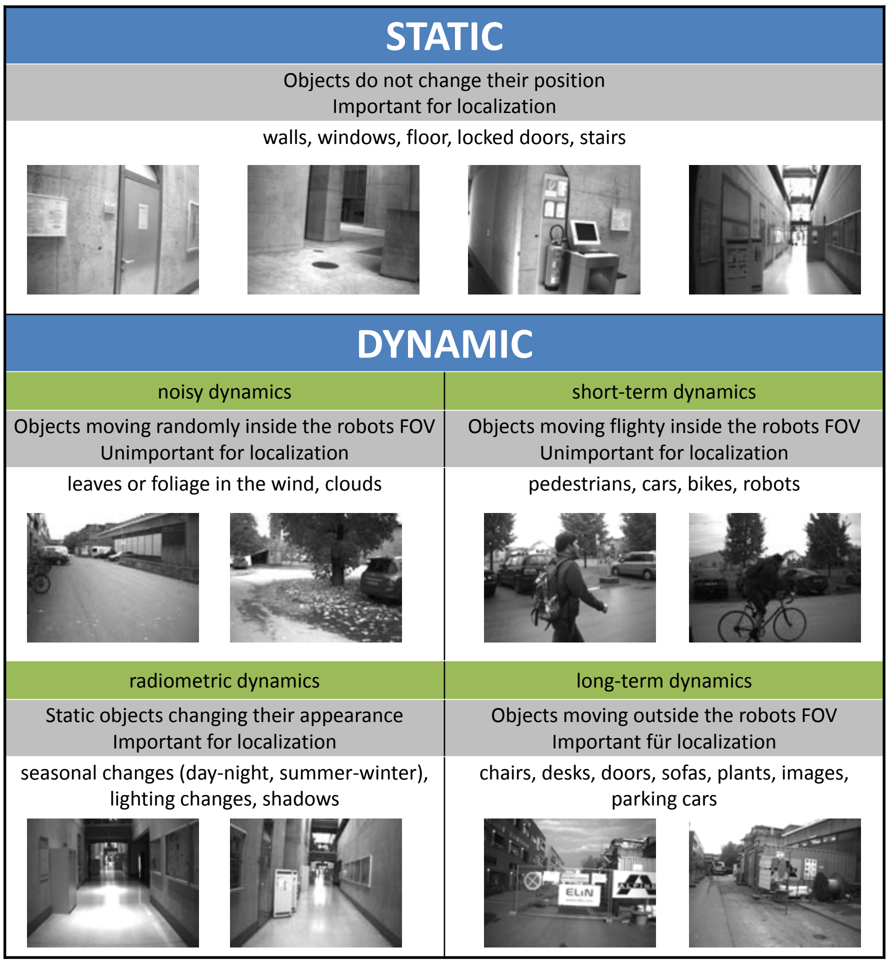

STATIC
Objects do not change their position
Important for localization
walls, windows, floor, locked doors, stairs
DYNAMIC
noisy dynamics
Objects moving randomly inside the robots FOV
Unimportant for localization
leaves or foliage in the wind, clouds
short-term dynamics
Objects moving flighty inside the robots FOV
Unimportant for localization
pedestrians, cars, bikes, robots
radiometric dynamics
Static objects changing their appearance
Important for localization
seasonal changes (day-night, summer-winter), lighting changes, shadows
long-term dynamics
Objects moving outside the robots FOV
Important für localization
chairs, desks, doors, sofas, plants, images, parking cars

Figure 5.1: Categorization of a robots environment. Besides static scene elements a visual SLAM algorithm is confronted with noisy, short-term, radiometric and low-dynamic elements. A complete visual SLAM algorithm should be able to handle all these scene configurations especially in the context of life-long map building.

frequently outside the robots field of view and can only be found at specific places in the map like sofas, posters, images, plants, desks or chairs. For example, a book lying on a desk is regarded stationary when faced by the robots sensor and therefore included in the map as a landmark. Afterwards, it can be moved or deleted outside the robots FOV. Nonetheless they represent important landmarks for localization process. Both, radiometric and long-term dynamic changes can only be detected if the robot revisits the same place again. In that case the same place results in different sensor measurements, which should be used to repair the existing map (i.e. update shifted objects, remove deleted objects or add new objects).

5.1.2 Challenges in Dynamic Environments

Considering this environment categorization a robot is confronted with a lot of challenges to be solved when performing life-long localization and mapping:

Noisy dynamics Randomly, unpredictably moving objects like leaves, grass or trees do not represent reliable landmarks for localization and should not be added to the map. As a consequence they should be filtered out or ignored throughout the whole SLAM procedure. Otherwise, these spurious landmarks would overfill the map and lead to increased storage requirements.

Short-term dynamics Objects moving rapidly inside the robots field of view like driving cars or pedestrians should be filtered out or ignored and not be incorporated into the map. As noisy dynamics, these temporary measurements would lead to an ever growing number of landmarks, a falsified map representation and wrong localization results.

Radiometric dynamics Radiometric entities do not change their position over time, but might alter their appearance induces through external forces. As long as their corresponding landmarks are detected reliably during pose tracking they should be kept in the map. Otherwise, their appearance information should be updated. In order to maintain a consistent and up-to-date map and to guarantee robust localization this process must be done permanently and online.

Long-term dynamics Long-term scene dynamics should be detected by comparing current sensor data to the already existent map. Landmarks corresponding to moved or vanished objects have to be deleted from the map. New entities should be added to the map. As above this process must be done permanently and online guaranteeing an up-to-date map.

Long-term localization and mapping Despite the handling of dynamic objects, localization and mapping over a long period of time comes along with many other challenges. Normally, the number of poses and landmarks grows proportional to the map building time since keyframes and landmarks are continuously added to the map. If the robot performs navigation over a longer period of time (e.g. days, weeks or even months) and revisits several places multiple times this would lead to increased computation time and storage requirements. Therefore, a desirable effect is to keep the map size proportional to explored space rather than navigation time. Additionally, when traveling a long period of time, multiple loop closures may occur. Here, visual information should be used to check continuously for loop closures and to aid in relocalization when tracking fails. A fast and accurate method to solve multiple, nested loops is indispensable.

In the following, two concepts aiding to solve above described challenges are developed.

5.2 A Novel Feature Descriptor for Visual Mapping

To tackle the problems of high scene dynamics, change detection, map update and scalability we developed a new feature descriptor adding spatial visibility information to each three-dimensional landmark. First, the intuition behind the descriptor is motivated in Section 5.2.1 followed by a theoretical description in Section 5.2.2. Finally, we show its incorporation in our visual SLAM framework and highlight its benefits in Section 5.2.5.

5.2.1 Motivation

Most existing SLAM solutions, including the one proposed in Section 4.6, assume a static environment containing only stationary objects. The map is updated with all incoming sensor data and used for localization afterwards. In reality the static world assumption does not hold since the world a robot is moving in contains a lot of non-stationary objects. When aiming at long-term mapping and exploring objects like doors, office desks or furniture may move. Simply adding all incoming sensor data to the map without reacting on environmental changes would continuously enlarge and even distort the map. The key problem thereby is how to handle map features which should be visible from a certain viewpoint, but are actually not observed. While probabilistic approaches like occupancy grids may implicitly solve this problem by downweighting the affected cells, approaches based on sparse local features can not. This results in ever growing maps and may also lead to data association problems caused by multiple occurrences of the same object in the map (e.g. moved sofa).

To implicitly handle the ambiguity between scene dynamics and occlusion, we propose to add spatial visibility information to local map features. To encode the visibility and importance of each three-dimensional landmark in a map, we develop the *Histogram of Oriented Cameras* (HOC) descriptor. The basic idea is, that each descriptor tracks how often its associated landmark has been observed from a specific location. Hence, the constructed map implicitly adapts to dynamic changes during mapping and dynamic entities should not affect localization.

5.2.2 Feature Descriptor

To build the three-dimensional feature descriptor, the volume around each map point is divided into a set of k concentric spheres with logarithmically increasing radii r_i, $i = 1, ..., k$. Hereby, each sphere is approximated through a *Platonic solid* defined by a set of m faces $f_{i,j}$, $j = 1, ..., m$. A *Platonic solid* is a convex, symmetric, regular polyhedron where all faces are congruent regular polygons and all its vertices lie on a common sphere (compare Figure 5.2).

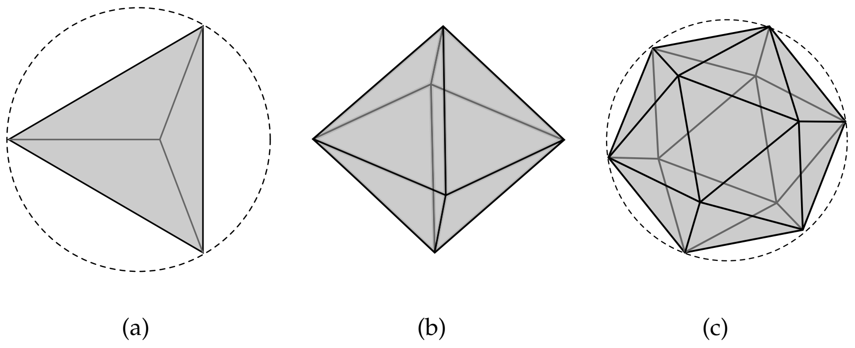


Figure 5.2: Platonic solids with triangular faces, where all vertices lie on a sphere: The three solids are a tetrahedron (a) an octahedron (b) and an icosahedron (c).

The discrete polyhedral discre of the space around each map point can be seen as a three-dimensional histogram. A single histogram bin is defined by the volume $V_{i,j}$ of a pyramid frustum between two consecutive radii r_i and r_{i+1} limited by two corresponding faces $f_{i,j}$ and $f_{i+1,j}$. Figure 5.3 shows a three-dimensional feature descriptor with four spheres approximated by tetrahedra. A single histogram bin is highlighted blue. Each descriptor bin tracks how often its associated landmark has been observed from a camera resting in this bin. Based on this number a probability $p_{i,j}$, that the map point is visible from within that frustum, is estimated. The higher $p_{i,j}$ the more probable it is to observe this map point with a sensor resting in volume $V_{i,j}$. Hence, the view-dependent importance of the landmark encoded in $p_{i,j}$ is defined by *how often* the specific landmark is observed from that viewpoint. A logarithmic spacing of the radii allows us to cover a large volume around each map point, assuming that spatial partitioning is more important for closer features. Additionally, a different feature descriptor may be added to each bin to model view-dependent appearance. Summarized, each descriptor H consists of the following components

$$H = \begin{cases} r_i & i = 1, ..., k \\ f_{i,j} & j = 1, ..., m \\ V_{i,j} = \{(r_i, r_{i+1}], (f_{i,j}, f_{i+1,j})\} \\ p_{i,j} \end{cases} \tag{5.1}$$

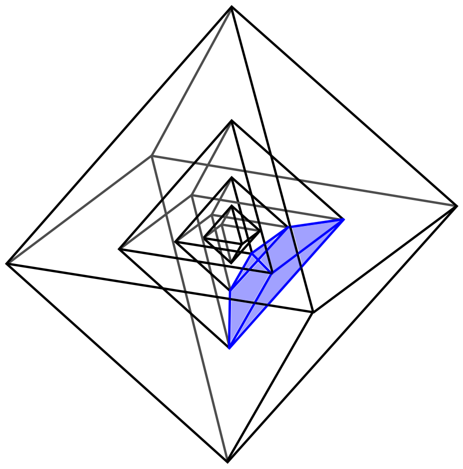


Figure 5.3: Three dimensional feature descriptor. The logarithmically spaced spheres are approximated by octahedron. A histogram bin defined by a pyramidal frustum is highlighted blue.

Every time a new map point is added to the map, its HOC-descriptor is initialized and bins corresponding to the observing camera poses are updated. All remaining bins are marked unseen. Map points and their associated HOC-descriptors already present in the map are only considered for an update if they lie within the view frustum of the current camera pose. The update routine consists of the following steps which are summarized in Algorithm 1: First, the descriptors within the field of view of the current camera pose are determined. Afterwards, the bins (i.e. volumes $V_{i,j}$), the camera is resting in, are computed. Finally, those bins producing a positive match during data association are upweighted while the rest is downweighted.

Figure 5.4 shows the update procedure given various camera positions, two map points and an occluding plane. Both points lie in the view frustum of camera $\mathbf{P_c}$, where the point producing a positive match $\mathbf{X_1}$ is updated (green bin) and the occluded one $\mathbf{X_2}$ is downweighted (red bin).

Algorithm 1 HOC Update Procedure

```
Input: camera pose $\mathbf{P}$, map points
Output: updated HOC descriptors
  for all HOC $H$ in FOV of $\mathbf{P}$ do
    $V_{i,j} \leftarrow computeBin(\mathbf{P}, H)$
    $p_{i,j} \leftarrow updateBin(V_{i,j})$
  end for
```

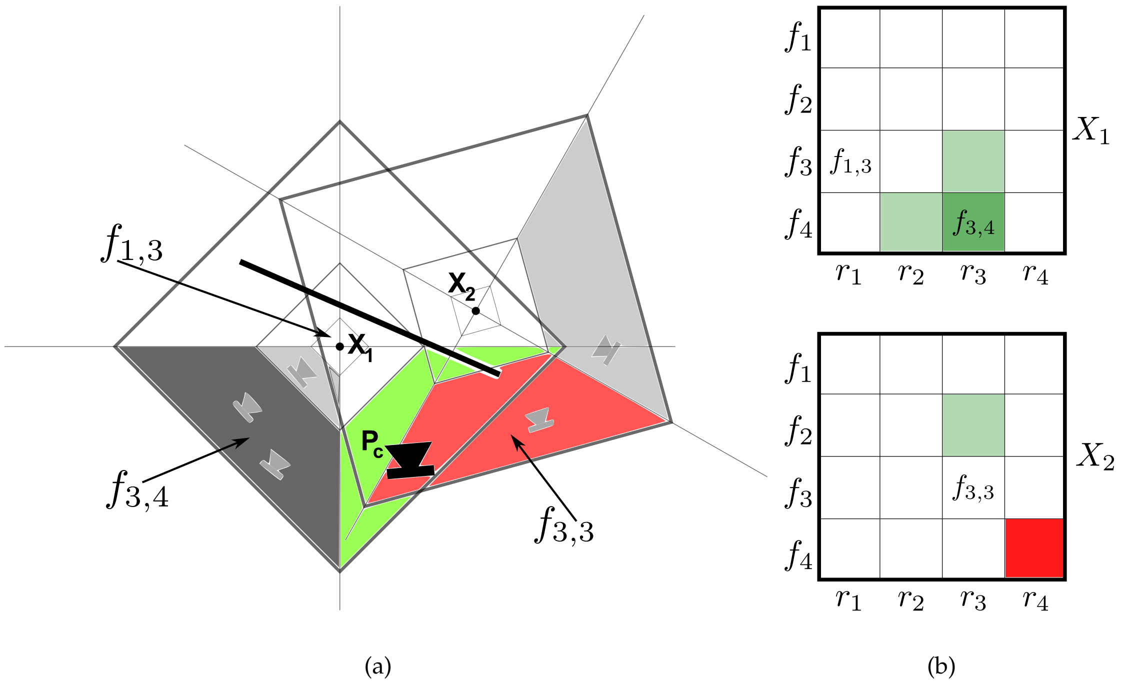


Figure 5.4: HOC update procedure. (a) Map points $\mathbf{X_1}$ and $\mathbf{X_2}$ lie within the view frustum of the current processed camera $\mathbf{P_c}$ and their HOCs are considered for an update. Given an occluding plane, only the bin corresponding to $\mathbf{X_1}$ is upweighted (green) while the other one is downweighted (red). Bins already visited by previous cameras are marked gray, where darker values indicate a higher weighted bin. (b) The associated descriptor weights of points $\mathbf{X_1}$ and $\mathbf{X_2}$ with visibility information encoded red (invisible) and green (visible). The color intensity encodes the reliability of the information.

The above described descriptor allows many different variations regarding appearance and update procedure. In the following, different polyhedral approximations as well as various weighting schemes are described in more detail. How this descriptor is embedded in our visual SLAM framework is addressed in Section 5.2.5.

5.2.3 Descriptor Shapes

Although there exist various platonic solids, we concentrate on icosahedra and octahedra because of the spatial resolution of the bins. Although upsampling through triangle dissection would result in more fine-grained bins, it turned out that the number faces of an icosahedron or octahedron are sufficient to handle dynamic scene elements. In the following both shapes are described in more detail and efficient algorithms for bin computation are presented.

Icosahedron

Each sphere is approximated by an icosahedron consisting of twenty faces represented as equal sized triangles (see Figure 5.2). Given a camera pose $\mathbf{P} = K[R|t]$ and a HOC-descriptor H within its field of view, the associated histogram bin $V_{i,j}$, $j = 1, ..., 20$ is computed the following way: First, we determine the corresponding radius interval $(r_i, r_{i+1}]$ by calculating the Euclidean distance d between camera center $C = -R^T \cdot t$ and map point $\mathbf{X}$ associated to H:

$$d = \|C - \mathbf{X}\| \tag{5.2}$$

$$r_i < d \leq r_{i+1}. \tag{5.3}$$

Second, the intersection between the icosahedron and the line segment from map point to camera center returns the valid face f_j. For fast triangle intersection the circumcenters of each triangle are organized in a kd-tree. After projecting the camera center onto the sphere a nearest-neighbor search in the tree returns the corresponding face. It has to be mentioned that the kd-tree is identical for all HOC-descriptors and needs to be stored only once. Computed radii and the two corresponding faces uniquely define the bins frustum $V_{i,j}$. The bin verification given an icosahedron is summarized in Algorithm 2.

Algorithm 2 Icosahedron - computeBin

```
Input: camera pose $\mathbf{P} = K[R|t]$, HOC descriptor $H$ associated to $\mathbf{X}$
Output: bin $V_{i,j}$
  $C = -R^t \cdot t$
  $d = \|C - \mathbf{X}\|$
  $i \leftarrow r_i < d \leq r_{i+1}$
  $\hat{C} \leftarrow sphereProject(C)$
  $j \leftarrow kdTreeNN(\hat{C})$
  $V_{i,j} = \{(r_i, r_{i+1}], (f_{i,j}, f_{i+1,j})\}$
```

Because of the rather fine approximation of the sphere consisting of twenty faces we propose to use this type of descriptor for small environments like office or desktop scenes. Even if the scenery consists of tiny, dynamic objects like telephones, cups or books this representation should be preferred over the octahedron.

Octahedron

One drawback of the above described sphere approximation is the computationally intensive identification of the histogram bins each time a descriptor is considered for an update. To overcome this problem we propose to use an octahedron as underlying geometry. Hereby, each histogram bin is defined by one octant in the descriptor coordinate system $O = \{E_1, E_2, E_3\}$. The octant domains are defined by three orthogonal planes E_1, E_2, E_3 in homogeneous form. Figure 5.5 shows the descriptor with its three planes bordering a single octant. Given the camera center $C = -R^T \cdot t$ a simple formula delivers the binary representation of the corresponding face index j

$$j = \frac{sgn\left(C^T \cdot [E_1 E_2 E_3]\right) + 1}{2}, \tag{5.4}$$

which can be computed in constant time. To allow each descriptor its own coordinate system we decided to use the three orthogonal planes. The proposed procedure is again summarized in Algorithm 3.

Algorithm 3 Octahedron - computeBin

```
Input: camera pose $\mathbf{P} = K[R|t]$, HOC descriptor $H$ associated to $\mathbf{X}$
Output: bin $V_{i,j}$
  $C = -R^t \cdot t$
  $d = \|C - \mathbf{X}\|$
  $i \leftarrow r_i < d \leq r_{i+1}$
  $j = \frac{sgn\left(C^T \cdot [E_1 E_2 E_3]\right)+1}{2}$
  $V_{i,j} = \{(r_i, r_{i+1}], (f_{i,j}, f_{i+1,j})\}$
```

5.2.4 Bin Weighting Schemes

As described in Section 5.2.2 each bin holds its visibility information in terms of a probability. In this section two possible weighting schemes to compute a bins probability are presented.

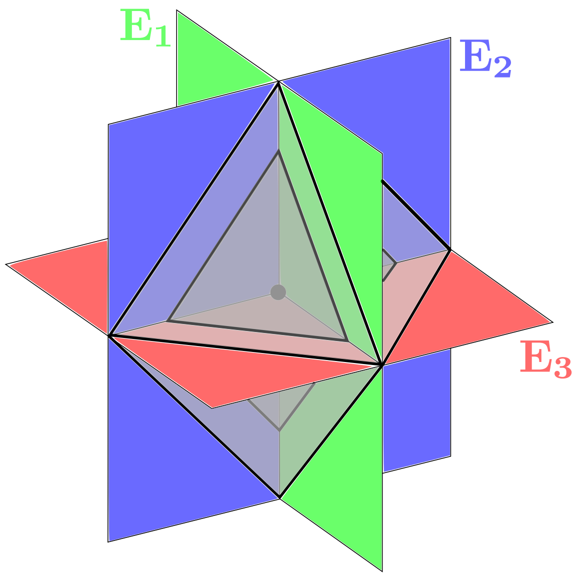


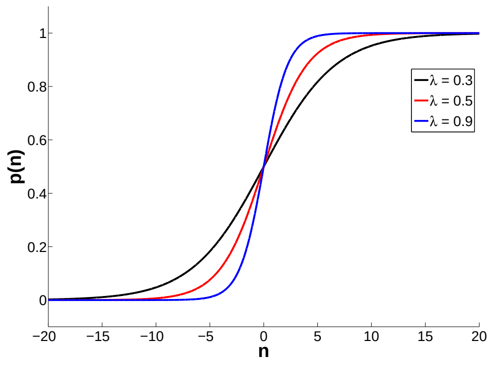
Figure 5.5: Octahedral representation. For computational reasons each sphere is approximated by an octahedron, where bin computation given a camera center can be computed in constant time using the three orthogonal planes E_1, E_2, E_3 defining the descriptor coordinate system.

A Counting Scheme for HOC Update

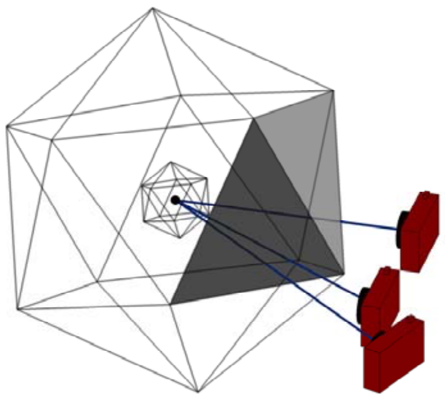
Each bins probability $p_{i,j}$ should reflect the importance of the specific landmark from within this frustum. The easiest way is an observation counter where each bin holds an integer $n_{i,j}$, corresponding to the number of observations from sensors resting in $V_{i,j}$. In case of a positive observation, $n_{i,j}$ is increased, otherwise decreased. A positive observation occurs if the map point lies within the cameras field of view and concurrently produces a positive match during data association (e.g. feature matching). A negative observation occurs if the sensor should observe the landmark but did not produce a positive match. From $n_{i,j}$ the importance weight $p_{i,j}$ is calculated according to a sigmoid function

$$p_{i,j} = \frac{1}{1 + e^{-\lambda n_{i,j}}}, \tag{5.5}$$

where λ is a user defined scalar. Looking at the sigmoid function displayed in Figure 5.6(a) for different values of λ, smaller values of λ yield in smoother curves and should be used for low dynamic scenes, whereas higher values are appropriate for a fast changing surrounding. Using this kind of voting scheme one should clamp the bin value $n_{i,j}$ between $0.05 < p_{i,j} < 0.95$ to avoid over- or undersaturated bins (i.e. environmental changes would not affect the objects probability). It should be mentioned that any function (e.g. truncated signed distance function) can be applied to compute a landmarks importance $p_{i,j}$.



(a)



(b)

Figure 5.6: Icosahedron update procedure. (a) Weighting function for different values of λ, which is used for assigning each bin a probability for being visible. (b) Update of the descriptor with three sensor positions. Darker colors indicate a higher weighted bin.

An example of a landmark together with its HOC descriptor observed from various cameras multiple times is shown in Figure 5.6(b), where darker colors indicate a higher weighted bin. The update procedure is summarized in Algorithm 4.

Algorithm 4 Update - Counting HOC Weighting

```
Input: bin $V_{i,j}$
Output: probability $p_{i,j}$
  if positive match then
    $n_{i,j} \leftarrow n_{i,j} + 1$
  else
    $n_{i,j} \leftarrow n_{i,j} - 1$
  end if
  $p_{i,j} = \frac{1}{1+e^{-\lambda n_{i,j}}}$
```

A Probabilistic Scheme for HOC Update

Here, the log-odd representation, as utilized by standard occupancy grid mapping algorithms, is employed to update the probability $p_{i,j}$. The probability of each bin is defined in its log-odd form $l_{i,j}$, where the value at time t is computed from the previous one as follows:

$$l_{i,j}^{t} = l_{i,j}^{t-1} + log\left(\frac{p(d)}{1-p(d)}\right) - l_{i,j}^{0}, \tag{5.6}$$

where $l_{i,j}^{0}$ denotes the initial log-odd value corresponding to the bin volume $V_{i,j}$ and $p(d)$ represents a probability the associated map point is visible from within that volume (in most cases $l_{i,j}^{0} = 0$). The probability $p_{i,j}$ can be recovered from Equation 5.6 as

$$p_{i,j} = 1 - \frac{1}{1 + exp\{l_{i,j}\}}. \tag{5.7}$$

In order to calculate $p(d)$ we make use of the matching score established during data association, where we seek to assign higher weights to better matches. Using Lowe's ratio [74] of the closest d_1 to the second closest d_2 matching distance

$$d = \frac{d_2}{d_1} \tag{5.8}$$

we compute $p(d)$ as

$$p(d) = \begin{cases} 0.7 & d > st_L \\ \frac{1}{s-1}\left(\frac{0.2}{t_l}d + 0.5s - 0.7\right) & t_l \leq d \leq st_L \\ 0.4 & d < t_L, \end{cases} \tag{5.9}$$

where t_L denotes Lowe's matching threshold and s a user specified scaling factor to adjust the sensitivity of the weighting function. Hence, a bin is downweighted if no match occurs (i.e. $d < t_L$) or upweighted proportional to the matching performance of its feature descriptor. Figure 5.7 shows the proposed weighting scheme for three different values of s. The more distinctive the feature descriptor (i.e. the higher Lowe's ratio d), the more reliable its associated landmark. The value of s just modifies the slope of the weighting function, where a lower value should be chosen for high dynamic scenes. Higher values flatten the weighting function suitable for low dynamic scenes. The update procedure is summarized in Algorithm 5.

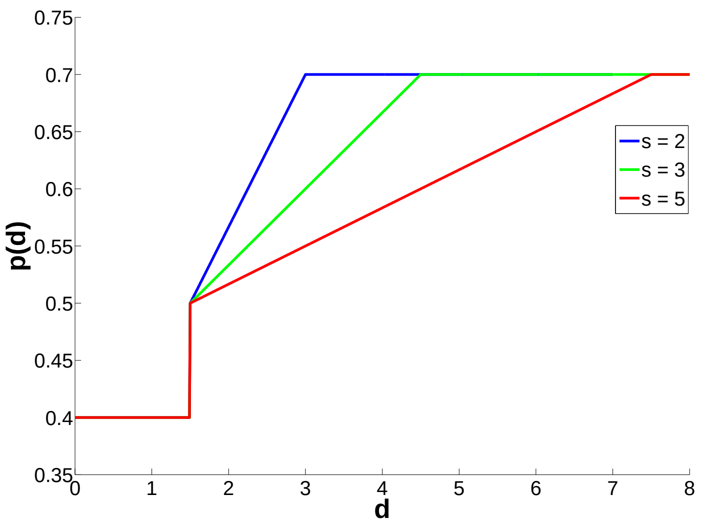


Figure 5.7: Probabilistic HOC weighting. Based on Lowe's ratio d during matching we assign higher weights to those map points associated with a more distinctive feature descriptor. We show three different plots of the weighting function stated in Equation 5.9 for $s = \{2, 3, 5\}$.

Algorithm 5 Update - Probabilistic HOC Weighting

```
Input: bin $V_{i,j}$
Output: probability $p_{i,j}$
  $d = \frac{d_2}{d_1}$
  compute $p(d)$ using Equation 5.9
  $l^t_{i,j} = l^{t-1}_{i,j} + log\left(\frac{p(d)}{1-p(d)}\right) - l^0_{i,j}$
  $p_{i,j} = 1 - \frac{1}{1+exp\{l^t_{i,j}\}}$
```

5.2.5 Incorporation of the HOC descriptor into the Visual SLAM Process

Throughout this chapter we describe how to embed the HOC descriptor into our visual SLAM framework, which processes are influenced and highlight the benefits arising from the descriptor incorporation. Figure 5.8 highlights the different states influenced by the HOC descriptor and its benefits.

To incorporate the proposed descriptor we sightly change our environment model defined in Section 4.6.1. Despite its three-dimensional coordinates and the feature de-

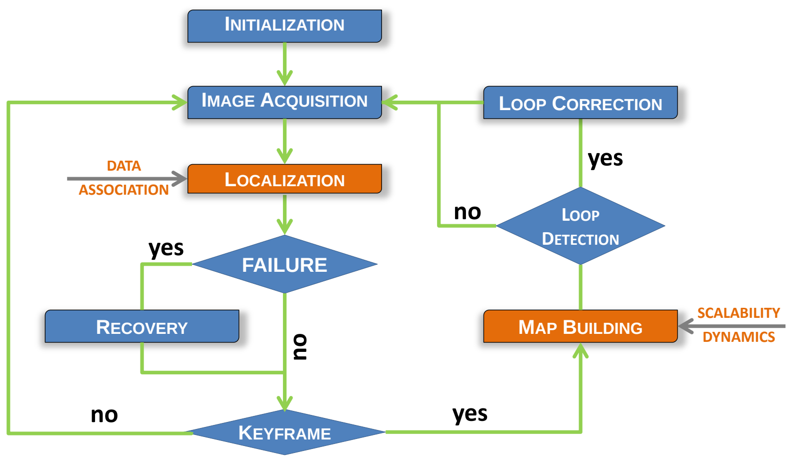


Figure 5.8: Building blocks influenced by the HOC descriptor (orange). The visibility information and importance measure stemming from the HOC descriptor affects the localization and map building process. During pose estimation the descriptor facilitates data association. During the map building stage dynamic scene elements can be filtered out which influences the algorithms scalability.

scriptor each landmark $\mathbf{X_j}$ is attached a HOC descriptor H_j:

$$\mathbf{X_j} = \begin{cases} (X_j, Y_j, Z_j, 1)^T \\ d_j \\ H_j \end{cases} \tag{5.10}$$

New descriptors are created everytime a new landmark is added to the map during map building while descriptor updates are performed during the localization phase.

The proposed HOC descriptor enables us to implicitly store the visibility of each three-dimensional landmark together with a view-dependent importance measure for localization. This more complex map representation comes along with a lot of benefits regarding dynamic environments, scalability and also facilitates data association. These advantages are demonstrated in the following on a simple toy example: A stereo camera is mounted on a tripod. We placed several objects in front of the camera onto a desk where nearly all of them are moved, removed and added over time. Some input data is shown in Figure 5.9.

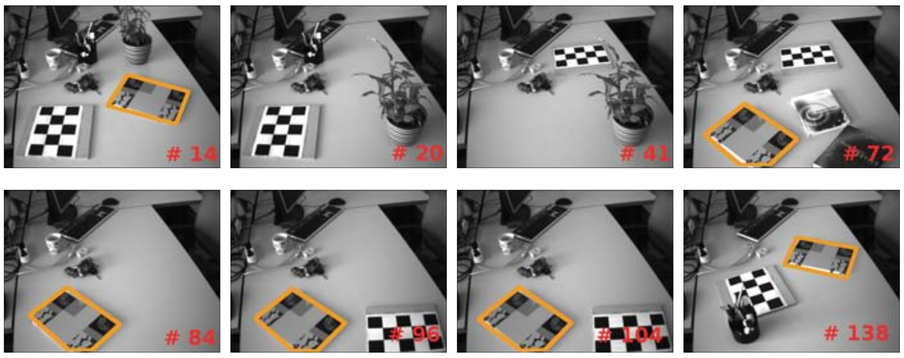


Figure 5.9: Toy example to highlight the benefits of the HOC descriptor. We show the left images of a stereo camera which has been mounted on a tripod. Several catchy objects in front of the camera have been moved, added or disappeared over time. Frame numbers are shown at the bottom right.

Dynamic Scene Updates

Short- and long term scene dynamics as well as occlusions are handled implicitly by the HOC descriptor. Instead of simply adding all incoming sensor data to the map we are able to include negative information in landmark based maps. This allows us to react on environmental changes by filtering out unstable or unseen map points by looking at the bins weight. Therefore, the resulting map only contains the most recent and most stable features.

The evolution of the map over time is shown in Figure 5.10. The top row depicts the ever growing environment without using any visibility information. Contrarily, adding the HOC descriptor to every landmark results in a consistent map holding only the most recent objects (bottom row). Here, we showed the most likely map by visualizing only those map points whose maximum bin weight exceeds 0.5. The landmarks corresponding to the book highlighted orange in Figure 5.9 are also bordered in Figure 5.10. After processing all frames there exist three entities of the same object at different places (top row). When using the HOC descriptor unseen landmarks have been removed successfully and only one book occurs at the right place in the map.

To further highlight the descriptors ability to handle dynamic objects, we also project the actual map onto the currently processed frame demonstrating that the map only contains the most recent environment configuration when using the HOC descriptor. Without visibility information the map gets populated with all moved objects resulting in duplicate occurrences of the same object and an ever growing number of landmarks. Projected map points (red) and the number of landmarks are shown in Figure 5.11.

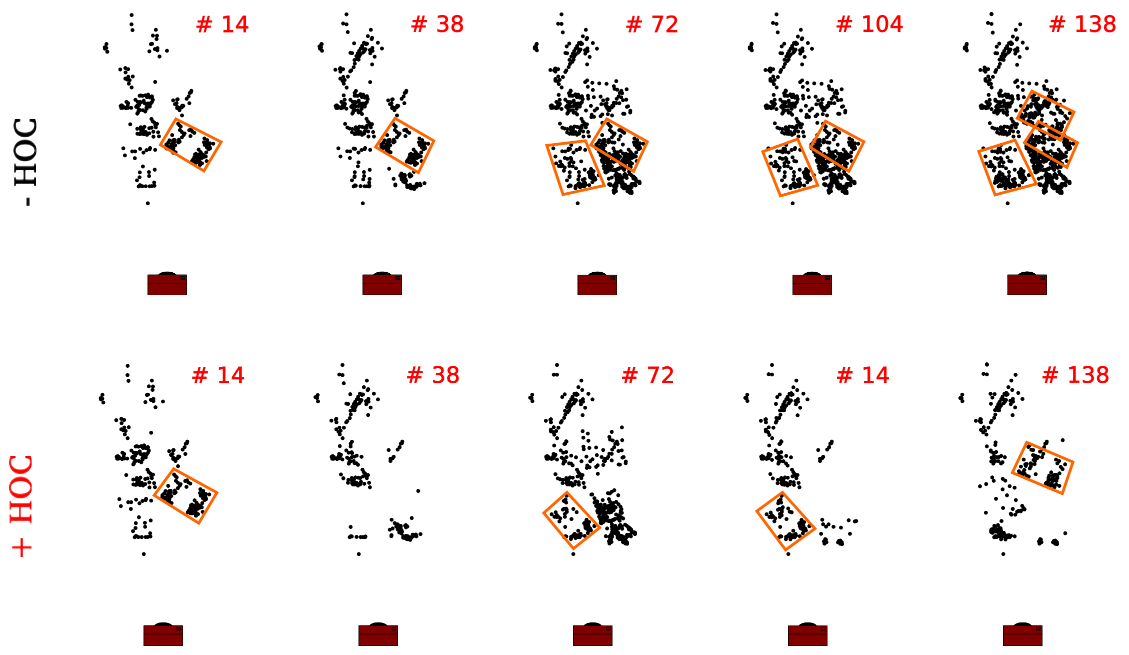


Figure 5.10: Dynamic environment - evolution of the map over time. An estimated map using stereo images (Figure 5.9) is presented. We oppose the map without using the proposed descriptor (top row) to the one estimated when using the HOC descriptor (bottom row). For visual inspection we highlighted the landmarks corresponding to the book marked orange in Figure 5.9. Compared with the input images this results in a three-dimensional pointcloud representing only the most recent environment configuration. Frame number is depicted on the upper right corner.

Scalability

The HOC descriptor allows us to keep the map size proportional to explored space rather than time. Landmarks whose maximum of all bins is smaller than a predefined threshold w_{min} are interpreted as vanished features, since they are not visible from any viewpoint. The effect of this simple threshold operation is again demonstrated on the toy example. This time we monitored the map size over time, which is shown in Figure 5.12. Instead of a steadily rising number of landmarks (black) the HOC descriptor guarantees a nearly constant map size.

Data association

The importance measure encoded in each histogram bin greatly enhances data association through effective prefiltering of visible map points. Instead of performing data

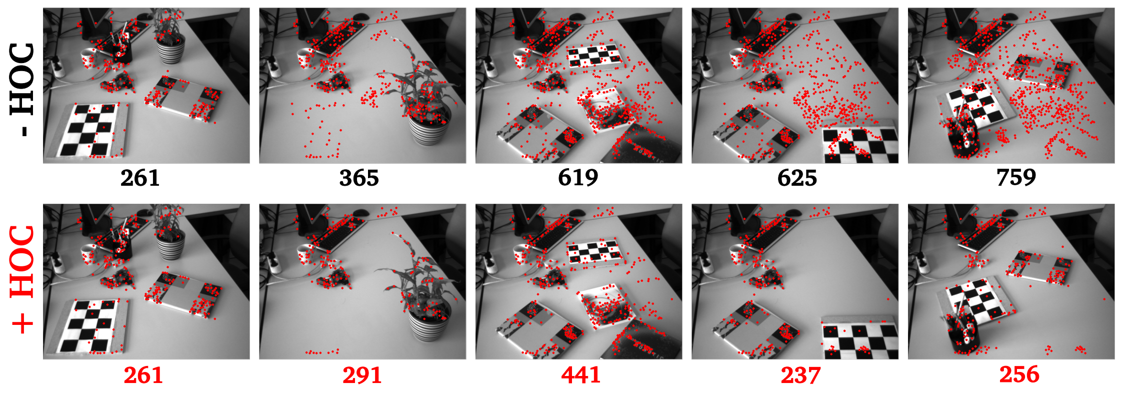


Figure 5.11: Dynamic environment - evolution of the map over time. All points inside the cameras view frustum are projected onto the image plane (red dots). We also stated the number of points visible in each image.

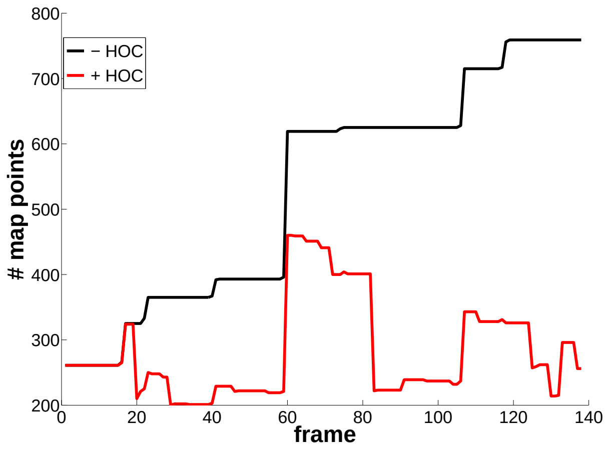
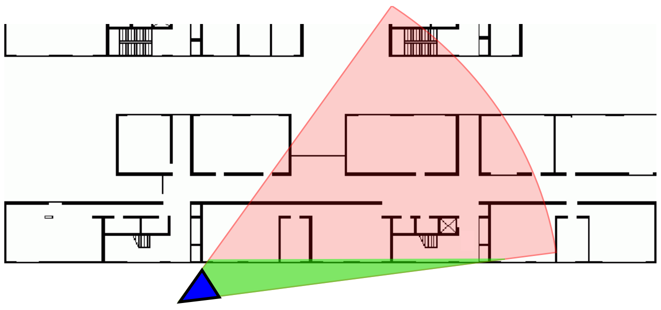


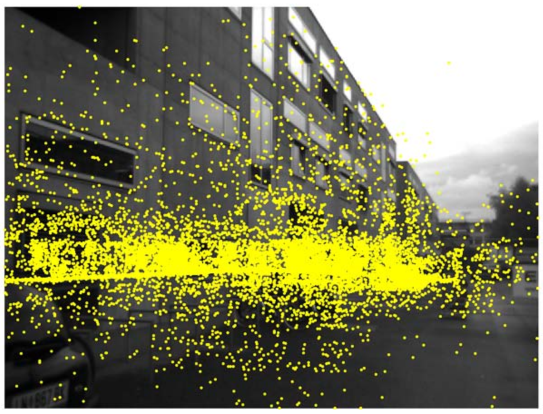
Figure 5.12: Scalability demonstration - evolution of the map over time. We recorded the number of landmarks in the map every frame. Thresholding the maximum bin value and erasing the invisible map points lead to a more constant map size (red) in contrast to the standard SLAM approach (black).

association (e.g. descriptor matching) using all points within the cameras view frustum we select those map points with an importance weight exceeding a predefined threshold. As a consequence we reduce the number of matching candidates which shortens computation time and delivers more robust matches, especially in large, high dynamic scenes.

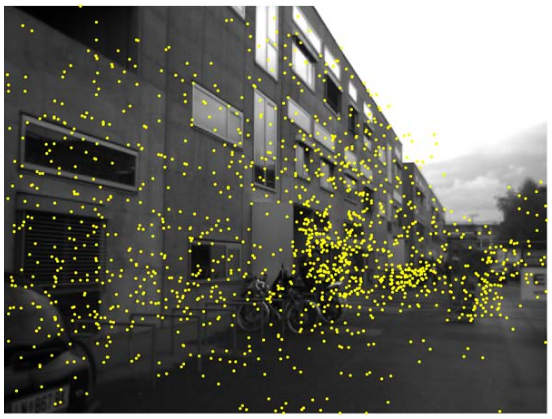
To sketch this behavior we picked out a camera pose from an already mapped test run as described in the experimental Section 6.3. As shown in Figure 5.13(a) the camera is located outside a building facing a huge portion of an already reconstructed areal.



(a)



(b)



(c)

Figure 5.13: Improving data association. (a) View of a camera located outside in front of a building already mapped. The potentially visible map with (green) and without (red) incorporating visibility information. (b) Assuming a three-dimensional pointcloud, simply reprojecting all points (yellow) within the view frustum of the camera would result in a huge matching effort every frame when performing tracking. (c) Selecting the most visible points drastically reduces the possible matching candidates.

Without using visibility information during data association all map points resting inside the cameras view frustum (red) are considered for descriptor matching (Figure 5.13(b)). With the HOC descriptor we can filter out those visible from that camera pose (green) by considering the descriptors weight, which reduces the possible matching candidates drastically (Figure 5.13(c)).

5.3 A Novel Method for Keyframe Organization

To tackle the challenges especially occurring in long-term operation - computation time, storage requirements or multiple nested loop closing - we propose a pose graph together with an adaptive keyframe selection routine and a novel loop closing approach. First, this is motivated in Section 5.3.1 followed by a theoretical description in Section 5.3.2. Finally, our solution to fast, nested loop closing is presented in Section 5.3.3.

5.3.1 Motivation

State-of-the-art simultaneous localization and mapping algorithms relying on sparse bundle adjustment using keyframes follow very naive approaches for keyframe selection, map point insertion, sliding window bundle adjustment and loop closing. Usually, they are designed to operate in a small, restricted area while visiting each place only once for map construction. Loop closing is performed once in order to correct wrong pose estimates caused by drift. If a robot should perform life-long visual localization and map building in a large scale environment, more sophisticated rules are necessary to reduce memory consumption and computation time. Hereby, the most challenging tasks are *when* and *where* to add new sensor information to the environment while reducing the number of points and keyframes in the map to a minimum. Adding a new keyframe everytime when the robot has traveled a certain distance would steadily increase computation time and memory consumption. Instead of that we should be able to decide if we are moving in already explored areas while tracking quality is considered well or if we are entering unknown terrain. In the first case tracking without mapping is sufficient and no additional keyframe is necessary. In the latter, keyframes have to be established and map points should be added to carefully selected areas - preferred to those regions lacking landmark information or where the environment has changed due to dynamic entities in the map. The state machine depicted in Figure 5.14 shows the desired interaction between tracking and exploring in a long-term SLAM system.

When tracking only is performed over a long period of time, keyframe insertion does not occur at periodic points in time. Therefore, sliding window bundle adjustment cannot be performed over the last N keyframes only, since no frame overlap would be guaranteed. Alternatively, keyframes around a certain neighborhood should be considered when performing sliding window bundle adjustment, which may also include keyframes stemming from previous test runs created long time ago.

Moreover, nested loop closure correction could be accelerated if only affected map

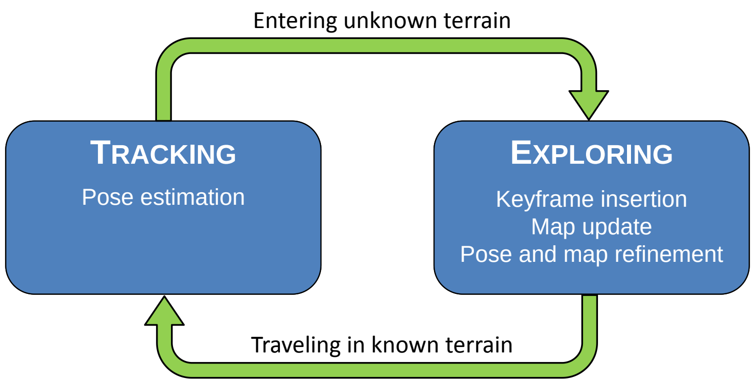


Figure 5.14: State machine describing the ideal transitions between tracking and exploring mode of a mobile robot. To reduce computation time and memory consumption keyframe insertion and map expansion should take place at selective points in time. Ideally, pose estimation should be performed when the robot is traveling in already explored areas. When entering unknown terrain or environment has changed, the robot should start exploring again and add keyframes and map points at carefully chosen places.

parts are considered without touching already consistent map portions. A typical scenario is shown in Figure 5.15, where a robot has already explored and corrected a large map portion (red trajectory) and only a small inconsistent part (blue trajectory) needs to be added to the large self-contained loop. Since no additional information could be gained when performing loop closure correction over the whole trajectory, it is sufficient to attach the small inconsistent part to the existing map.

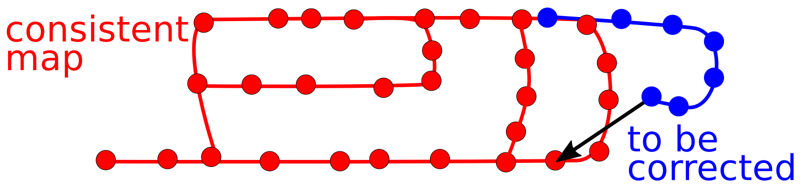


Figure 5.15: Loop closure scenario within a long-term mapping experiment. Hereby, only a small inconsistent part (blue) has to be added to an already corrected map and trajectory (red). To gain efficiency it would be wise to identify the affected parts for loop closing, while the rest remains untouched.

In the following solutions to the above mentioned problems are presented, which can be easily integrated into our visual SLAM framework proposed in Section 4.6. New heuristics for advanced keyframe selection, sliding window bundle adjustment and guided landmark insertion utilizing a view graph are presented in Section 5.3.2. To gain efficiency and scalability in loop closing, we identify affected map regions through an efficient graph search, and apply a large scale nested loop optimization as described in Section 5.3.3. Figure 5.16 highlights the different states influenced by the presented solutions and their benefits.

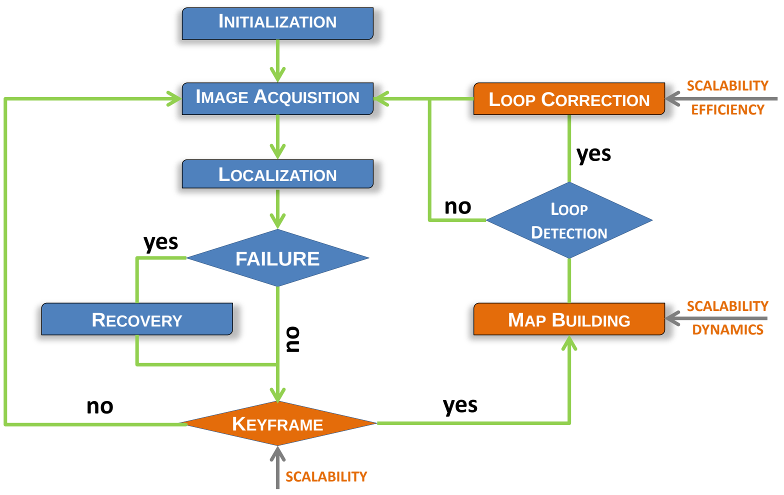


Figure 5.16: Keyframe selection, map building and loop closing processes are affected by the proposed pose graph. Adaptive keyframe selection reduces the number of poses in the graph. Heuristics for map expansion and loop closing reduces the number of points in the map as well as computation time during sliding window bundle adjustment and loop correction.

5.3.2 Pose Graph for Keyframe Organization and Map Extension

Similarly to topological maps we propose to organize camera poses in an undirected, unweighted graph, where each keyframe represents a node. Links are established incrementally after map expansion, which is described in the following section. Figure 5.17 depicts a pose graph of keyframes after two independent runs.

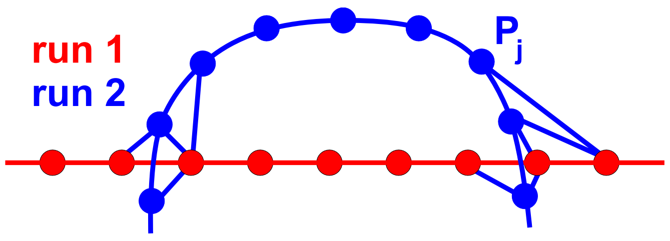


Figure 5.17: Graph structure. Keyframes $\mathbf{P}_j$ are organized in an unweighted, undirected graph. Links between nodes are established incrementally during pose tracking.

An Image based Measure for Keyframe Selection

A crucial step in a visual SLAM system is the decision when to add a keyframe and expand the map. Typically, an incoming frame is assigned to be a new keyframe if the camera has moved a certain distance from the previous keyframe [66, 127]. This helps to maintain visual connectivity for the subsequent bundle adjustment procedure, but also steadily increases the number of poses to be optimized in continuous mapping over a long period of time. Hence, a more sophisticated rule for keyframe selection is needed during pose tracking. Pose tracking for an incoming frame $\mathbf{P_c}$ is performed by minimizing the reprojection error over $2D \leftrightarrow 3D$ correspondences between image features and potentially visible map points. Keypoints x_c extracted from the current frame are matched against reprojected map points $\mathbf{X_m}$ using a guided search, which (when implemented on the GPU) allows for real-time performance [19]. To keep the map size proportional to explored space rather than exploration time, we propose the following rule: In every new frame we form a two-dimensional histogram of keypoints in the image domain with a bin size of 50×50 px. Let N_c be the set of bins which contain at least one keypoint x_c, whereas N_m denotes the set of bins containing a keypoint successfully matched during visual tracking. We create a new keyframe from $\mathbf{P_c}$, if

$$\frac{|N_m|}{|N_c|} < 0.55, \tag{5.11}$$

which assures that at least 55% of the image area is covered by keypoints used for tracking. Additionally, tracking quality measured by the mean reprojection error after pose correction defines a further criterion for keyframe selection. We create a new keyframe, if

$$d(m_{ij}, \mathbf{P_i}\,\mathbf{X_j}) < \tau, \tag{5.12}$$

where τ is a user defined threshold (throughout our experiments we set $\tau = 2.5px$). If Equations 5.11 or 5.12 are fulfilled, $\mathbf{P_c}$ becomes a new keyframe $\mathbf{P_k}$ and is added to the pose graph as a new node. We link $\mathbf{P_k}$ to these four keyframes which produce the most correspondences during tracking (sketched as dashed lines in Figure 5.18). The described procedure is again visualized in Figure 5.19.

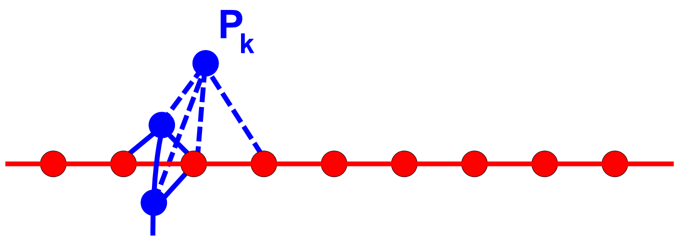


Figure 5.18: Keyframe insertion. When performing map expansion a new keyframe $\mathbf{P_k}$ is added as a node to the graph and linked (dashed lines) to those keyframes producing the four most matching correspondences during tracking.

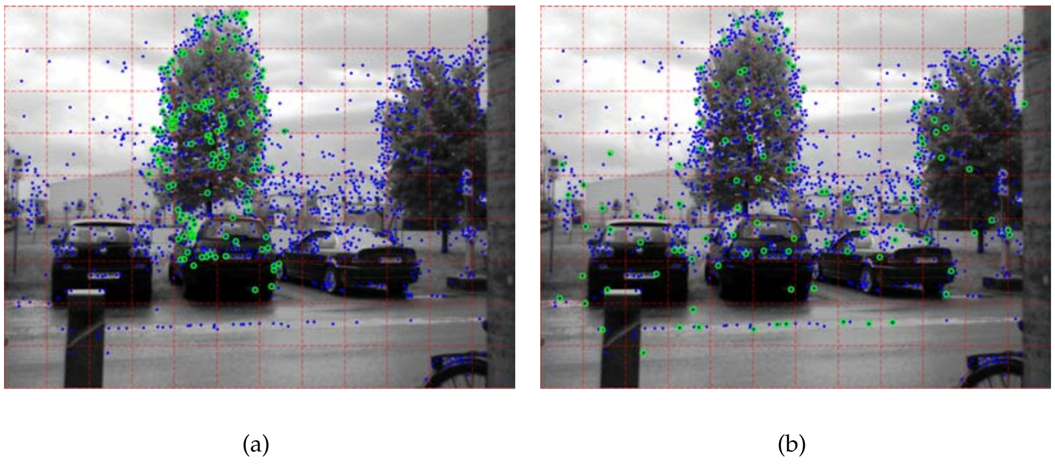


Figure 5.19: Keyframe selection. The red-dotted lines depict the borders of a two-dimensional histogram. Extracted keypoints are shown as blue dots defining the set of bins N_c containing at least one keypoint. The green keypoints have been successfully matched during visual tracking. (a) 138 (green) out of 1665 (blue) keypoints produce a positive match during data association but cover only 25% of the whole image. Thus a new keyframe is established. (b) 101 (green) out of 1665 (blue) keypoints are matched and cover 62% of the whole image area. Although there are less matches than in (a) no keyframe is created because of the better distribution of matched map points.

Map Expansion based on Scene Coverage

After a new keyframe has been inserted new map points have to be created. A naive approach would be to add every unmatched feature point to the existing map. In contrast to that we would like to add new landmarks only in unexplored surroundings or where the environment has changed. Therefore, we again use the histogram from the tracking process and consider a subset $\overline{x_c}$ of all keypoints x_c

$$\overline{x_c} = \{x_c | x_c \in \{N_c \setminus N_m\}\}, \tag{5.13}$$

which form a candidate set of features for map expansion. The bins containing the candidate set $\overline{x_c}$ are highlighted yellow in Figure 5.20. Each feature is matched with a carefully chosen subset of keyframes already stored in the map to guarantee visual connectivity in the subsequent bundle adjustment process and to establish point correspondences for triangulation in case of monocular map building. To reduce the matching effort candidate keyframes are determined through a breadth first search within the graph up to a depth of four. Figure 5.21(a) highlights the frames considered for map extension compared to a fix sliding window.

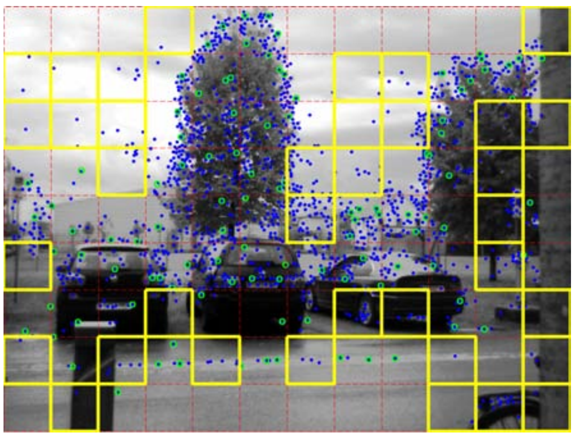


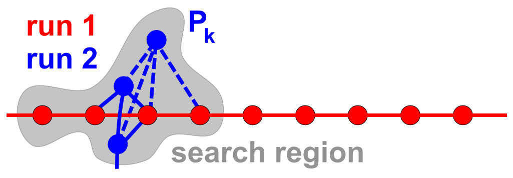
Figure 5.20: Candidate features. The histogram bins containing no matched feature from the pose tracking process are highlighted yellow. The keypoints (blue dots) resting in these regions are considered for map extension. Hence, we assure only constructing landmarks in unknown terrain.

When creating new map points we have to distinguish if we are equipped with a range image device or a monocular camera.

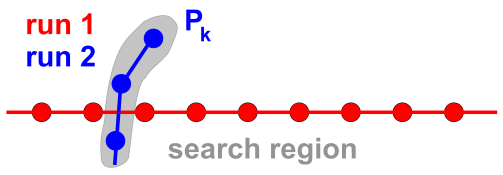
In the case of a range image device map extension is an easy task and we only have to decide which of the candidate features $\overline{x_c}$ are added to the map. From each bin the feature producing the most matches with the candidate frames is added as a landmark to the map. Its three-dimensional position is already given through stereo triangulation as described in Section 3.2.1 followed by a transformation into the global world coordinate frame using the pose of the keyframe $\mathbf{P_k}$.

For monocular devices we eliminate those candidate keyframes with insufficient baseline for accurate triangulation and the reduced subset is used for subsequent feature matching. As above, the feature producing the most matches of each bin is triangulated using multi-view triangulation given the keyframe poses from the view-graph.

After feature matching and triangulation we perform sparse bundle adjustment with the candidate frames, the current keyframe $\mathbf{P}_k$ and their associated map points to refine both structure and motion in a local subregion.



(a)



(b)

Figure 5.21: Candidate keyframes. (a) Keyframes used for feature matching, landmark generation and subsequent sliding window bundle adjustment are established through a breadth first search in the view graph. The affected region is highlighted gray. (b) Other approaches only consider the last N keyframes where each test run is considered individually although visiting the same area.

The presented rules allow to create a new keyframe if we explore unknown surroundings, environment has changed or tracking quality has reduced. Additionally, it guarantees that only new observations are considered for map expansion. In contrast to existing approaches [63] we do not use a fixed number of frames for sliding window bundle adjustment and frames considered for bundling are taken from various test runs as shown in Figure 5.21(a), which is a desirable side effect in long-term mapping or when visiting the same place multiple times.

5.3.3 An Accelerated Loop Closing Routine

Throughout this chapter we propose a new procedure allowing for fast loop closure correction. Again candidate frames for loop closure and relocalization are detected by a vocabulary tree approach relying on the visual bag of words scheme as described in Section 4.5.

Once a loop has been detected, a possible solution would be to perform structure and motion optimization over the whole trajectory, which is computationally very demanding. To correct translation, rotation and scale (in case of monocular sensor input) we make use of the pose optimization procedure proposed in [127].

Hereby, the pose graph optimization procedure estimates new keyframe poses by minimizing the residual of relative pose constraints $\Delta_{i,j} = P_j \cdot P_i^{-1}$, where each constraint is described by a similarity transformation

$$\Delta_{i,j} = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}, \tag{5.14}$$

where s handles the scale drift occurred over time when performing monocular SLAM. In case of a range imaging device

$$\Delta_{i,j} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}, \tag{5.15}$$

for all relative pose constraints since no drift needs to be corrected.

If a single camera is used then $s = 1$ for all constraints despite the loop constraint $\Delta_{c,l}$, where a similarity transformation is estimated by a RANSAC routine between two corresponding 3D point sets as follows:

After loop closure detection we establish $2D \leftrightarrow 3D$ correspondences $x_c \leftrightarrow X_l$ between the current frame $\mathbf{P_c}$ and the map points visible from the potential loop frame $\mathbf{P_l}$ (i.e. the best matching keyframe). We further employ $3D \leftrightarrow 3D$ correspondences $X_c \leftrightarrow X_l$ between map points associated with $\mathbf{P_c}$ and $\mathbf{P_l}$, respectively. A RANSAC [36] routine is applied to robustly compute a similarity transformation T_{lc} between both point sets. A sketch of the loop closing situation is shown in Figure 5.22.

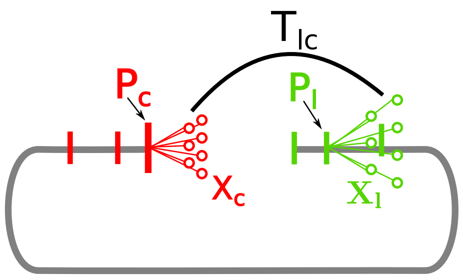


Figure 5.22: Calculation of the similarity transformation during loop closing between $3D \leftrightarrow 3D$ correspondences.

Because we potentially carry a very large and consistent map with a comparably small, inconsistent loop attached to it, we need to identify the subset of keyframes $\overline{\mathbf{P_i}}$ to be optimized (compare Figure 5.15). Starting from the current pose $\mathbf{P_c}$ we traverse the graph in all directions, until we hit an already optimized keyframe $\mathbf{P_o}$. All visited nodes, denoted as $\mathbf{P_v}$, are used for loop correction. The final subset $\overline{\mathbf{P_i}} = \{\mathbf{P_l}; \mathbf{P_o}; \mathbf{P_c}; \mathbf{P_v}\}$ serves as input for the pose graph optimization, where $\mathbf{P_o}$ and $\mathbf{P_l}$ are held fixed. The relative constraints $\Delta_{i,j}$ between poses are also derived from the graph. To geometrically close the loop, we employ the pose graph optimization procedure which minimizes the residual of relative pose constraints, where the loop constraint is modeled the following way:

$$\Delta_{c,l} = \mathbf{P_l} \cdot (\mathbf{P_c} \cdot T_{lc})^{-1} \tag{5.16}$$

An iterative Levenberg-Marquardt algorithm minimizes the following error function

$$\epsilon = \sum_{i,j} r_{i,j}^T \, r_{i,j} \tag{5.17}$$

$$r_{i,j} = log \left[\Delta_{i,j} \cdot \left(\mathbf{P_j} \cdot \mathbf{P_i^{-1}} \right)^{-1} \right], \tag{5.18}$$

where $\mathbf{P_l}$ and $\mathbf{P_o}$ are kept fixed, because they define the entry points of an already closed loop (compare Figure 5.23). This results in a set of corrected similarity poses $\mathbf{P_i}$

$$\mathbf{P_i} = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}. \tag{5.19}$$

Given the subset of optimized poses $\mathbf{P_i}$, we are able to correct a map point $\overline{X_m}$ by computing its centroid

$$X_m = centroid \left(\mathbf{P_i^{-1}} \left(\overline{\mathbf{P_i}} \, \overline{X_m} \right) \right). \tag{5.20}$$

In a final step we perform structure bundling of those map points only visible from $\overline{\mathbf{P_i}}$, to refine the resulting environment.

If we followed a path in the graph and did not visit a previously optimized node, we have detected an open map trajectory (e.g. from relocalization). In this case there is no information available about map consistency and we reorient the keyframes $\overline{\mathbf{P_o}} = \{\mathbf{P_c}; \mathbf{P_v}\}$ and their associated map points with the similarity transformation T_{lc}. Figure 5.23 shows a schematic view of the optimization procedure used for loop correction. If a closed loop exists (left drawing) we end up in an already optimized pose P_o, otherwise we started at an unknown location and re-entered the map (right drawing).

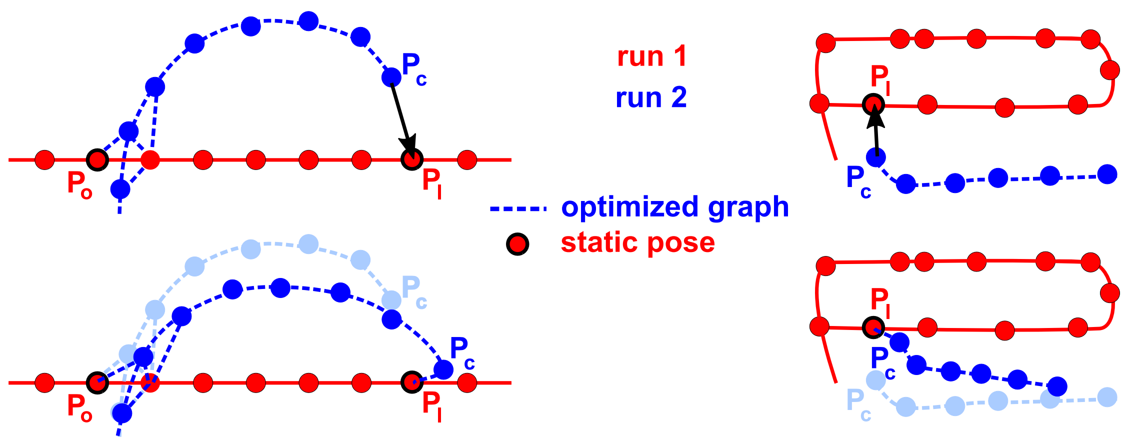


Figure 5.23: Advanced loop closing. Through graph traversal we establish a subset of poses considered for loop closing. Starting from the current camera frame $\mathbf{P_c}$, we traverse the graph until an already optimized keyframe $\mathbf{P_o}$ is reached. All visited keyframes (connected through dashed lines) form the set of poses used for optimization. Two possible scenarios may occur when performing loop closure correction: First (left drawing), the whole loop is corrected. Second (right drawing), we reached an open end and poses are only transformed to fit correctly into the map.

5.4 Conclusion

We presented a novel feature descriptor called histogram of oriented cameras (HOC) holding visibility information and an importance weight for localization for each landmark. In contrast to most existing approaches, which encode visibility in a camera-centered way (e.g. using keyframes), we propose to add a per-feature spatial histogram of the number of observations. Two different descriptor setups (icosahedron or octahedron) has been presented, whereas the octahedron allows for bin computation in constant time given a camera pose. To compute the importance weight of each histogram bin two weighting schemes has been presented. The first one simply counts the number of observations for each bin. The second one follows a probabilistic update procedure taking the descriptors matching score into account. We highlighted the descriptors benefits when incorporated into a visual SLAM framework. Although the HOC descriptor requires more storage it comes along with advantages like the implicit handling of dynamic scene elements, the ability to reduce the number of points in the map and the reduced matching effort during visual tracking.

Additionally, an adaptive keyframe selection routine which compares the current sensor reading to the already existing map has been presented. It allows us to converge to tracking only when traveling in known terrain. On the other hand only sensor readings describing new or unseen portions of the environment are added to the map. These new rules in conjunction with the HOC descriptor guarantee a stabilized map size, a minimal number of keyframes in the map and an up-to-date map being able to adapt to dynamic environments.

The proposed graph structure together with robust loop closure detection allows us to efficiently close large, nested loops. More specifically, when traveling in a large, consistent map, only small, inconsistent parts have to be corrected. Affected camera views can be detected in linear time and subsequent structure refinement only involves a small portion of the whole map. The presented routine assumes a well-defined map the loop is attached to. In future work one could investigate the quality of both maps, where qualitative better map should remain fix whereas the other one is warped to it.

6 Experiments

To deal with highly dynamic scenes within life-long operation a novel feature descriptor and a modified topological map representation have been proposed in Chapter 5. Furthermore, to tackle the problems of storage requirements, computation time and multiple nested loop closures, an accelerated loop closing strategy together with an adaptive keyframe selection and map point insertion routine has been presented. The proposed methods have been successfully embedded in our visual SLAM framework described in Section 4.6.

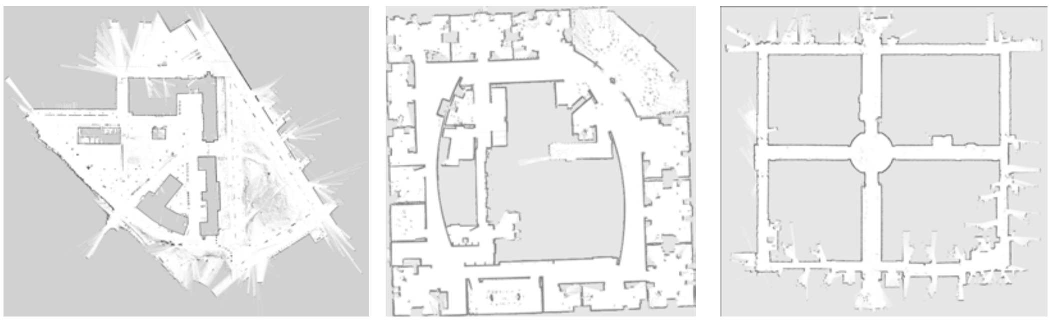
Since our main focus lies on long-term mapping within a dynamic environment a new benchmark dataset containing images of different times of day has been recorded, which is described in Section 6.1. To highlight the benefits of the novel feature descriptor proposed in Section 5.2 several synthetic and real-world experiments have been performed and evaluated in Section 6.2. Here, we focused on the evaluation of scalability, localization performance and map adaption. The fast and adaptive loop closing strategy utilizing the pose graph described in Section 5.3 is evaluated in Section 6.3.

6.1 The Problem of Evaluation in Dynamic SLAM

Although there exist a lot of publicly available datasets, none of them addresses the problem of life-long localization and map building in an ever changing world. After reviewing a few of them in Section 6.1.1, we motivate and present our own dataset named *CDBench* in Section 6.1.2.

6.1.1 Introduction and Motivation

In order to evaluate a SLAM solution one can choose between a vast amount of datasets available online. Most of these datasets only provide raw sensor information without any groundtruth data, where the majority of them consists of odometry and laser scan data. The *KTH dataset*[1] provides laser and odometry readings of a planar moving robot from an area outside the university building. In order to evaluate the mapping performance a list of walls of the building is provided. For experiments with this dataset we refer the reader to Folkesson and Christensen [38]. Similarly, the *Killian Court* dataset provided by Bosse et al. [10] contains laser, sonar and odometry readings of a robot moving in the plane over a trajectory length of 2.2*km*. The greatest repository consisting of laser, sonar and wheel encoder data is called *Radish dataset*[2] [57]. Here, the most prominent ones are the Freiburg Campus (by Cyrill Stachniss), the Intel Reasearch Lab (by Dirk Hähnel) or the ACES Campus (by Patrick Beeson). Some estimated maps are shown in Figure 6.1.



(a) (b) (c)

Figure 6.1: Estimated maps of the Radish dataset repository all consisting of laser and odometry data. (a) Freiburg Campus dataset (courtesy of [125]). (b) Intel Research Lab dataset (courtesy of [47]). (c) ACES Texas dataset (courtesy of [44]).

A dataset consisting of vision, odometry and laser scan data has been recorded by Smith et al. [118]. They maneuvered a robot several kilometers through their campus and the adjacent park altogether resulting in three different testsets containing multiple loops and moving people. For accuracy evaluation of the vehicle position GPS data is also provided.

More recently, Sturm et al. [129] published a large dataset comprising of color images

[1]`http://www.nada.kth.se/~johnf/kthdata/dataset.html`

[2]`http://radish.sourceforge.net/index.php`

and related depth data from Microsoft KINECT and a groundtruth trajectory of all camera poses[3]. They recorded nine different sequences of a desk or office environment where groundtruth is estimated by a highly accurate motion-capture system.

All mentioned datasets do not focus on long-term or radiometric dynamics and data logging has been done in a single run only. A dataset with laser and odometry data of three independent loops containing exclusively short-term dynamics (e.g. moving people) has been recorded by Wang et al. [142], which aims at moving object tracking only. The most complete repository is presented by the *Rawseeds Project*[4] providing static, dynamic, indoor or outdoor environments. Here, the robot is equipped with a huge amount of sensors like sonar, laser, odometry, IMUs, GPS and cameras. Additionally, they provide groundtruth data of the robot pose in a small, predefined area within their test circuit (see Figure 6.2). Indoors they mounted multiple cameras at the ceiling detecting visual tags mounted on the robot in order to estimate the robot's position within the restricted area. Outdoors the robots pose is captured with a high precision RTK (Real Time Kinematics) GPS system. Unfortunately, only 3DOF motion estimation is necessary since all sensors are mounted rigidly on the robot moving exclusively in planar environments. Two out of nine datasets contain short-term scene dynamics and trajectories overlap at segments only. However, their main purpose lies in the comparison of SLAM algorithms with different sensor technologies. Therefore, several error measurements are suggested, which are described more precisely by Bonarini et al. [9] and Ceriani et al. [14]. Some images of the indoor datasets together with a floorplan and the traveled trajectory are presented in Figure 6.2. At the time of writing nine different SLAM algorithms mainly operating on laser data have been tested on single runs only.

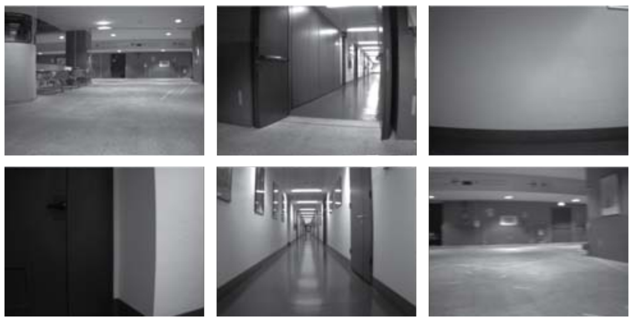
All benchmarks mentioned so far do not provide any evaluation of the environment estimated by a SLAM algorithm. Furthermore, most of them contain data captured in a single run only and therefore only short-term scene dynamics may occur (see Table 6.1). To evaluate long-term or radiometric dynamics a longer recording time would be necessary. Moreover, there exist just three datasets containing one or more vision sensors, where only the *Rawseeds Project* captured data at different days. A dataset for evaluating long-term visual localization and mapping presumes images captured at multiple days while traversing the same area multiple times. It should contain both long- and short-term dynamics as well as different lighting and weather conditions. At the time of writing such a dataset does not exist.

[3]`http://cvpr.in.tum.de/data/datasets/rgbd-dataset`

[4]`http://www.rawseeds.org/home/`

Dataset	sensors	indoor(i) outdoor(o)	motion	dynamic elements	GT poses	GT map	recording time
KTH	laser odo	o	3DOF	×	×	×	1d
Killian Court	laser sonar odo	i	3DOF	×	×	×	1d
Radish	laser sonar odo	i, o	3DOF	×	×	×	1d
New College	laser odo mono stereo	i, o	3DOF	short-term	GPS	×	1d
RGB-D	Kinect	i	6DOF	×	mocap	×	1d
Wang	laser odo	i	3DOF	short-term	×	×	1d
Rawseeds	multi sensor	i	3DOF	short-term radiometric	mocap	×	3d
		o	3DOF		GPS	×	5d
CDBench	stereo	i	6DOF	short-term	tripod	visual	1d
		i,o	6DOF	short-term long-term radiometric noisy	measured land-marks	×	10d

Table 6.1: Comparison of different SLAM benchmarks. There exist a lot of publicly available datasets, where the most of them contain laser and odometry data. Groundtruth map (GT map) and poses (GT poses) are rarely available and data is acquired in a single run only. The *CDBench* instead has been captured at 10 different days while traversing a fixed number of landmarks every run. Notation: (i) indoor, (o) outdoor, (mocap) motion capture system, (odo) odometry provided by wheel encoder.


(a)

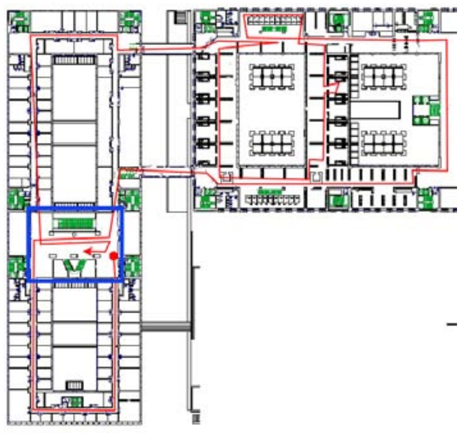

(b)

Figure 6.2: The Rawseeds dataset. (a) Images from the indoor dataset containing challenging pictures like shiny corridors, blurred hallways or featureless areas. (b) Associated floorplan with the trajectory sketched (red) and the groundtruth area (blue).

6.1.2 A New Dataset for Benchmarking Dynamic SLAM Approaches

Traditionally, a SLAM dataset consists of a moving sensor which explores a mostly static environment. The main difficulties when constructing a benchmark dataset especially for large-scale dynamic environments are manifold. First, groundtruth for the traveled trajectory has to be provided. While outdoors this would be possible with GPS, indoor sequences require more sophisticated setups. Especially when covering large areas, groundtruth generation would need an inside-out tracking system monitoring the sensor pose over the whole scenery. Second, groundtruth generation for an ever changing environment poses further challenges. One would require an accurate, three-dimensional drawing of the whole scenery both indoors and outdoors everytime the environment has changed. Finally, every SLAM algorithm produces a different environment model (e.g. planes, lines, pointclouds, occupancy grids) where a generally accepted mapping error measure is necessary. Consequently, evaluating the localization and mapping performance in static and dynamic environments becomes impractical and the problem of benchmarking dynamic scenes needs to be divided into simpler evaluation tasks.

To evaluate the robustness of a SLAM algorithm against dynamic changes one can keep a static camera in a restricted environment whilst objects are moved randomly within the camera's view frustum. This results in groundtruth poses every frame and mapping performance can be evaluated at least visually. If both camera and scene are moving, benchmarking localization and mapping performance becomes impossible. To capture as much scene dynamic as possible a long-term experiment throughout several days traversing the same area over and over again is necessary. This allows for the evaluation of localization repeatability and accuracy within a dynamic world as well as the algorithm's robustness against an ever changing world.

Due to the lack of publicly available datasets in both high and low dynamic environments captured over a long period of time, we recorded our own data, which is available online[5]. To account for different dynamic scene elements and to permit some sort of map evaluation we decided to provide two separate datasets: First, an indoor dataset with a non-moving stereo camera, where a lot of different objects in front of the camera have been moved, deleted or occluded. This dataset aims at evaluating localization accuracy, robustness and map building accuracy. Second, a long-term experiment over 10 days in a mixed indoor and outdoor environment is used to evaluate the performance, robustness, accuracy and scalability of a SLAM algorithm in a life-long setting. In contrast to existing datasets we focus on long-term dynamic scene elements, which pose a greater challenge for every visual SLAM application. Therefore, several objects have been moved deliberately.

All datasets have been recorded with the CCD stereo-rig *Bumblebee2* from *PointGrey*[6] featuring a resolution of 640×480 pixels and a baseline of approximately $12cm$. In order to receive internal camera parameters and the relative orientation of both cameras to each other we used the *Camera Calibration toolbox for Matlab* [11]. For electric power supply we used a standard 12V, 7.2Ah lead-acid battery. Frame grabbing has been implemented using libdc1394[7] on a standard consumer laptop running Ubuntu 10.04. The whole hardware setup is shown in Figure 6.3. In the following the content of both datasets is described in detail.

Static Camera - Moving Scene

The first dataset consists of three sequences (named *indoor_a*, ..., *indoor_c*) with the stereo-rig mounted statically on a tripod to get groundtruth data for camera pose and relocalization performance in high dynamic scenes. Objects in front of the camera have been moved, deleted or occluded several times. In an additional dataset (named *in-*

[5]`http://rvlab.icg.tugraz.at/project_page/project_slam/project_slam.htm`
[6]`http://www.ptgrey.com/products/bumblebee2/bumblebee2_stereo_camera.asp`
[7]`http://damien.douxchamps.net/ieee1394/libdc1394/`

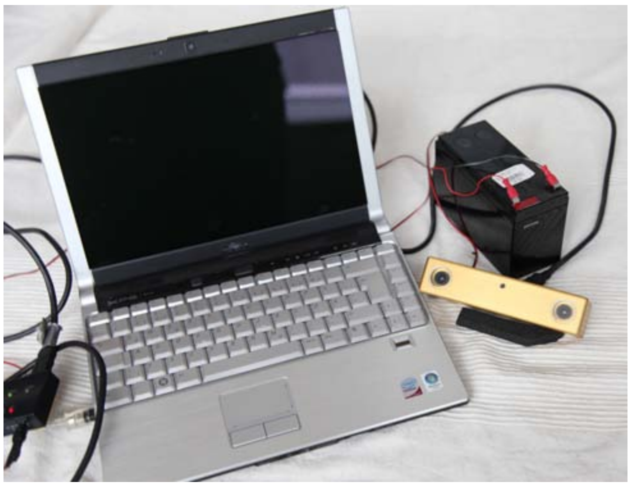


Figure 6.3: Hardware setup. The camera is attached through Firewire to a consumer laptop. For electric power supply we used a 12V lead-acid battery.

door_d) the camera has been moved several times over an office table while manipulating dominant objects . By moving these dominant objects to different places we can evaluate the algorithms robustness to adapt an existing map to the currently visible configuration and the stability regarding data association and pose estimation. The testsets consist of 571, 497, 277 and 349 stereo-image pairs, respectively. We choose a uniform background in order to visually evaluate the performance of the mapping algorithm to repair and update an existing map. A subset of images from each dataset is presented in Figure 6.4.

The *static camera - moving scene* data aims at evaluating localization accuracy, scalability, the algorithms robustness against short-term and noisy scene dynamics and its ability to update and repair an existing map.

Moving Camera - Moving Scene

The second dataset consists of 15443 gray-scale images recorded with a hand-held stereo-rig. A total of 14 independent trajectories named *outdoor_a*, ..., *outdoor_n* have been collected over a trajectory length of 1.4km in a mixed indoor/outdoor scenery. Figure 6.6 presents a schematic overview of the test scenery, where all trajectories are roughly outlined. We grabbed at different times of day, covering an area of approximately $30 \times 130m^2$. To test for relocalization (kidnapped robot problem) the trajectories also differ in their starting point locations. The scenery contains moving people and vehicles, narrow corridors, shiny floors, glass doors, metallic surfaces, different weather

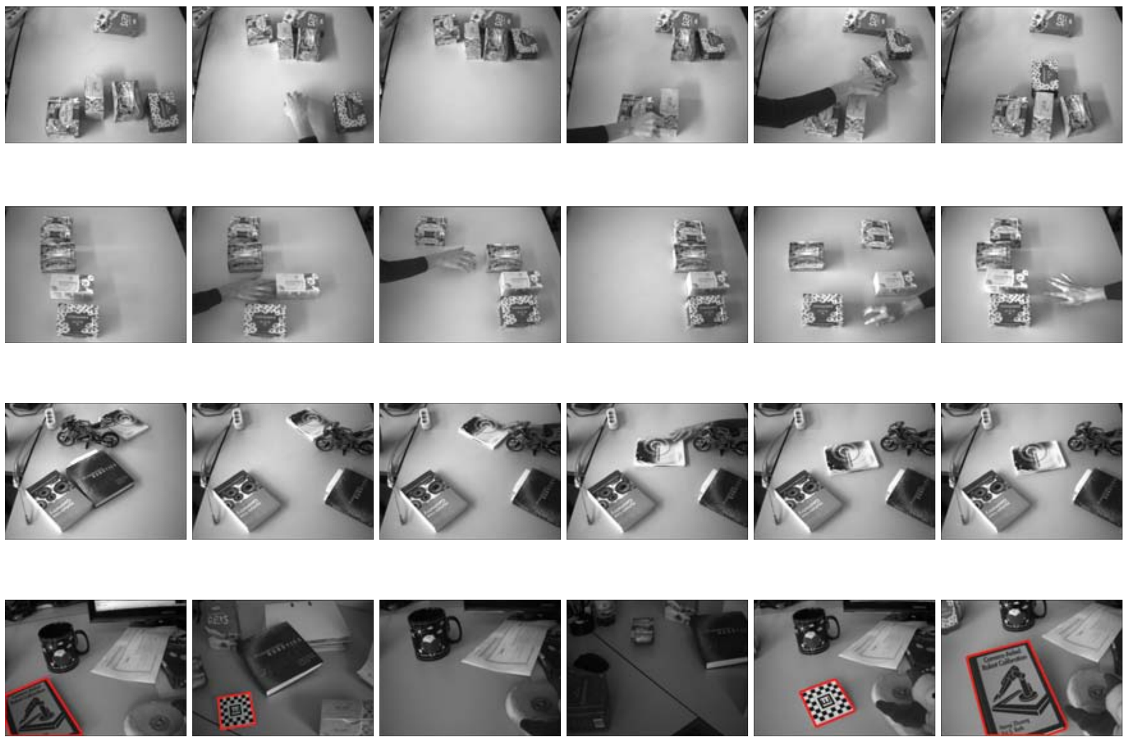


Figure 6.4: Static Camera - Moving Scene. A total of four datasets have been captured where three of them assume a static camera mounted on a tripod with various moving objects inside the robots field of view. In the fourth dataset the camera has been moved over an office desk several times while catchy objects (red) have changed their position or have been deleted outside the robots field of view.

(foggy, sunshine, wet) and lighting conditions (cloudy, sunny, morning, late afternoon) as well as long-term changes such as moved cars, bicycles and furniture. Some prominent images contained in the *CDBench* are presented in Figure 6.5.

When processing all images a visual SLAM algorithm is confronted with many challenging tasks such as robust data association, multiple loop closures, relocalization, kidnapped robot problem, multiple traversals of the same area, storage requirements, processing time as well as map adaption.

The *moving camera - moving scene* dataset aims at evaluating localization accuracy, scalability, the algorithms robustness against scene dynamics, computation time and storage requirements in a long-term experiment.

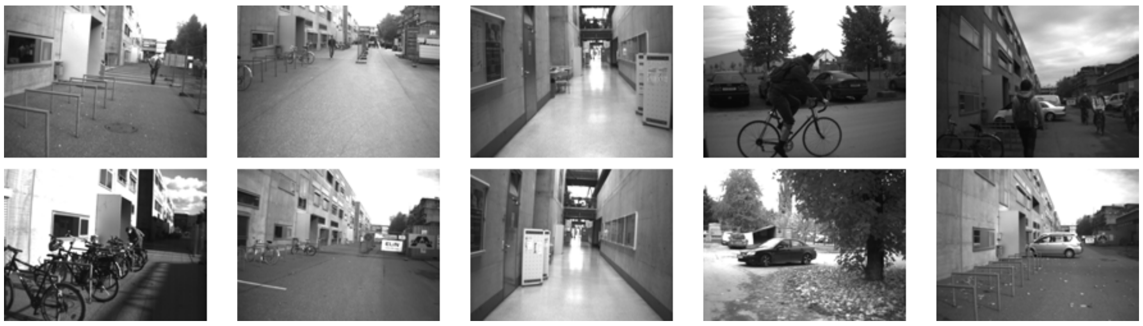


Figure 6.5: Moving Camera - Moving Scene. We captured short-term noise such as moving people or vehicles as well as varying surroundings and different weather conditions. Additionally, multiple occurrences of the same object at different locations affect map consistency and data association.

Groundtruth Acquisition

Since there is no GPS information available indoors we decided to use fixed landmarks such as duct covers, boundary stones, traffic signs or door frames to evaluate localization accuracy and repeatability. Spread over the whole scenery we defined 15 landmarks, both indoors and outdoors. In each testrun a subset of them is passed. Everytime a landmark has been traversed the testruns name and the filename of the currently grabbed image was recorded (an excerpt of the resulting file containing various landmark positions is shown in Table 6.2). Figure 6.7 shows the covered environment from a birds-eye view with the building overlain in red and landmark locations marked as yellow grids. How often a specific landmark has been traversed is summarized in Table 6.3, where all of them have been visited at least three times. Landmarks at crossings have been visited more frequently.

outdoor_a	right_img_000016.pgm	1
outdoor_a	right_img_000201.pgm	8
outdoor_a	right_img_000294.pgm	4
	. . .	
outdoor_d	right_img_001262.pgm	15
outdoor_e	right_img_000007.pgm	1
outdoor_e	right_img_000246.pgm	9
	. . .	

Table 6.2: Groundtruth landmarks. In each testrun (left column) a subset of landmarks (right column) has been passed and the associated images have been recorded (middle column) for accuracy evaluation later on.

Additionally, we manually measured the distance between two adjacent landmarks

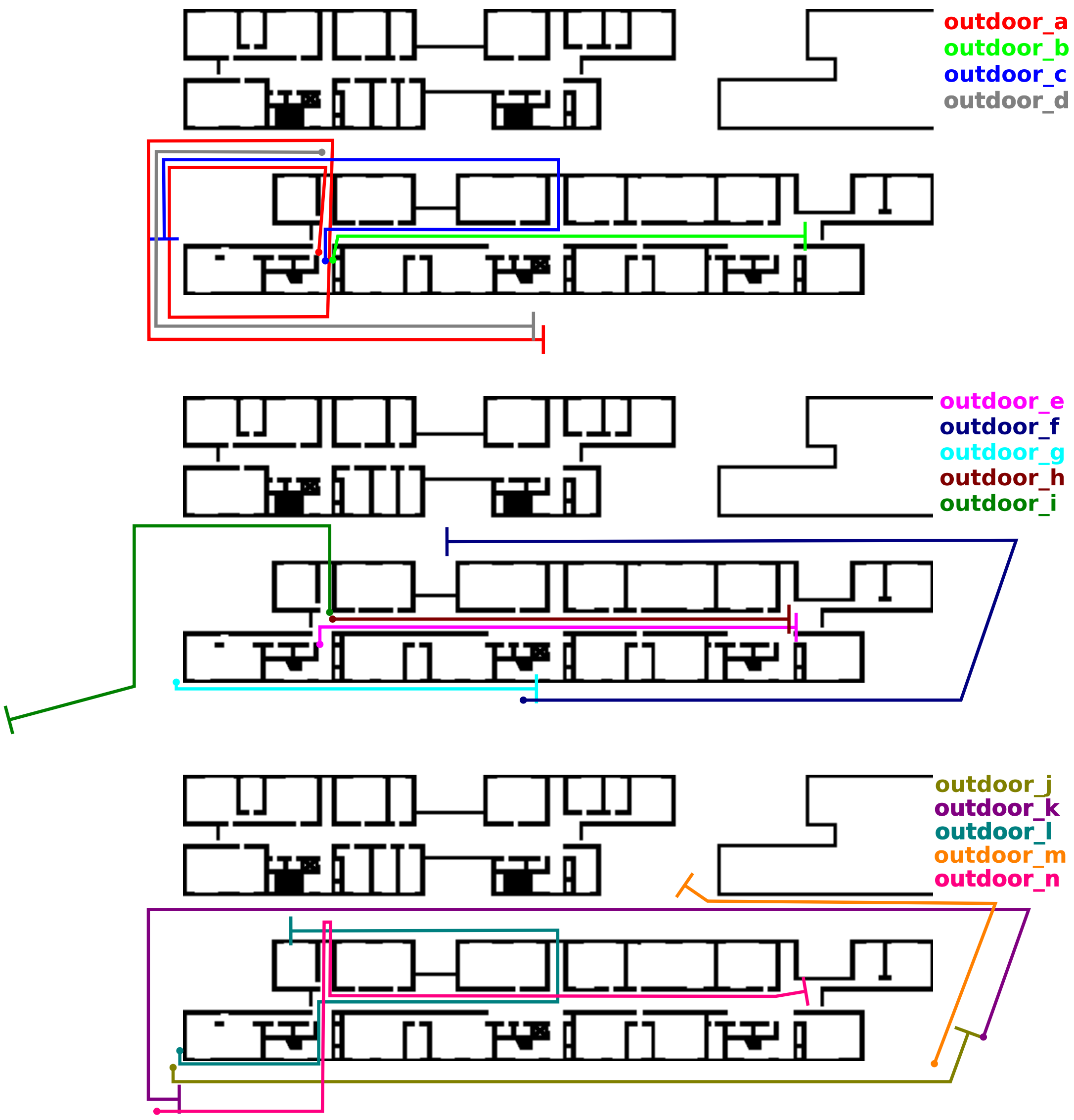


Figure 6.6: Sketched trajectories. 14 different trajectories with different starting point locations have been recorded. The dataset contains images captured at 10 different days.

landmark id	occurrences	landmark id	occurrences	landmark id	occurrences
#1	7	**#9**	6	**#15**	5
#4	7	**#10**	4	**#16**	3
#5	6	**#12**	4	**#17**	3
#6	7	**#13**	4	**#18**	4
#8	3	**#14**	4	**#20**	3

Table 6.3: Landmark occurrences. Each landmark in the testset is at least passed three times, where landmarks near crossings have been visited more frequently.

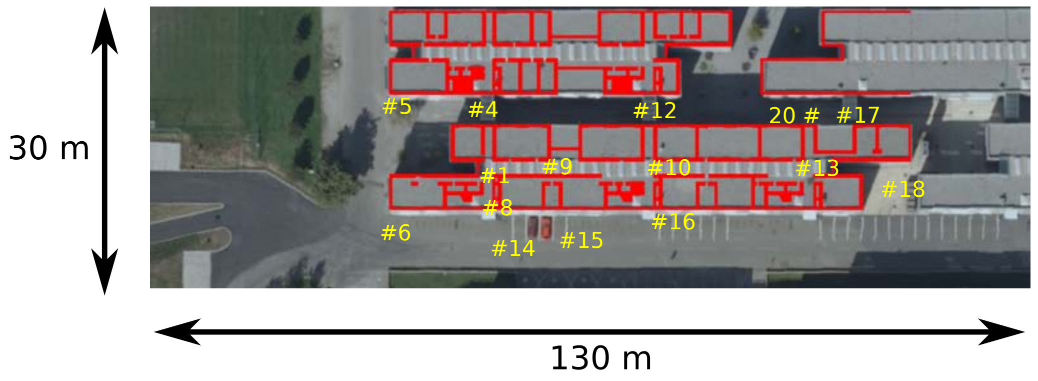


Figure 6.7: Overview of the test scenery. A subset of 15 fixed, predefined landmarks (boundary stone, duct cover, parking places), shown as yellow grids, are passed each run to provide groundtruth information. The building's outline is highlighted red. (courtesy of Google[9]

with a laser distance measuring device provided that they are facing each other. The groundtruth distances are given in Table 6.4.

adjacent landmarks	groundtruth distance [m]
#1 ↔ #9	11.60
#9 ↔ #10	19.61
#10 ↔ #13	25.57
#1 ↔ #8	4.77
#8 ↔ #14	6.38
#1 ↔ #4	13.66
#14 ↔ #6	21.48
#6 ↔ #5	22.94
#5 ↔ #4	18.72
#4 ↔ #12	30.16
#12 ↔ #20	30.32
#20 ↔ #17	6.33
#14 ↔ #15	12.95
#15 ↔ #16	17.63

Table 6.4: Groundtruth distances between adjacent landmarks. These have been measured with a laser range finder, where only measurements between neighboring landmarks facing each other are available.

6.2 Evaluation of HOC-based SLAM

To evaluate the performance and to highlight the benefits of the proposed feature descriptor, we performed a series of simulated and real-world experiments with the stereo-setup described in Section 6.1.2. We compared the outcome of our standard visual SLAM algorithm (-HOC) as described in Section 4.6 with its extension using the HOC-descriptor (+HOC) described in Section 5.2. In both scenarios we evaluated the map growth over time (i.e. scalability) and the pose estimation error (i.e. accuracy), where groundtruth was available.

6.2.1 Synthetic Experiments

The four synthetic experiments were used to demonstrate the performance and correctness of the HOC descriptor. Groundtruth map and sensor poses allow us to directly measure pose estimation accuracy, mapping performance and scalability by monitoring the map size over time.

The four synthetic experiments, named *Simulation_a*, ..., *Simulation_d*, assume a static stereo-system using the camera calibration parameters from the hardware setup described in Section 6.1.2. Objects are represented as 3D pointclouds of variable size covering approximately 10% of the image. Throughout the experiments we captured at 25 frames per second and the resulting image measurements, i.e. projections of the 3D points, were corrupted with Gaussian noise ($\sigma = 0.5px$). Objects were moved and lied approximately 80 to 220 cm in front of the camera. The parameters used for each experiment are summarized in Table 6.5. Some input frames of *Simulation_a* are shown in the top row of Figure 6.8.

testcase	**# frames**	**# objects**	**object velocity**
Simulation_a	62	6	0.6 m/sec
Simulation_b	59	5	0.6 m/sec
Simulation_c	62	5	0.4 m/sec
Simulation_d	62	5	0.6 m/sec

Table 6.5: Synthetic experiments. Parameters used in the four synthetic experiments.

Throughout the synthetic experiments we used our standard visual SLAM framework (named -HOC) described in Section 4.6 taking stereo-images as input. Additionally, the HOC-descriptor proposed in Section 5.2 is added to each landmark (named +HOC). To deal with moving objects we used an icosahedron as HOC-shape with three radii and the Sigmoid function to calculate the importance weight of each bin with $\lambda = 0.9$. In order to discard moved or removed objects we set the threshold parameter $w_{min} = 0.35$ as described in Section 5.2. We evaluated camera pose (translational and rotational error) at every frame and monitored the evolution of the map size over time.

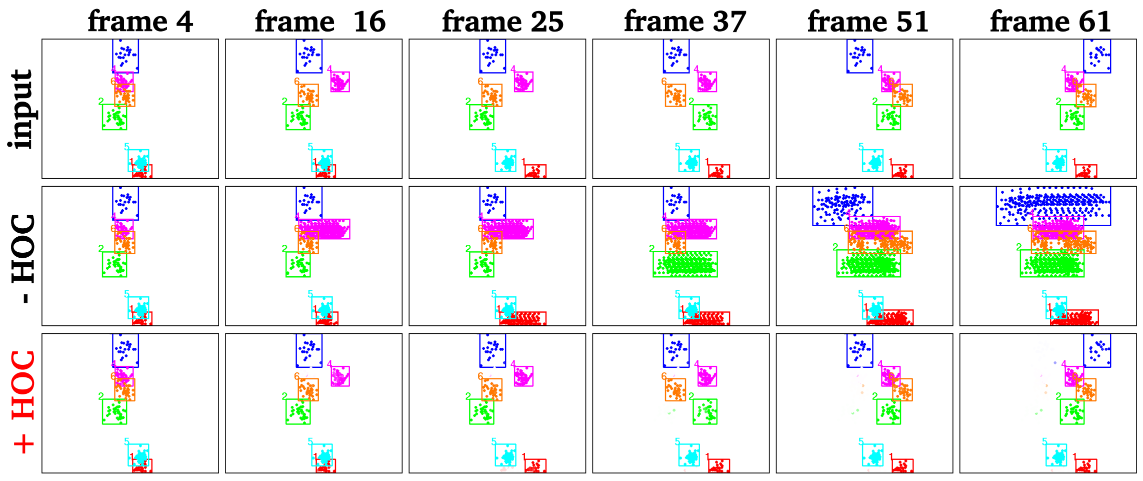


Figure 6.8: Synthetic experiment - map evolution for *Simulation_a*. (top row - input) Six objects move in horizontal direction in front of the stereo camera. We show the left input image at different points in time. (middle row -HOC) Backprojected map estimated with our standard visual SLAM procedure. (bottom row +HOC) Backprojected map constructed by the SLAM algorithm using the proposed HOC descriptor for each landmark. The saturation encodes the weight of each landmark.

Pose errors account for the algorithm's localization accuracy and robustness against scene dynamics. The number of points present in the map are a measure of scalability and map adaption where a constant number of landmarks is preferable. The translational error err_{trans} is given as Euclidean distance between the two camera centers of the estimated pose and the groundtruth pose. To get the rotational error err_{rot} we compute the angle between the viewing directions of the estimated and groundtruth pose. err_{trans} and err_{rot} between the estimated sensor pose $P_e = [R_e|t_e]$ and the given groundtruth pose $P_g = [R_g|t_g]$ are defined as follows:

$$\begin{aligned} err_{trans} &= \| \left(-R_e^T \cdot t_e\right) - \left(-R_g^T \cdot t_g\right) \| \\ err_{rot} &= acos\left(\frac{v_e \cdot v_g}{\|v_e\| \cdot \|v_g\|}\right), \end{aligned} \tag{6.1}$$

where v_e and v_g denote the third column of R_e and R_g, respectively.

Additionally, we compared the position of the landmarks estimated by both approaches to the groundtruth map by computing the Euclidean distance between them. Because of object movements there are multiple occurrences of the same object in the map as visible in Figure 6.8. If images are processed with the HOC descriptor we compare the most likely map point with its associated groundtruth landmark. Here, the most likely landmark is the one with the highest bin weight. If images are processed with our standard visual SLAM algorithm we compute the Euclidean distances to the nearest and most distant corresponding landmark.

error measure			**testcase**			
			Sim a	***Sim b***	***Sim c***	***Sim d***
translational error [mm]	- HOC	mean (max)	12.06 (179.10)	8.6 (79.8)	65.0 (205.7)	113.5 (352.5)
	+ HOC	mean (max)	0.97 (41.49)	0.2 (0.4)	0.3 (0.8)	0.4 (2.1)
rotational error [deg]	- HOC	mean (max)	0.45 (6.10)	0.3 (2.8)	0.03 (0.4)	2.8 (8.7)
	+ HOC	mean (max)	0.05 (2.41)	0.004 (0.01)	0.0004 (0.001)	0.005 (0.01)
mapping error [mm]	- HOC	min	15.24	14.07	21.47	16.75
		max	156.72	156.45	177.49	200.60
	+ HOC	mean	10.41	9.75	7.26	8.39
map size [pts]	GT		277	246	264	264
	- HOC		1526	1670	3035	3425
	+ HOC		435	441	644	825

Table 6.6: Results of the four synthetic test sequences. We compared the standard SLAM implementation (-HOC) to the one incorporating visibility information (+HOC). We present mean and (maximal) translational and rotational errors of the estimated sensor pose as well as the deviation of the estimated map to the groundtruth. Furthermore, the groundtruth map size (input) and the final map size after the last processed frame using both approaches is given.

Mapping and pose errors are evaluated for every frame and the averaged results are summarized in Table 6.6, where mean and maximal errors are given for the translational and rotational component. The average mapping error for the HOC approach is given, where for the standard SLAM procedure (-HOC) the mean over the minimal and maximal distances is provided.

In contrast to the standard SLAM approach the additional visibility information encoded in the HOC descriptor drastically boosts localization and mapping accuracy. Here, the error ranges between 0.2 and 0.97mm whereas the standard SLAM procedure produces much larger pose errors ranging from 8.6 to 113.5mm. Since objects are moving sideways the translational error is much higher than the rotational offset. Furthermore, the map size can be reduced to a quarter which greatly enhances data association in terms of matching performance and speed while simultaneously improving the pose estimates. When compared to the groundtruth (GT), our standard SLAM approach produces up to 13 times more landmarks. The HOC descriptor greatly reduces the number of map points but did not reach the minimum map size as given by the groundtruth. This may arise from the effect that some bins of a landmark are not downweighted properly due to some jitter in the camera pose. This aliasing effect is caused by a slight deviation of the pose, where the camera crosses the bin borders and previously upweighted bins are not downweighted properly anymore. We also reach better mapping accuracy when incorporating the new descriptor. Here, the mean deviation ranges between 7.26 to 10.41mm only. The standard SLAM procedure instead produces much higher map points offsets.

The evolution of the backprojected map points compared to the groundtruth over time is shown in Figure 6.8. The bottom row represents the image of the estimated map when incorporating the HOC descriptor, which is nearly similar the groundtruth image (top row). Hereby, old positions of the object disappear after some time since they get a lower weight during localization. More recent positions instead are given a higher weight and assist the localization process. Constructing the map without visibility information steadily increases the map size and results in multiple occurrences of the same object in the map (see middle row).

6.2.2 Static Camera - Moving Scene

These experiments are used to demonstrate the algorithm's robustness against short-term and noisy dynamics. A groundtruth sensor pose allows us to directly measure pose estimation accuracy. Again the evolution of the map size over time provides a measure for the system's scalability. We used the four indoor sequences of the benchmark dataset described in Section 6.1.2. For exemplary images see Figure 6.4.

Throughout the experiments we take the stereo-images as input. Each landmark is further assigned a HOC descriptor with spheres approximated by octahedrons. Since the testset only contains images captured close to the objects we decided to use four distance bins. As weighting function we employ the sigmoid function with $\lambda = 0.9$. To get rid of vanished map points we compute the histograms maximum value and discard those with $w_{min} < 0.35$.

Again we evaluated the translational and rotational error using Equation 6.1 for all testsets where groundtruth was available (*indoor_a*, ..., *indoor_c*). We confront the pose accuracy of our standard SLAM algorithm (-HOC) proposed in Section 4.6 with the one using the HOC descriptor (+HOC). We also compare the map size after the final frame has been processed. The accuracy, completeness and actuality of the resulting map can only be inspected visually. Table 6.7 presents the mean and maximum pose errors for each real-world experiment and the final map size. Similar to the synthetic experiments we gain pose accuracy and drastically reduce the number of landmarks when incorporating visibility information during SLAM.

A visual comparison of the estimated maps and trajectories of the *indoor_b* and *indoor_d* testsets are given in Figures 6.9 and 6.10. Using visibility information from the HOC descriptor the map always represents the most recent environmental configuration, while out-of-date landmarks disappear after some time. Small relicts (light gray map points on he right) result from the aliasing effect already mentioned in Section 6.2.1. On the contrary, including every sensor measurement results in overcrowded maps where up to eight times more landmarks are present in the standard case. Multiple occurrences of the same object lead to wrong data associations and pose estimates as demonstrated in Figure 6.9 and Table 6.7. In Figure 6.10 the moved and deleted objects are highlighted red for better visibility. When using the HOC descriptor one can see that the map always contains the most recent objects, provided that the sensor is visiting the same place again. Deleted objects disappear in the map (second frame) and reappear (third and fourth frame) at the correct place. Furthermore, a map constructed with the HOC descriptor contains only the most prominent features while noisy map points caused by inaccurate or wrong triangulations are filtered out. It has to be mentioned, that the occluded object - although providing prominent, stable features - will disappear after some time, since its bins are getting downweighted. On reappearance it is added to the map as a new landmark. Since landmark importance is defined by the number of positive observations this is a valid behaviour of our algorithm.

In addition, we also evaluated re-localization performance every 30-th frame where groundtruth is available. Therefore, we perform exhaustive SIFT-matching of the features contained in the current frame against all features in the map. The HOC descriptor can be used to filter out the most probable map points by looking at the histogram's maximum value, which reduces matching effort and boosts data association accuracy. The camera pose is then estimated through a 3-point RANSAC between $3D \leftrightarrow 3D$ correspondences.

	translational error [mm]		rotational error [deg]		map size [pts]	
testcase	+ HOC	- HOC	+ HOC	- HOC	+ HOC	- HOC
indoor_a	5.67 (18.05)	16.53 (28.94)	0.28 (1.03)	0.82 (1.44)	816	7904
indoor_b	17.56 (55.71)	190.21 (281.42)	0.95 (3.07)	8.99 (12.94)	1593	11899
indoor_c	1.57 (25.61)	15.57 (20.36)	0.07 (1.00)	0.81 (1.10)	855	6192
indoor_d	-	-	-	-	3817	6838

Table 6.7: Static camera - moving scene: Results of the four real-world test sequences with a rigidly mounted stereo rig. We compared our standard SLAM implementation (-HOC) to the one incorporating visibility information (+HOC). We present mean and maximal translational and rotational errors of the estimated sensor pose when compared to groundtruth. Furthermore, the final map size after the last processed frame using both approaches is given.

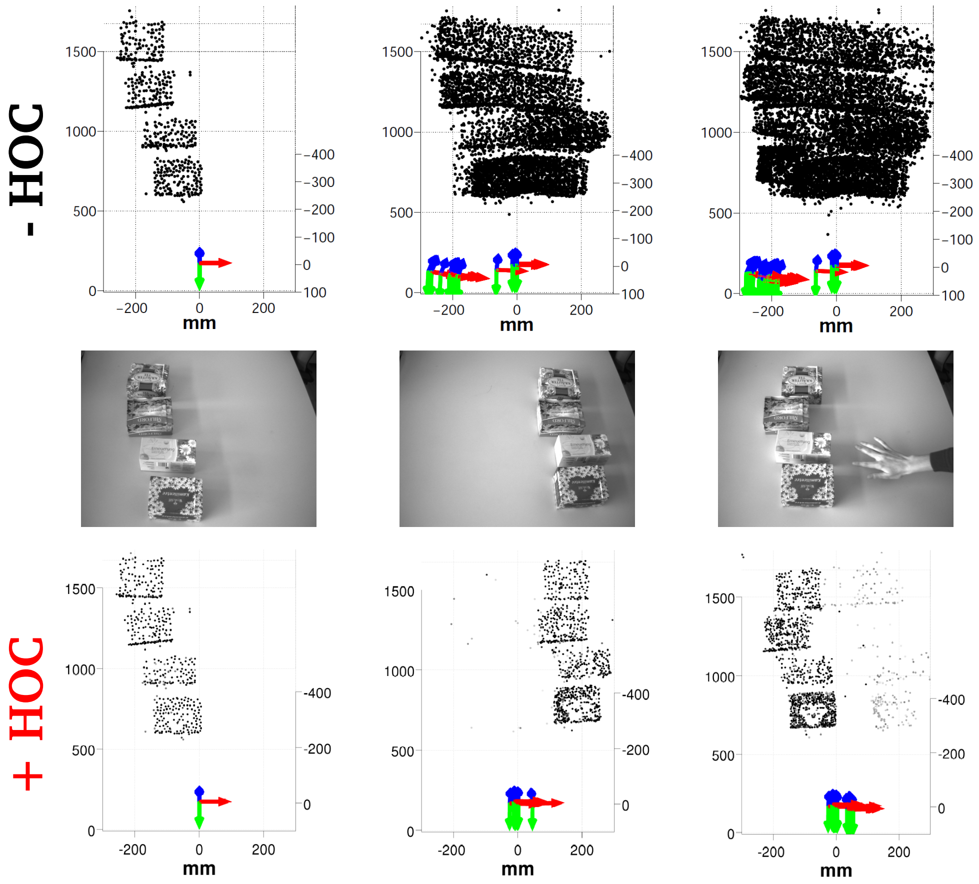


Figure 6.9: Results of the *indoor_b* testset. We show estimated camera poses and landmarks with (+HOC) and without (-HOC) using visibility information. Input images are given for visual map comparison.

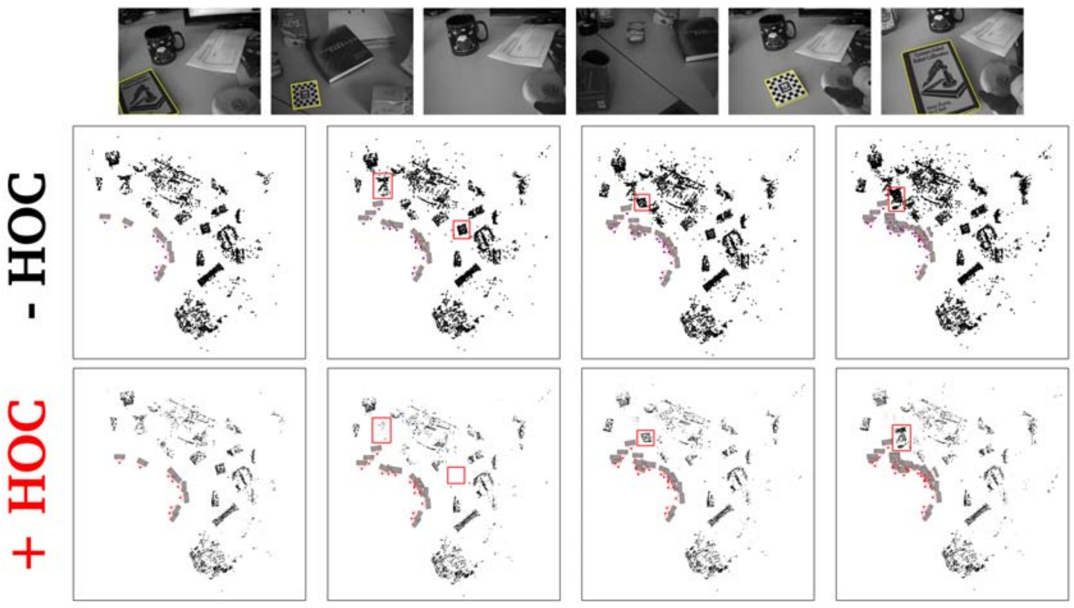
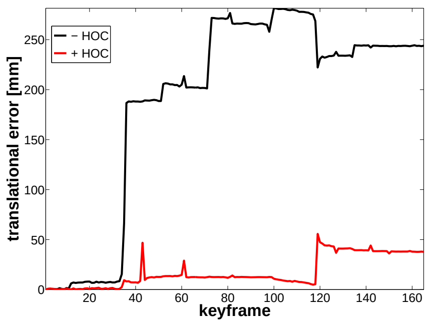


Figure 6.10: Results of the *indoor_d* testset. A camera has been moved several times over an office table where prominent objects (yellow) have been transfered to different places. We show the resulting maps estimated by our standard SLAM approach (- HOC) and the algorithm using the HOC descriptor (+HOC). For better visualization the dynamic objects are marked red in both landmark maps.

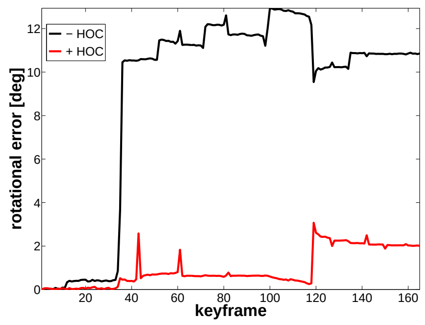
The results for both translational and rotational error are summarized in Table 6.8. As expected even the re-localization accuracy is increased since the map constructed with the HOC descriptor only contains the most recent objects. A graphical visualization of the results for pose error, scalability and relocalization of the *indoor_b* dataset is presented in Figure 6.11.

	translational error [mm]		**rotational error [deg]**	
testcase	+ HOC	- HOC	+ HOC	- HOC
indoor_a	6.67 (28.54)	15.89 (41.27)	0.31 (1.31)	0.78 (1.96)
indoor_b	20.06 (76.36)	187.66 (281.20)	1.03 (3.53)	8.86 (12.75)
indoor_c	3.53 (8.62)	9.23 (17.04)	0.19 (0.43)	0.56 (1.10)

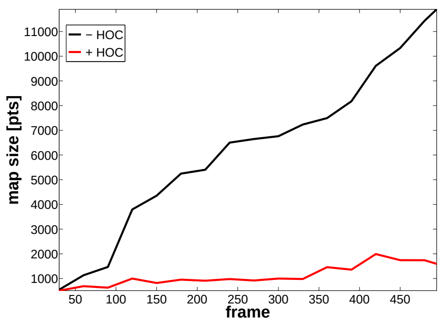
Table 6.8: Static camera - moving scene: Re-localization performance evaluated at three real-world test sequences assuming a static camera. We compared our standard SLAM implementation (-HOC) to the one incorporating visibility information (+HOC). We present mean and maximal translational and rotational errors of the estimated sensor pose when compared to groundtruth.



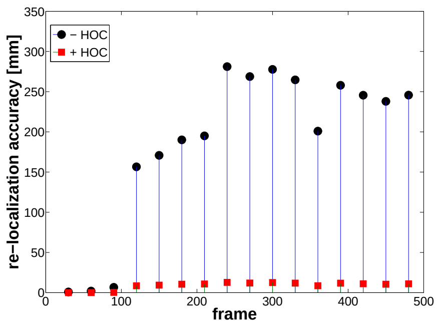
(a)



(b)



(c)



(d)

Figure 6.11: Static camera - moving scene: Graphical visualization of the *indoor_b* dataset. Translational (a) and rotational (b) error for every keyframe. (c) Map size over time. (d) Re-localization accuracy of the translational component. Our standard SLAM approach (black) is compared to the one using the HOC descriptor (red).

6.2.3 Moving Camera - Moving Scene

The experiment is used to investigate speedup and accuracy during descriptor matching (i.e. data association), the system's scalability and localization accuracy. The improvements during data association are demonstrated on the long-term dataset dataset. Scalability and localization accuracy instead are shown in a small, clearly represented area only. The algorithms performance in a long-term experiment is addressed in Section 6.3.

Throughout these experiments we only use the left image of the stereo-pair and perform monocular visual SLAM as described in Section 4.6. Again we employ the HOC descriptor to react on environmental changes (compare Section 5.2.5). Since we are traveling in a wider area we use an icosahedron as underlying shape with nine radii.

To compute the weight of he bins we used the probabilistic weighting scheme described in Section 5.2.4 with $s = 5$. To delete vanished map points we set $w_{min} = 0.3$. Each keyframe is further stored in a pose graph and keyframe insertion is done on demand as described in Section 5.3.

Data association

Employing the visibility information during pose tracking as described in Section 5.2.5 greatly reduces the average matching candidates per frame. For demonstration purpose Figure 6.12 shows the number of visible map points per frame while processing the *outdoor_1* testrun highlighted green in Figure 6.18. Using the HOC descriptor during tracking (red) reduces the matching candidates from 7753 to 902 on average, compared to matching without considering visibility (black). This results in a speedup of factor 4. Especially when traveling in z-direction (i.e. viewing direction of the camera) in the map, we greatly reduce possible matching candidates. This can be seen particularly at the three black ridges in Figure 6.12.

We also summarized the averaged number of visible points per frame for each of the 14 testruns separately in Table 6.9. Notably we got a reduction of up to 91.18% per frame. The reduced number of putative matching candidates per frame results in less false matches, a speed up of the tracking process and more accurate localization results (as shown later in this section).

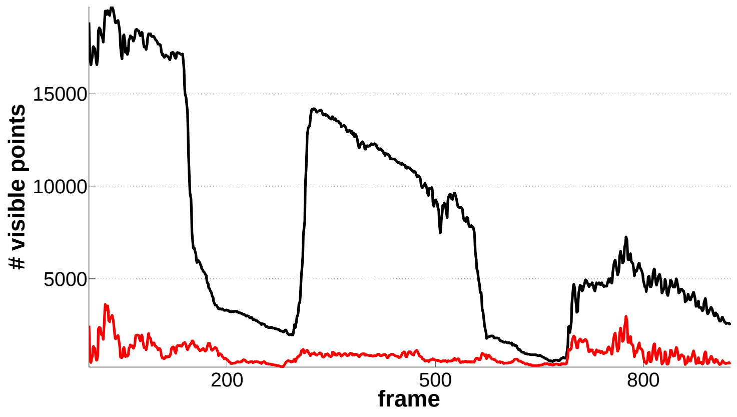


Figure 6.12: Moving camera - moving scene: Data association evaluation. Putative matching candidates per frame with (red) and without (black) using visibility information for the run highlighted green in Figure 6.18 are compared. We reduced the number of visible map points from 7753 to 902 per frame on average.

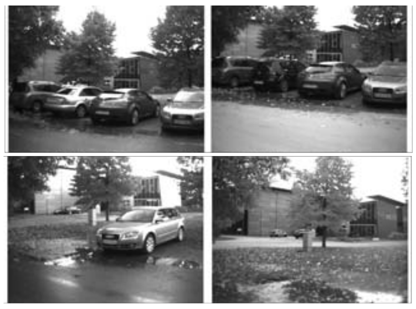
testrun	**avg. matching effort [pts]**		**reduction [%]**
	+ HOC	- HOC	
outdoor_a	292	512	42.92%
outdoor_b	237	423	43.99 %
outdoor_c	395	1614	75.52 %
outdoor_d	444	1823	75.63 %
outdoor_e	272	1900	85.68 %
outdoor_f	585	2746	78.71 %
outdoor_g	923	9179	89.94%
outdoor_h	523	5932	91.18%
outdoor_i	444	901	50.68%
outdoor_j	406	3287	87.64%
outdoor_k	761	3968	80.83%
outdoor_l	902	7753	88.36%
outdoor_m	1187	7998	85.16%
outdoor_n	2704	7796	65.31%

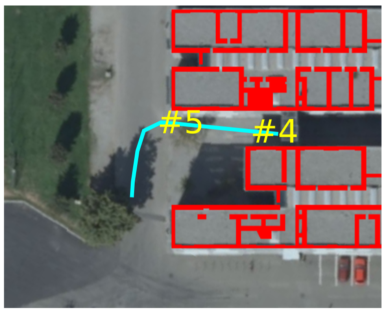
Table 6.9: Moving camera - moving scene: Data association evaluation. We present the average number of visible points per frame for each testrun separately with (+HOC) and without (-HOC) considering the visibility information. The percentaged reduction is also given.

Scalability

To highlight the systems scalability, we only considered a subset of images traversing the same area four times. The reasons for considering only a small map portion are two-fold: First, the area has been recorded four times at different days at changing weather conditions. Hence, the algorithms robustness against long-term, short-term and radiometric dynamics is evaluated. Second, it provides a more descriptive example than processing the whole dataset and allows better visualization and understanding of the whole process. Figure 6.13 shows images from the processed subset named *subset_1*, ..., *subset_4* and a snapshot of the traversed area. We evaluated the evolution of the map size over time and the putative matching candidates per frame by comparing the systems performance with (+HOC) and without (-HOC) visibility information.

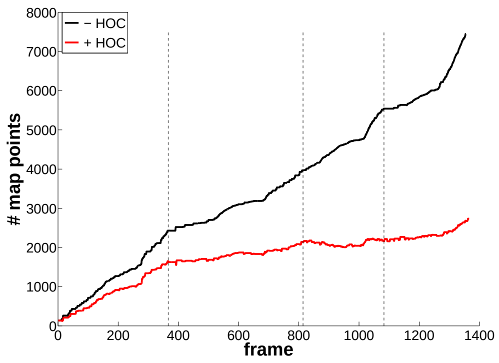
Map growth over time is shown in Figure 6.14(a) and map size after each run is listed in Table 6.10. The map size grows more rapidly in our standard SLAM procedure (-HOC), since no landmark deletion took place. Even when using the HOC descriptor a small increase can be observed, which is explained by the ever changing environment where new landmarks have to be established to guarantee an up-to-date map. As byproduct we reduced the average matching effort per frame each run as shown in Table 6.10 and Figure 6.14(b).

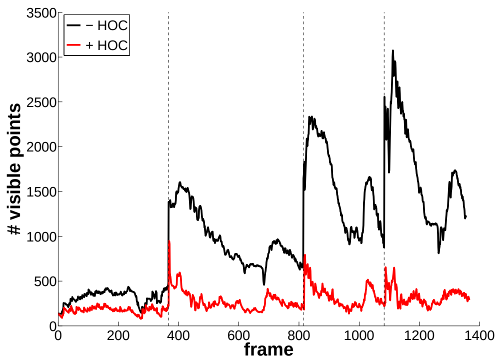




(a) (b)

Figure 6.13: Analysis of the systems scalability. (a) Images taken from the four testruns traversing the same area at different days. (b) Snapshot of the subarea traversed four times to evaluate the systems scalability and localization performance. Two landmarks are passed each run.





(a) (b)

Figure 6.14: Moving camera - moving scene: Analysis of the systems scalability. (a) Map growth over time throughout the four runs with (red) and without (black) visibility information from the HOC descriptor. (b) Per frame matching effort throughout the four testruns with (red) and without (black) incorporating the HOC descriptor.

	map size (increase) [pts]		avg. matching effort [pts]	
run	**- HOC**	**+ HOC**	**- HOC**	**+ HOC**
subset_1	2424	1461	322.90	222.03
subset_2	3969 (+1540)	1749 (+288)	962.18	530.36
subset_3	5541 (+1572)	1752 (+3)	1536.82	657.80
subset_4	7448 (+1907)	2045 (+293)	1716.64	621.10

Table 6.10: Moving camera - moving scene: Analysis of the systems scalability. We compared the map size and average matching effort per frame with (+HOC) and without (-HOC) using the proposed descriptor.

Localization Accuracy

During the traversal of the four subsets in the restricted area we also passed two landmarks (#4 and #5) every testrun as shown in Figure 6.13(b). To get a measure for localization accuracy we compute the mean deviation from the centroid for the i-th landmark as follows:

$$\begin{aligned} C^i &= \frac{\sum_{j=1}^{n} C_j^i}{n} \\ err^i &= \frac{\sum_{j=1}^{n} \left(\|C_j^i - C^i\| \right)}{n}, \end{aligned} \tag{6.2}$$

where n is the total number of traversal of landmark i. The error measures for both landmarks are presented in Table 6.11. Due to the huge variety in the input data like moved cars or changing weather conditions (compare Figure 6.13(a)) our standard SLAM approach fails during localization in the fourth run. Furthermore, without using the localization importance encoded in the HOC descriptor the map gets polluted with already vanished landmarks or wrongly triangulated map points. Figure 6.15 shows the estimated trajectories (red) and the underlying map from a birds-eye-view when using both approaches on the restricted area. The blue dots mark the estimated positions of the traversed landmarks, where our standard SLAM approach fails in the fourth run.

6.3 Evaluation of the Novel Pose Graph, the Adaptive Keyframe Selection and Map Point Insertion

In order to evaluate the performance of the pose graph together with the adaptive keyframe selection and map point insertion routine described in Section 5.3, we pro-

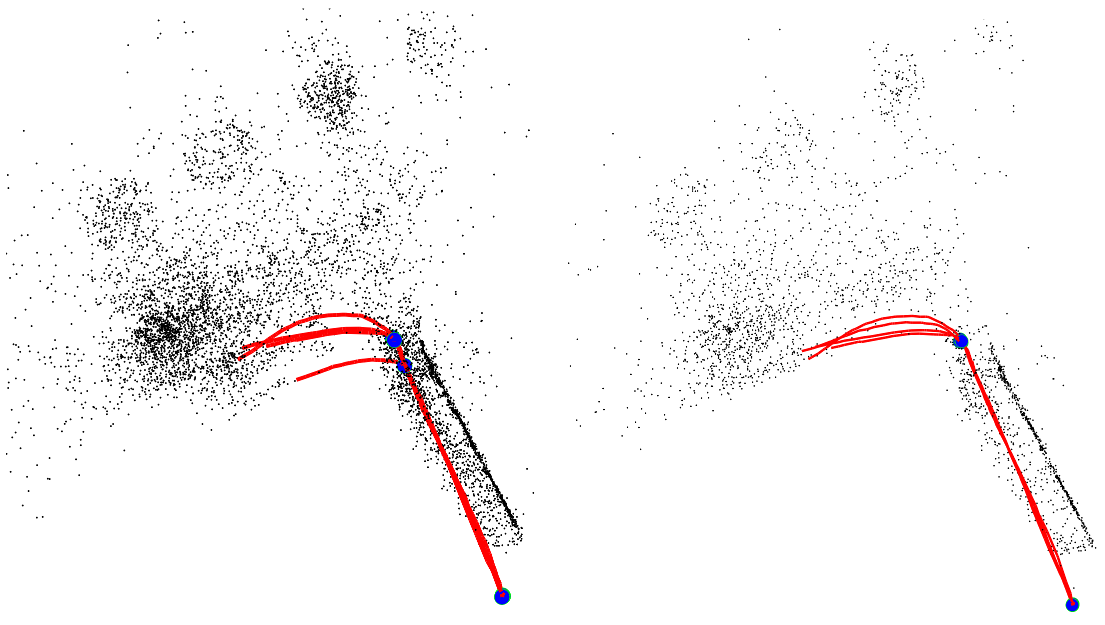

(a) (b)

Figure 6.15: Moving camera - moving scene: Localization accuracy. Map and trajectories estimated by the visual SLAM algorithms. The positions of the two traversed landmarks are marked in blue. (a) Estimated environment when using our standard SLAM algorithm. (b) Results when incorporating the HOC descriptor.

landmark	**mean deviation [cm]**	
	- HOC	**+ HOC**
#4	4.46	3.71
#5	73.68	9.78

Table 6.11: Moving camera - moving scene: Accuracy evaluation. We processed the images of the four subsets taken in the restricted area. We compared the mean deviation from the centroid of two landmarks (#4 and #5) with (+HOC) and without (-HOC) using the proposed descriptor.

cessed the entire outdoor dataset (see Section 6.1.2). Here, we focused especially on the life-long mapping context and evaluated the localization accuracy as well as the scalability of our approach.

Throughout this experiment we take the monocular grayscale images as input and perform incremental SfM combined with the HOC descriptor described in Section 5.2. We use an icosahedron as underlying shape with nine radii. To assign a weight to each bin we used the probabilistic weighting scheme described in Section 5.2.4 with $s = 5$.

To delete vanished map points we set $w_{min} = 0.3$. Each keyframe is further stored in the pose graph and both, keyframe and map point insertion is done on demand as described in Section 5.3.

Localization Accuracy

To evaluate the localization accuracy we saved the estimated camera projection center $C_j^i = -R^T \cdot t$ whenever a groundtruth-image has been processed. Ideally, all pose estimates C_j^i of landmark i should be identical. So the mean deviation from the centroid C^i of all estimates acts as a measure for localization repeatability and stability of the whole system (compare Equation (6.2), describing the proposed error measure err^i for landmark i). It is important to note that the first pose estimate for each landmark is computed before loop closure took place. Therefore, it is omitted from the centroid computation and its localization error is given separately. This scenario is also visualized in Figure 6.16 showing a closeup of all recorded poses for landmark # 4. There is a single outlier contained in the pose estimates stemming from the trajectory estimation before loop closing took place and therefore only the poses after loop closure correction green box) are considered for accuracy evaluation.

Within each of the 14 trajectories we passed a subset of predefined landmarks exactly and recorded the estimated pose. The mean deviation from the centroid of all pose estimates is presented in Table 6.12. We successfully (re)localized over the whole trajectory while exploring perpetually. The mean deviation ranges between 1.36cm and 10.84cm, which are reasonable numbers for hand-held recording. Landmarks #13 and #10 are outliers where localization failed when processing testrun *outdoor_n* evoked by high lighting variations in the input data. Here, existing map points have not been recognized and a new map has been established. The wrong trajectory would be corrected if reentering the map at a known position and correcting the wrong pose estimates through loop closing.

Additionally, we computed the deviation between adjacent landmark centroids and compare them to the groundtruth distance given in Table 6.4. Since groundtruth distances are measured in a nearly planar region, above computed landmark centroids C^i have to lie in the same plane before computing the distances between them. The computation of the distance between neighboring landmark works as follows:

- A plane is fitted to all camera centers over the whole trajectory using a RANSAC approach to retrieve the plane parameters.

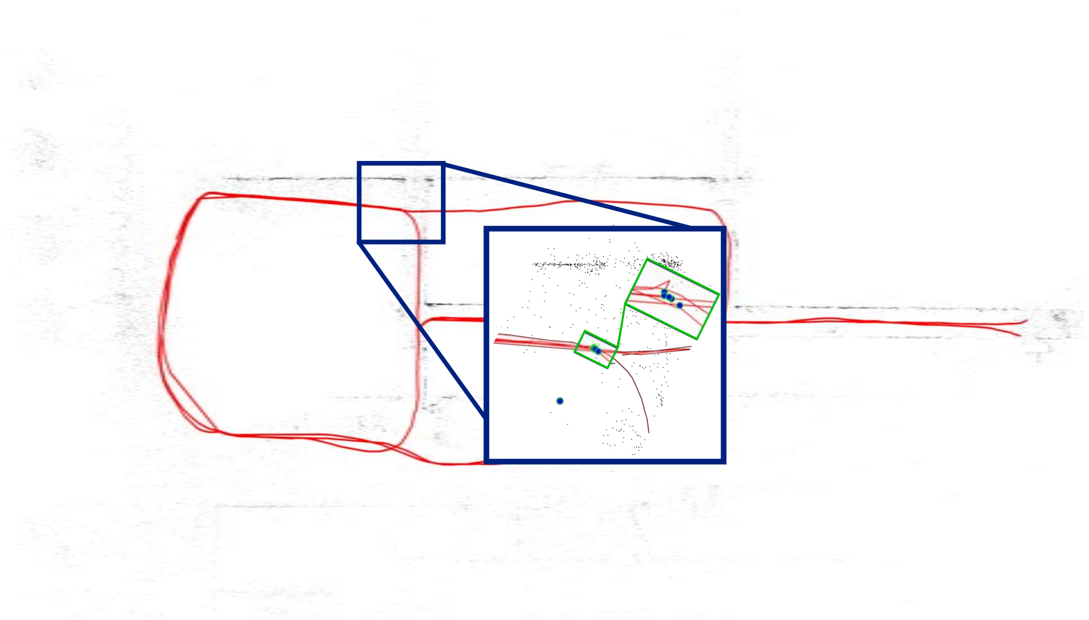


Figure 6.16: Accuracy evaluation. Recorded camera poses when passing landmark # 4 are marked as blue circles. The final trajectory (red) is shown together with the estimated map points (black). The single pose on the left corner has been recorded before loop closure correction took place. Localization accuracy is only evaluated at the poses recorded after loop closing (green window).

landmark	**average err [cm]**	1^{st} **pose error**
#1	6.64	176.40
#4	5.70	290.00
#5	6.01	3.22
#6	7.50	86.59
#8	5.39	180.91
#9	10.84	12.03
#10	65.14	94.70
#12	3.44	78.01
#13	238.70	195.06
#14	4.58	8.91
#15	10.31	23.09
#16	3.06	4.77
#17	1.36	70.87
#18	4.42	20.07
#20	1.56	82.32

Table 6.12: Localization accuracy. Average deviation from the centroid at each landmark after loop closure correction is measured. Deviations for poses before loop closing are listed separately (right most column).

- All centroids C^i are projected onto the plane.
- The distances $d_{l,m}$ between facing landmark centroids l and m (see Table 6.4 for neighboring landmarks) are described by the Euclidean distance measure.
- Since we are performing monocular SLAM, the estimated distance $d_{l,m}$ and groundtruth distances $\hat{d}_{l,m}$ are related by a scaling factor s, which is computed the following way:
  - The scale estimates $s_{l,m}$ between $d_{l,m}$ and $\hat{d}_{l,m}$ are simply given through $s_{l,m} = \frac{\hat{d}_{l,m}}{d_{l,m}}$.
  - In order to be robust against wrong centroid estimates, the overall scaling factor s is calculated by taking the median estimate of all $s_{l,m}$.
- The final error measure $err_{l,m}$ between estimated $d_{l,m}$ and groundtruth distances $\hat{d}_{l,m}$ is given through $errl, m = |\hat{d}_{l,m} - s \cdot d_{l,m}|$.

The deviation between groundtruth and estimated landmark distances is summarized in Table 6.13. As expected the error between landmarks #10 and #13 is remarkably high, because even their deviation from the centroid varies a lot (compare Table 6.12).

landmarks $l \leftrightarrow m$	error $err_{l,m}$ [m]
#1 $\leftrightarrow$ #9	0.09
#9 $\leftrightarrow$ #10	0.49
#10 $\leftrightarrow$ #13	6.87
#1 $\leftrightarrow$ #8	0.06
#8 $\leftrightarrow$ #14	0.43
#1 $\leftrightarrow$ #4	0.18
#14 $\leftrightarrow$ #6	0.19
#6 $\leftrightarrow$ #5	0.54
#5 $\leftrightarrow$ #4	0.31
#4 $\leftrightarrow$ #12	0.39
#12 $\leftrightarrow$ #20	0.61
#20 $\leftrightarrow$ #17	0.09
#14 $\leftrightarrow$ #15	0.18
#15 $\leftrightarrow$ #16	0.13

Table 6.13: Accuracy evaluation comparing groundtruth distances to estimated ones between adjacent landmarks. The euclidean distances between neighboring landmark centroids has been measured and, after applying an overall scaling factor, compared to the groundtruth measures reported in Table 6.4.

The final map and trajectory are overlain on a satellite image as shown in Figure 6.18. We successfully built a correct map and trajectory of the surrounding area. Only the large loop as shown in Figure 6.17 has not been corrected properly since there is a small offset between the dots representing the facades and the groundplan at the rightmost side of the image near landmark #18. Here, the scale drift has not been estimated exactly during loop closure correction resulting in a slightly wrong correction correction of map points and camera poses. Full 3D models after processing all 14 trajectories of the *CDBench* dataset are shown in Figure 6.19.

Scalability

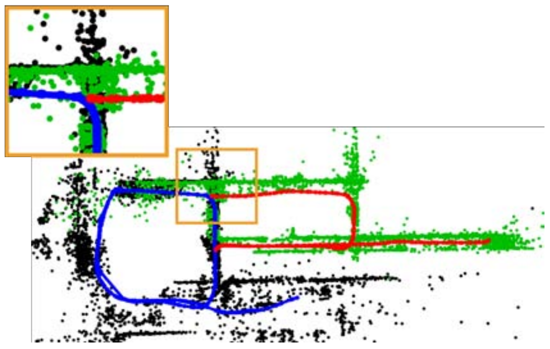
Scalability is measured by the reduction in the number of points and poses to be optimized during the loop closing procedure. The loop closing strategy presented in Section 5.3 boosts computing time and scalability. Most of the time only small inconsistent parts have to be added to a large, self-contained map. The situation for the loop closures while processing the outdoor sequence is shown in Figure 6.17. We successfully identified the subset of camera poses to be optimized (red), while the larger part of the map already corrected remains fixed (blue). Even subsequent structure bundling can be reduced to a small subset of points only (green dots), which speeds up the whole loop closing procedure. In total we correctly detected three loop closures. The reduction in the number of poses and points to be optimized (compared to our standard approach taking all poses and points into account) is given in Table 6.14.

	+(pose graph)		**-(pose graph)**	
loop closure	# points	# poses	# points	# poses
# 1	6434	240	6434	240
# 2	5636	244	11486	569
# 3	8805	346	21 499	1148

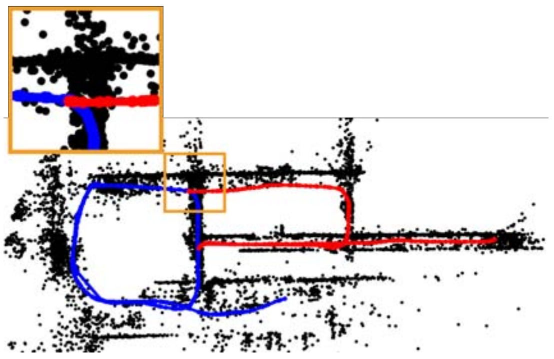
Table 6.14: Scalability evaluation. Utilizing the pose graph to detect affected map regions for loop closing greatly reduces the parameters to be optimized in contrast to standard methods.

6.4 Conclusion and Discussion

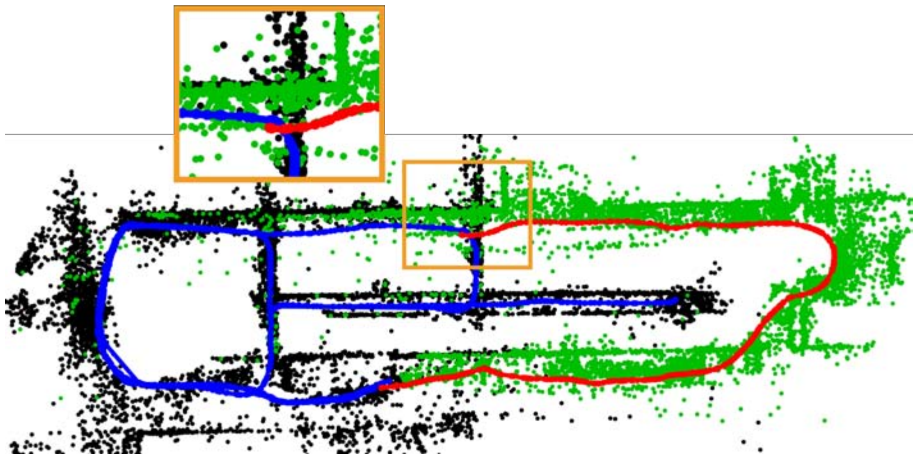
In this chapter we performed extensive experiments, including both simulated and real-world setups, to evaluate the performance of the novel three-dimensional landmark descriptor HOC, the adaptive loop closing algorithm and keyframe selection routine. Since we mainly concentrate on life-long mapping within a dynamic world, we recorded and



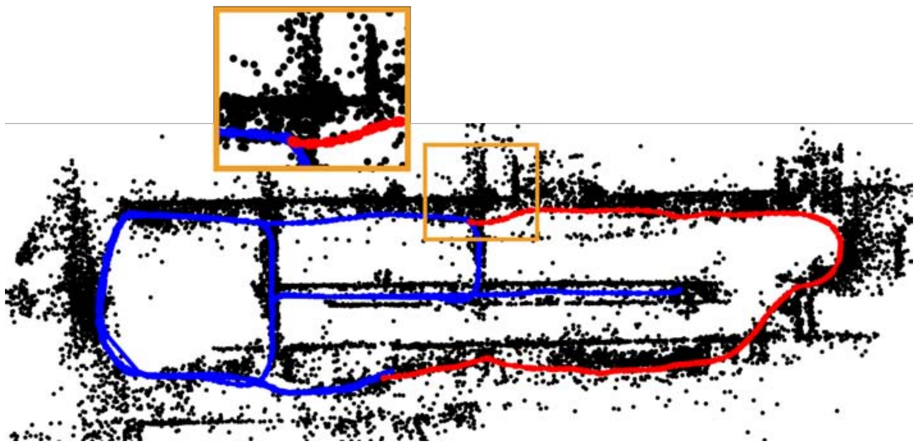
(a)



(b)



(c)



(d)

Figure 6.17: Accelerated loop closing. (a),(c) Instead of performing structure and motion refinement over the whole trajectory, we establish a subset of poses (red) through efficient graph traversal. Also structure estimation reduces to a smaller subset of points to be optimized (green). (b),(d) Successfully corrected map and trajectory.

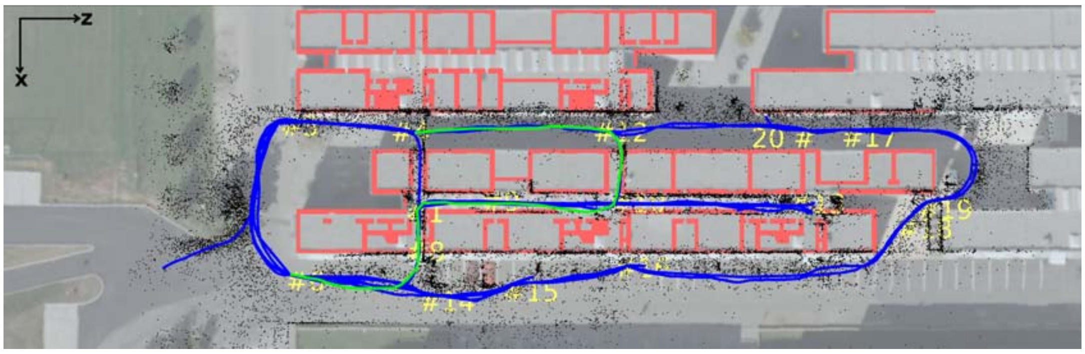
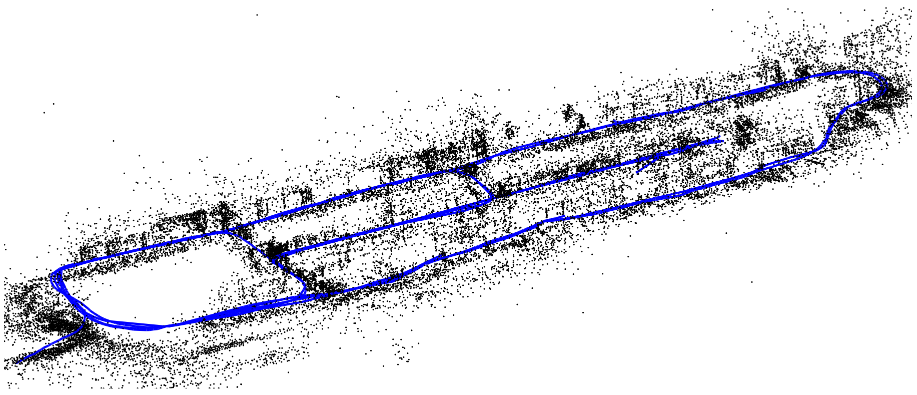
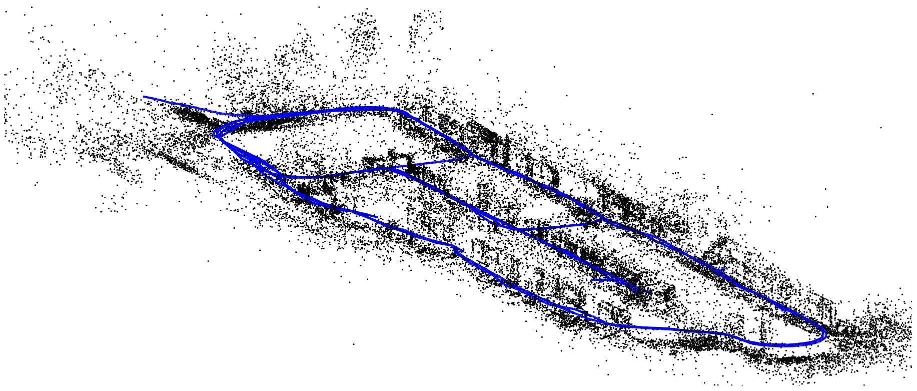


Figure 6.18: Result of the outdoor dataset. The final map (black dots) and trajectories (blue) overlain on a satellite image.



(a)



(b)

Figure 6.19: Resulting 3D models after processing all 14 trajectories of the mixed indoor/outdoor images contained in the *CDBench* dataset. The estimated trajectories are marked in blue.

presented a new benchmark dataset called *CDBench*, where through a long recording time (14 runs at 10 different days over more than two weeks) as much scene dynamics as possible have been captured. Additionally, groundtruth by means of manually measured distances between predefined landmarks has been provided.

Throughout our experiments we focused on measuring scalability, localization accuracy, relocalization ability and mapping performance of our visual SLAM algorithm. Herby, we confronted our standard SLAM framework (compare Section 4.6) with a solution using the HOC descriptor as well as the pose graph described in Chapter 5:

The systems scalability has been measured in many different ways: Stability of the map size over time, matching effort during data association, computation time and memory consumption. Simply incorporating all sensor information would continously enlarge the map in every run, especially during life-long operation within a changing environment. Looking at Figure 6.14(a) and Tables 6.6, 6.7 and 6.10 we are able to stabilize the mapsize in synthetic, small-scale and large-scale experiments throughout mulitple runs. This also drastically reduces memory consumption which is a desirable effect in the context of continuous localization and mapping. Although the amount of saved data per feature is larger than in standard SLAM algorithms, we finally save memory by keeping the overall map size small.

During tracking we greatly benefit from the visibility information encoded in the HOC descriptor, where we reduced the average matching effort from 42.92% to 91.18% per frame as presented in Table 6.9. This is also visualized in Figure 6.12, when comparing the number of visible map points per frame with and without visibility information, particularly at the three black ridges. The reduction in possible matching candidates further boosts matching performance, matching effort and consequently the accuracy of the subsequent pose tracking.

An advantage regarding computing time comes along with the proposed loop closing strategy. Most of the time only small inconsistent parts have to be added to a large self-contained map. Instead of performing structure and motion bundling over the whole map only a small portion is considered for refinements, which speeds up the whole loop closing procedure. While being able to process 10 frames per second when doing tracking and sliding window bundle adjustment, loop closing (graph pose optimization followed by structure bundling) requires about 5 seconds. However, this could also be implemented parallel to the subsequent tracking procedure.

Localization accuracy has been measured in synthetic and real-world experiments by comparing the estimated poses to groundtruth data or by passing predefined, manually measured landmarks multiple times. The static camera-moving scene experiments have shown that localization is more robust with our method (compare Table 6.7 and Figure 6.9). Especially in high dynamic scenes the standard map becomes filled up with ambiguous data and correct localization may fail.

As demonstrated in the long-term experiment we successfully (re)localized while exploring perpetually. For accuracy evaluation pose estimates when passing predefined landmarks have been stored. Here, the mean deviation from the centroid of all pose estimates and the distance between those act as a measure of localization accuracy as reported in Tables 6.12 and 6.13. Unfortunately, high lighting variations lead to wrong data associations during tracking and caused the algorithm to fail in the last run, as seen by the high offset in Table 6.12 at landmarks #10 and # 13. As a consequence a parallel map is established while the existing one is downweighted by the HOC descriptor. In order to deal with radiometric dynamics a more sophisticated update rule has to be developed.

Since providing groundtruth data in an ever changing world becomes infeasible, mapping performance has just been evaluated in the synthetic experiments. When dealing with dynamic scene elements we clearly outperform our standard SLAM procedure as presented in Table 6.6. The novel feature descriptor correctly adapts an existing map by downweighting vanished or moving landmarks and creating new map points upon new measurements. One problem might arise when dealing with periodic occurrences like opened or closed doors, temporary occlusions or recurrent lighting conditions which may be filtered out by the HOC descriptor and added again to the map on reappearance. A simple approach is to change the parameter p in Equation 5.9, but in general more sophisticated strategies are needed. Moreover, due to the aliasing effect caused by small deviations of the camera pose estimate, already vanished landmarks are not downweighted properly (compare Figure 6.10). Here, a weighted update procedure taking neighboring bins into account would solve the aliasing effect.

7 Conclusion and Outlook

7.1 Conclusion

In this thesis the problem of simultaneous localization and mapping of autonomous mobile robots using vision sensors has been addressed, with special focus on life-long operation in a dynamic surrounding. Popular SLAM algorithms assume a static environment or simply do not react on environmental changes. Operating in large-scale environments over a long period of time becomes intractable since map sizes increase steadily and dynamic scene elements may cause wrong pose estimates.

We presented a keyframe-based SLAM framework taking both monocular or stereo data as input. Keyframes are organized in an undirected, unweighted view-graph, which facilitates sliding window bundle adjustment, keyframe insertion and map expansion. To speed up loop closure correction we employ the graph structure to determine a subset of keyframes to be corrected for rotation, translation and scale. To deal with dynamic scene elements we attach the Histogram of Oriented Cameras descriptor to each map point which encodes, given a camera view, visibility information as well as its importance for localization. We presented two different descriptor shapes - octahedron and icosahedron - together with specific update algorithms and importance measures. The proposed algorithm is able to operate over long-periods of time indoors and outdoors and is able to implicitly handle dynamic scene elements during map building.

An extensive review of available visual SLAM solutions showed that they are restricted to operate in static environments only, where map construction and loop closure correction is performed only once. Dynamic scene elements are either ignored, filtered out or not taken into account leading to wrong pose estimates and a falsified map representation. The solutions presented here are able to deal with a variety of different scene dynamics. More specifically, we are able to implicitly update, repair and expand a given map upon a change in the surroundings. The consequence of this are more accurate pose estimates, an up-to-date map and reduced memory effort.

Finally, we addressed the lack of a benchmark dataset aiming at long-term mapping within an ever changing world. Therefore, a benchmark dataset consisting of 14 independent runs made of noisy (e.g. leaves moved by the wind), short-term (e.g. pedestrians), radiometric (e.g. different lighting conditions) and long-term (e.g. parking cars) dynamics has been recorded and published, which pose a great challenge for every visual SLAM system. As demonstrated in our experiments we successfully (re)localized while exploring perpetually. Especially after a long time of operation, in a standard SLAM approach the map becomes filled up with ambiguous data, and correct localization may fail. Contrarily, we are able to react on environmental changes and improve localization accuracy implicitly. During pose tracking we greatly benefit from the visibility information encoded in the HOC descriptor, where we reduced the average matching effort, which also affects computation time per frame. Finally, we saved memory by keeping the overall map size small.

7.2 Outlook

In this thesis, we presented two powerful methods allowing a SLAM algorithm to operate in a dynamic world over a long time interval. Both have been evaluated with a novel benchmark dataset. However, there are still open questions and drawbacks, which may be addressed in future work:

Evaluation

In this thesis we mainly employed our *CDBench* dataset to perform detailed evaluation of our SLAM framework. Here, we only compared our method to a common state-of-the-art algorithm, as there is hardly any solution available, which is able to deal with dynamic surrounding using vision sensors only. However, in future work a detailed comparison of various visual SLAM methods in the *CDBench* dataset needs to be done.

In order to provide some useful measures for the mapping performance dynamic scene elements should be annotated over the whole dataset and tagged according to the dynamic categorization provided in Section 5.1.1. This would further allow to evaluate the algorithms robustness against different scene dynamics.

Additionally, the effects of the parameters of the HOC descriptors and their weighting schemes has not been investigated. Future work should include an extensive parameter study and their effect on handling dynamic environments should be investigated. Moreover, the effect of using different HOC shapes should be evaluated.

Finally, recording further trajectories should keep going on - preferably throughout several seasons. This would allow us to investigate the effect of seasonal changes (e.g. summer or winter) on localization performance and, even more important, on map construction and adaption when using the HOC descriptor.

Incorporation of Additional Information

Currently, we only used the negative information from the data association process to update the HOC descriptors weight and to react on environmental changes. The incorporation of semantic information (e.g. timestamps, objects detection) could further boost the algorithms performance. In our work the different time scales each dynamic object is related to have not been investigated. For example, landmarks related to a pedestrian behave different than landmarks belonging to an opened/closed door. Expanding the HOC descriptor with the ability to implicitly handle such different behavior could improve the handling of periodic events.

Moreover, it is not clear how additional information like date, time or some higher level semantic information (e.g. person, car, door, building) can be used to improve the ability of an algorithm to deal with scene dynamics.

Computational Effort

In this work we did not focus on runtime improvements of the algorithms used in the different buildings blocks of our framework. Since our algorithms have been implemented as MATLAB prototypes more engineering effort is necessary to achieve real-time performance. Here, graph based optimization presented by Kümmerle et al. [68] would boost sliding window bundle adjustment during both map building and loop closure correction.

Radiometric Dynamics

As shown in the experiments radiometric dynamics (e.g. abrupt lighting changes) cause the algorithm to fail, since all landmarks simultaneously change their appearance. In our case, a further loop closure detection may have corrected wrong pose estimates caused by lighting variations. However, radiometric dynamics need to be investigated more extensively.

Loop Closing Strategy

The presented loop closing strategy aims at accelerating the whole loop closing process by attaching a small inconsistent map part to a large consistent map. Contrary, it does not take the quality of the previously mapped environment into account assuming that perpetual sliding window bundle adjustment ensures a consistent, accurate map. Further investigations are necessary to identify the less accurate part of a map by defining a reasonable quality measure. Consequently, the less accurate part should be aligned to the qualitative higher map.

A Non-Linear Least-Squares Parameter Estimation Techniques

A.1 Introduction

Many computer vision problems like camera calibration or bundle adjustment try to estimate a set of parameters by minimizing some kind of image space error through an iterative least-square solver. In case of camera calibration a set of image points (checker board corners) with known world coordinates is imaged by the camera. The central perspective projection function (see Section 3.1.5) defines the relation between the world coordinates and the extracted image points. The aim of the parameter estimation procedure is to identify the unknowns of the central perspective projection function, namely the intrinsic and extrinsic parameters by minimizing the euclidean distance between projected world points and extracted image points.

This Chapter deals with the most popular iterative non-linear least-square solver, namely Newton, Gauss-Newton and Gradient-Descent. The Levenber-Marquardt algorithm which is a combination of Gauss-Newton and Gradient-Descent is reviewed in Section A.3.

A.2 Iterative Parameter Estimation Methods

Suppose you are given a measurement vector $M \in \mathbb{R}^N$, a parameter vector $P \in \mathbb{R}^M$ and a non-linear function $f : \mathbb{R}^M \to \mathbb{R}^N$ constraining the parameters P through $f(P) = M$. In most practical cases measurements M are subject to noise and only an estimate $\hat{P}$ can be computed being as close as possible to M. Hence, we try to seek $\hat{P}$ minimizing some error metric in the measurement space:

$$\begin{aligned} &\arg\min_P \|\varepsilon(P)\| \\ \varepsilon(P) &= \|f(\hat{P}) - M\| \end{aligned} \tag{A.1}$$

Starting with an initial guess P_0 close to the minimum we try to iteratively refine the estimate assuming that f is locally linear

$$f(P_0 + \Delta) = f(P_0) + J\,\Delta \tag{A.2}$$

$$J = \frac{\partial f}{\partial P}, \tag{A.3}$$

where J is the Jacobian with respect to the parameters P. Within the next iteration we seek for a $P_1 = P_0 + \Delta$, which minimizes

$$\varepsilon_1 = f(P_1) - M = f(P_0) + J\,\Delta - M = \varepsilon_0 + J\,\Delta,$$

which is the same as minimizing the linear minimization problem $\|\varepsilon_0 + J\,\Delta\|$ over Δ. The shift vector Δ is found by solving the normal equation

$$(J^T J)\,\Delta = -J^T\,\varepsilon_0 \tag{A.4}$$

or by computing the pseudo inverse J^+

$$\Delta = -J^+\,\varepsilon_0. \tag{A.5}$$

This results in an iterative procedure summarized as:

$$P_{i+1} = P_i + \Delta_i, \tag{A.6}$$

where Δ_i is the solution to

$$\begin{aligned} &\|\varepsilon_i + J\,\Delta_i\| \quad \text{with} \\ &J = \frac{\partial f}{\partial P_i} \\ &\varepsilon_i = f(P_i) - M. \end{aligned} \tag{A.7}$$

A.2.1 Newton-Iteration

To be general, Newton-Iteration tries to find the minimum of an arbitrary scalar-valued function $g(P)$ provided an initial guess P_0 close to the optimum. A minimum with respect to Δ is found by expanding the function $g(P)$ around P_0 by a Taylor series and setting the first derivative to zero:

$$g(P_0 + \Delta) = g(P_0) + g'(P_0)\Delta + \Delta\, g''(P_0)\,\Delta/2 + ... \tag{A.8}$$

$$\frac{\partial g(P_0 + \Delta)}{\partial \Delta} = g'(P_0) + g''(P_0)\,\Delta = 0. \tag{A.9}$$

Hereby, g' and g'' denote the gradient and Hessian, respectively.

Turning to the specific squared error norm, $g(P)$ is defined as

$$g(P) = \frac{1}{2}\|\varepsilon(P)\|^2 \tag{A.10}$$

$$= \frac{1}{2}\varepsilon(P)^T\varepsilon(P), \tag{A.11}$$

where $\varepsilon(P) = f(P) - M$. Consequently $g' = \varepsilon'^T\varepsilon = f'^T\,\varepsilon = J^T\varepsilon(P)$ and the Hessian $g'' = \varepsilon'^T\varepsilon + \varepsilon''^T\varepsilon = J^TJ + \varepsilon''^T\varepsilon$.

A.2.2 Gauss-Newton Method

Assuming that $f(P)$ is linear then the second derivative within the Hessian vanishes and $g'' = J^TJ$. Substituting into Equation A.9 results in

$$\frac{\partial g(P_0 + \Delta)}{\partial \Delta} = J^T\varepsilon(P) + J^TJ\,\Delta = 0 \tag{A.12}$$

$$= J^TJ\,\Delta = -J^T\varepsilon(P), \tag{A.13}$$

which is the same as the normal equations A.4. Approximating the Hessian through $J^T J$ defines the Gauss-Newton Method, which gives reasonable results close to the actual minimum.

A.2.3 Gradient-Descent

To reach the minimum the gradient descent method follows the negative gradient, which defines the direction of the most rapid decrease of the cost function:

$$\Delta = -\lambda\, g' = -\lambda J^T \varepsilon(P), \tag{A.14}$$

where the length of the step is controlled by λ. Although having slower convergence than the Gauss-Newton Method it may be the preferred method when starting with an initial estimate far away from the optimum.

A.3 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm combines the advantages of Gradient-Descent and Gauss-Newton where the normal equations to be solved

$$(J^T J)\Delta = -J^T \varepsilon$$

are replaced by

$$(J^T J + \lambda I)\Delta = -J^T \varepsilon. \tag{A.15}$$

One Levenberg-Marquardt iteration is defined by solving repeatedly Equation A.15 for different values of λ until an acceptable Δ is found.

If λ is very small the method favors a Gauss-Newton iteration, which converges fast to the minimum near the optimal solution. If λ is large then the method favors a Gradient-Descent iteration guaranteeing a decrease in the cost function when being far away from the minimum.

Unfortunately, the Levenberg-Marquardt algorithm is only suited for a small amount of parameters, because solving the normal equations has complexity M^3. Therefore, partitioning the parameter space and exploiting the sparse structure of the resulting Jacobian leads to great time savings especially when trying to estimate a huge amount of parameters.

A.3.1 Partitioned Levenberg-Marquardt

Partitioning the parameter vector $P = (a, b)$ the Jacobian can be written as $J = \frac{\partial f}{\partial P} = [\frac{\partial f}{\partial a} | \frac{\partial f}{\partial b}] = [A|B]$, which is also known as the Schur complement trick. Then the normal equation A.15 can be written as

$$\begin{pmatrix} A^T A & A^T B \\ B^T A & B^T B \end{pmatrix} \begin{bmatrix} \Delta_a \\ \Delta_b \end{bmatrix} + \lambda I = \begin{pmatrix} A^T \varepsilon \\ B^T \varepsilon \end{pmatrix}. \tag{A.16}$$

After adding the damping factor λ to the diagonal entries of $A^T A$ and $B^T B$ results in $\lambda(A^T A) = (A^T A)^* = U^*$ and $lambda(B^T B) = (B^T B)^* = V^*$. Hence, Equation A.16 is rewritten as

$$\begin{pmatrix} U^* & W \\ W^T & V^* \end{pmatrix} \begin{bmatrix} \Delta_a \\ \Delta_b \end{bmatrix} = \begin{pmatrix} \varepsilon_A \\ \varepsilon_B \end{pmatrix}, \tag{A.17}$$

where $W = A^T B$, $\varepsilon_A = A^T \varepsilon$ and $\varepsilon_B = B^T \varepsilon$. Multiplying both sides with

$$\begin{bmatrix} I & -WV^{*-1} \\ 0 & I \end{bmatrix}$$

results in a system of equations where the top right block vanishes:

$$\begin{bmatrix} U^* - WV^{*-1}W^T & 0 \\ W^T & V^* \end{bmatrix} \begin{bmatrix} \Delta_a \\ \Delta_b \end{bmatrix} = \begin{bmatrix} \varepsilon_A - WV^{*-1}\varepsilon_B \\ \varepsilon_B \end{bmatrix}. \tag{A.18}$$

Finally, one has to solve for Δ_a by

$$\left(U^* - WV^{*-1}W^T\right) \Delta_a = \varepsilon_A - WV^{*-1}\varepsilon_B$$

and back-substitute to get Δ_b

$$W^T \Delta_a + V^* \Delta_b = \varepsilon_B$$

.

A.3.2 Sparse Levenberg-Marquardt

A further break up of the measurements $M = (M_1^T, M_2^T, ..., M_n^T)^T$ and parameters $P = (a^T, b_1^T, ..., b_n^T)^T$ and the assumption that M_i only depends on a and b_j results in

$$\frac{\partial M_i}{\partial b_j} = 0 \;\; i \neq j.$$

This results in a sparse block-structure of the Jacobian J

$$J = \begin{pmatrix} A_1 & B_1 & & & \\ A_2 & & B_2 & & \\ \vdots & & & \ddots & \\ A_n & & & & B_n \end{pmatrix}. \tag{A.19}$$

As a consequence each step of the algorithm only requires a computation time linear in M instead of M^3.

B List of Publications

B.1 Thesis Relevant Publications

Histogram of Oriented Cameras - a New Descriptor for Visual SLAM in Dynamic Environments

Katrin Pirker, Matthias Rüther, Horst Bischof
In: *Proceedings of British Machine Vision Conference* (BMVC),
August 2010, Aberystwyth, Great Britain
(Accepted for poster presentation, 34 % acceptance rate)

Abstract: Simultaneous localization and mapping (SLAM) is a basic prerequisite in autonomous mobile robotics. Most existing visual SLAM approaches either assume a static environment, or simply forget old parts of the map to cope with map size constraints and scene dynamics. We present a novel map representation for sparse visual features. A new 3D point descriptor called Histogram of Oriented Cameras (HOC) encodes anisotropic spatial visibility information and the importance of each three-dimensional landmark. Each feature holds and updates a histogram of the poses of observing cameras. It is hereby able to estimate its probability of occlusion and importance for localization from a given viewpoint. In a series of simulated and real-world experiments we prove that the pro- posed descriptor allows to cope with dynamic changes in the map, improves localization accuracy and enables reasonable control of the map size.

Large-Scale Robotic SLAM through Visual Mapping

Christof Hoppe, Katrin Pirker, Matthias Rüther and Horst Bischof
In: *Proc. 35rd Workshop of the Austrian Association for Pattern Recognition* (AAPR/OAGM), May 2011, Graz, Austria
(Accepted for oral presentation)

Abstract: Keyframe-based visual SLAM systems perform reliably and fast in medium-sized environments. Currently, their main weaknesses are robustness and scalability in large scenarios. In this work, we propose a hybrid, keyframe based visual SLAM system, which overcomes these problems. We combine visual features of different strength, add appearance-based loop detection and present a novel method to incorporate non-visual sensor information into standard bundle adjustment frameworks to tackle the problem of weakly textured scenes. On a standardized test dataset, we outperform EKF-based solutions in terms of localization accuracy by at least a factor of two. On a self-recorded dataset, we achieve a performance comparable to a laser scanner approach.

CD SLAM - Continuous Localization and Mapping in a Dynamic World

Katrin Pirker, Matthias Rüther and Horst Bischof
In: *IEEE/RSJ Proceedings of Conference on Intelligent Robots and Systems* (IROS), September 2011, San Francisco, USA
(Accepted for poster presentation, 32 % acceptance rate)

Abstract: When performing large-scale perpetual localization and mapping one faces problems like memory consumption or repetitive and dynamic scene elements requiring robust data association. We propose a visual SLAM method which handles short- and long-term scene dynamics in large environments using a single camera only. Through visibility-dependent map filtering and efficient keyframe organization we reach a considerable performance gain only through incorporation of a slightly more complex map representation. Experiments on a large, mixed indoor/outdoor dataset over a time period of two weeks demonstrate the scalability and robustness of our approach.

B.2 Other Publications

Human Body Volume Estimation in a Clinical Environment

Katrin Pirker, Matthias Rüther, Horst Bischof, Falko Skrabal and Georg Pichler
In: *Proc. 33rd Workshop of the Austrian Association for Pattern Recognition* (AAPR/OAGM), 2009, Stainz, Austria
(Accepted for oral presentation, 58.97 % acceptance rate)

Abstract: Medical scientists try to improve the accuracy of human body composition measurement by segmental Bioelectrical Impedance Analysis (BIA). Determination of body composition includes estimation of total body water, fat-free mass and extra- and intra-cellular water, which are of great importance in disease states like cancer, malnutrition, disturbances of hydration as seen in heart-, hepatic- and renal failure. Currently, body composition is derived by measuring the specific resistivity of the total body or of its segments and heuristically determining their volumes. Whereas the resistivity measurement is very accurate, the main uncertainty factor remains the volume estimation, which is performed through empirical models. To improve the accuracy of volume estimation a vision based measurement system is proposed. A portable system consisting of stereo cameras and projectors has been constructed, which can be used in a clinical environment even for bedridden, disabled and intensive care patients. In this work the main focus lies on realistic human body modeling and volume calculation from noisy three-dimensional data. The accuracy of the volume measurement has been compared to groundtruth data of two volunteers.

An Omnidirectional Time-of-Flight Camera and its Application to Indoor SLAM

Katrin Pirker, Matthias Rüther, Gerald Schweighofer, Horst Bischof and Heinz Mayer
In: *IEEE Proceedings of 11th International Conference on Control, Automation, Robotics and Vision* (ICARCV),
December 2010, Singapore
(Accepted for oral presentation)

Abstract: Photonic mixer devices (PMDs) are able to create reliable depth maps of indoor environments. Yet, their application in mobile robotics, especially in simultaneous localization and mapping (SLAM) applications, is hampered by the limited field of view. Enhancing the field of view by optical devices is not trivial, because the active

light source and the sensor rays need to be redirected in a defined manner. In this work we propose an omnidirectional PMD sensor which is well suited for indoor SLAM and easy to calibrate. Using a single sensor and multiple planar mirrors, we are able to reliably navigate in indoor environments to create geometrically consistent maps, even on optically difficult surfaces.

GPSlam: Marrying Sparse Geometric and Dense Probabilistic Visual Mapping

Katrin Pirker, Gerald Schweighofer, Matthias Rüther and Horst Bischof
In: *Proceedings of British Machine Vision Conference* (BMVC),
August 2011, Dundee, Great Britain
(Accepted for poster presentation, 32 % acceptance rate)

Abstract: We propose a novel, hybrid SLAM system to construct a dense occupancy grid map based on sparse visual features and dense depth information. While previous approaches deemed the occupancy grid usable only in 2D mapping, and in combination with a probabilistic approach, we show that geometric SLAM can produce consistent, robust and dense occupancy information, and maintain it even during erroneous exploration and loop closure. We require only a single hypothesis of the occupancy map and employ a weighted inverse mapping scheme to align it to sparse geometric information. We propose a novel map-update criterion to prevent inconsistencies, and a robust measure to discriminate exploration from localization.

Fast and Accurate Environment Modeling using Three-Dimensional Occupancy Grids

Katrin Pirker, Gerald Schweighofer, Matthias Rüther and Horst Bischof
In: *Proceedings of 1st IEEE Workshop on Consumer Depth Camera for Computer Vision* (ICCV,CDC4CV),
November 2011, Barcelona, Spain
(Accepted for poster presentation, 32 % acceptance rate)

Abstract: Building a dense and accurate environment model out of range image data faces problems like sensor noise, extensive memory consumption or computation time. We present an approach which reconstructs 3D environments using a probabilistic occupancy grid in real-time. Operating on depth image pyramids speeds up computation time, whereas a weighted interpolation scheme between neighboring pyramid layers boosts accuracy. In our experiments we compare our method with a state-of-the-art mapping procedure. Our results demonstrate that we achieve better results. Finally, we present its viability by mapping a large indoor environment.

Photogrammetric Camera Network Design for Micro Aerial Vehicles

Christof Hoppe, Andreas Wendel, Stefanie Zollmann, Katrin Pirker, Arnold Irschara, Horst Bischof, Stefan Kluckner

In: *Proceedings of Computer Vison Winterworkshop* (CVVW),
February 2012, Mala Nedelja, Slovenia
(Accepted for poster presentation)

Abstract: Micro Aerial Vehicles (MAVs) equipped with high resolution cameras have the ability of cost efficient and autonomous image acquisition from unconventional viewpoints. To fully exploit the limited flight-time of current MAVs view planning is essential for complete and precise 3D scene sampling. We propose a novel camera network design algorithm suitable for MAVs for close range photogrammetry that exploits prior knowledge of the surrounding. Our algorithm automatically determines a set of camera positions that guarantees important constraints for image based 3D reconstruction. On synthetic experiments we demonstrate that our camera network design obtains detailed 3D reconstructions with a reduced number of images at the desired accuracy level. Comparable results are also computed on an outdoor experiment using our MAV in autonomous flight mode.

Bibliography

[1] M. Altermatt, A. Martinelli, N. Tomatis, and R. Siegwart, *SLAM with corner features based on a relative map*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, 2004, pp. 1053 – 1058. (cited on page 59)

[2] J. Andrade-Cetto and A. Sanfeliu, *Concurrent map building and localization on indoor dynamic environments*, International Journal on Pattern Recogniton, 16 (2002). (cited on pages 16 and 19)

[3] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, *Visual topological SLAM and global localization*, in Proceedings of the IEEE International Conference on Robotics and Automation, 2009, pp. 2029–2034. (cited on page 62)

[4] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun, *Learning hierarchical object maps of non-stationary environments with mobile robots*, in Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2002. (cited on pages 7, 16, and 19)

[5] D. Anguelov, D. Koller, E. Parker, and S. Thrun, *Detecting and modeling doors with mobile robots*, in Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, 2004, pp. 3777 – 3784. (cited on page 16)

[6] D. Arbuckle, A. Howard, and M. Mataric, *Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, 2002, pp. 409 – 414. (cited on pages 17 and 19)

[7] Q. Baig, T.-D. Vu, and O. Aycard, *Online localization and mapping with moving objects detection in dynamic outdoor environments*, in IEEE International Conference on Intelligent Computer Communication and Processin, 2009, pp. 401–408. (cited on pages 16 and 19)

[8] P. Biber and T. Duckett, *Dynamic maps for long-term operation of mobile service robots*, in Proceedings of Robotics: Science and Systems, 2005. (cited on pages 17 and 19)

[9] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardos, *RAWSEEDS: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets*, in Proceedings of IROS'06 Workshop on Benchmarks in Robotics Research, 2006. (cited on page 121)

[10] M. Bosse, P. Newman, J. Leonard, and S. Teller, *Simultaneous localization and map building in large-scale cyclic environments using the atlas framework*, The International Journal of Robotics Research, 23 (2004), pp. 1113–1139. (cited on page 120)

[11] J. Y. Bouguet, *Camera calibration toolbox for Matlab*, 2008. (cited on pages 33 and 124)

[12] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, *Estimating the absolute position of a mobile robot using position probability grids*, in Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996, pp. 896–901. (cited on page 69)

[13] A. Cassandra, L. Kaelbling, and J. Kurien, *Acting under uncertainty: Discrete bayesian models for mobile-robot navigation*, in Proceedings of the IEEE/RSJ International Conference on Intelligend Robots and Systems, 1996, pp. 963–972. (cited on page 69)

[14] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, *Rawseeds ground truth collection systems for indoor self-localization and mapping*, Autonomous Robots, 27 (2009), pp. 353–371. (cited on page 121)

[15] S. Ceriani, D. Marzorati, M. Matteucci, D. Migliore, and D. G. Sorrenti, *On feature parameterization for EKF-based monocular SLAM*, in World Congress of the International Federation of Automatic Control, 2011, pp. 6829–6834. (cited on page 52)

[16] M. Chli and A. J. Davison, *Automatically and efficiently inferring the hierarchical structure of visual maps*, in International Conference on Robotics and Automation, 2009, pp. 2211–2218. (cited on page 12)

[17] J. Civera, A. Davison, and J. Montiel, *Inverse depth parametrization for monocular slam*, IEEE Transactions on Robotics, 24 (2008), pp. 932–945. (cited on pages 12 and 54)

[18] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardós, *Mapping large loops with a single hand-held camera*, in Proceedings of Robotics: Science and Systems, 2007. (cited on page 12)

[19] B. Clipp, J. Lim, J.-M. Frahm, and M. Pollefeys, *Parallel, real-time visual slam*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 3961–3968. (cited on pages 13 and 111)

[20] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, *Discrete-continuous optimization for large-scale structure from motion*, in IEEE Conference on Computer Vision and Pattern Recognition, june 2011, pp. 3001 –3008. (cited on page 39)

[21] A. Criminisi, I. Reid, and A. Zisserman, *Single view metrology*, International Journal of Computer Vision, 40 (2000), pp. 123–148. (cited on page 39)

[22] M. Cummins and P. Newman, *FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance*, The International Journal of Robotics Research, 27 (2008), pp. 647–665. (cited on page 76)

[23] ——, *Highly scalable appearance-only SLAM - FAB-MAP 2.0*, in Robotics Science and Systems, Seattle, USA, June 2009. (cited on pages 62, 76, and 83)

[24] M. Daniele, M. Matteo, M. Davide, R. Roberto, and S. D. Giorgio, *Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments*, in ICRA09 Workshop on Safe navigation in open and dynamic environments - Application to autonomous vehicles, 2009. (cited on pages 15 and 19)

[25] A. Davison, *Real-time simultaneous localisation and mapping with a single camera*, in IEEE International Conference on Computer Vision, oct. 2003, pp. 1403 –1410 vol.2. (cited on pages 3, 12, 49, and 60)

[26] A. Davison and D. Murray, *Simultaneous localization and map-building using active vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 865 –880. (cited on page 12)

[27] A. Davison, I. Reid, N. Molton, and O. Stasse, *MonoSLAM: Real-time single camera slam*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1052 –1067. (cited on pages 12 and 15)

[28] F. Dayoub and T. Duckett, *An adaptive appearance-based map for long-term topological localization of mobile robots*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, sept. 2008, pp. 3364 –3369. (cited on pages 17 and 19)

[29] F. Dellaert and M. Kaess, *Square Root SAM: Simultaneous localization and mapping via square root information smoothing*, International Journals of Robotic Research, 25 (2006), pp. 1181–1203. (cited on page 12)

[30] G. Dissanayake, P. M. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, *A solution to the simultaneous localization and map building (slam) problem*, IEEE Transactions on Robotics and Automation, 17 (2001), pp. 229–241. (cited on page 11)

[31] E. Eade and T. Drummond, *Scalable monocular SLAM*, in IEEE International Conference on Computer Vision, 2006, pp. 469–476. (cited on page 12)

[32] A. I. Eliazar and R. Parr, *DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks*, in International Joint Conference on Artificial Intelligence, 2003, pp. 1135–1142. (cited on page 48)

[33] P. Elinas, R. Sim, and J. Little, *σ SLAM: stereo vision SLAM using the rao-blackwellised particle filter and a novel mixture proposal distribution*, in IEEE International Conference on Robotics and Automation, 2006, pp. 1564 –1570. (cited on page 12)

[34] O. Faugeras and Q.-T. Luong, *The Geometry of Multiple Images*, MIT Press, 2001. ISBN: 0-262-06220-8. (cited on page 22)

[35] E. Fink and K. B. Pratt, *Indexing of compressed time series*, in Data Mining in Time Series Databases, World Scientific, pp. 51–78. (cited on page 17)

[36] M. A. Fischler and R. C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Commun. ACM, 24 (1981), pp. 381–395. (cited on pages 71, 82, and 115)

[37] A. W. Fitzgibbon and A. Zisserman, *Automatic camera recovery for closed or open image sequences*, in Proceedings of the European Conference on Computer Vision, 1998, pp. 311–326. (cited on page 39)

[38] J. Folkesson and H. Christensen, *Graphical SLAM - a self-correcting map*, in IEEE International Conference on Robotics and Automation, 2004, pp. 383 – 390. (cited on page 120)

[39] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, *Monte Carlo Localization: Efficient position estimation for mobile robots*, in Proceedings of the International Conference on Artificial Intelligence, 1999, pp. 343–349. (cited on page 5)

[40] D. Fox, W. Burgard, and S. Thrun, *Markov localization for mobile robots in dynamic environments*, Journal of Artificial Intelligence Research, 11 (1999), pp. 391–427. (cited on page 15)

[41] F. Fraundorfer, C. Engels, and D. Nister, *Topological mapping, localization and navigation using image collections*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 3872 –3877. (cited on page 62)

[42] R. Gherardi, M. Farenzena, and A. Fusiello, *Improving the efficiency of hierarchical structure-and-motion*, in IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1594 –1600. (cited on page 39)

[43] G. Graber, T. Pock, and H. Bischof, *Online 3d reconstruction using convex optimization*, in 1st Workshop on Live Dense Reconstruction From Moving Cameras (ICCV), 2011. (cited on page 14)

[44] G. Grisetti, C. Stachniss, and W. Burgard, *Improved techniques for grid mapping with rao-blackwellized particle filters*, IEEE Transactions on Robotics, 23 (2007), p. 2007. (cited on pages 48 and 120)

[45] G. Grisetti, C. Stachniss, and W. Burgard, *Nonlinear constraint network optimization for efficient map learning*, Transactions on Intelligent Transport Systems, 10 (2009), pp. 428–439. (cited on page 13)

[46] D. R. H. Durrant-Whyte and E. Nebot, *Localisation of automatic guided vehicles*, in The 7th International Symposium in Robotics Research, 1995. (cited on page 11)

[47] D. Hähnel, W. Burgard, D. Fox, and S. Thrun, *A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (cited on pages 48, 49, and 120)

[48] D. Hähnel, D. Schulz, and W. Burgard, *Map building with mobile robots in populated environments*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, 2002, pp. 496 – 501 vol.1. (cited on pages 15 and 19)

[49] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, *Map building with mobile robots in dynamic environments*, in IEEE International Conference on Robotics and Automation, 2003. (cited on page 15)

[50] R. Haralick, D. Lee, K. Ottenburg, and M. Nolle, *Analysis and solutions of the three point perspective pose estimation problem*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 592 –598. (cited on pages 68 and 82)

[51] C. Harris and M. Stephens, *A combined corner and edge detector*, in Fourth Alvey Vision Conference, 1998, pp. 147–151. (cited on page 51)

[52] R. I. Hartley, *In defense of the eight-point algorithm*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (1997), pp. 580–593. (cited on page 39)

[53] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second ed., 2004. (cited on pages 22, 26, 33, 35, 38, 39, and 42)

[54] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, *RGBD mapping: Using depth cameras for dense 3d modeling of indoor environments*, in In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS, 2010. (cited on pages 13 and 14)

[55] S. Hochdorfer and C. Schlegel, *Towards a robust visual slam approach: Addressing the challenge of life-long operation*, in International Conference on Advanced Robotics, june 2009, pp. 1 –6. (cited on page 16)

[56] S. Holmes, G. Sibley, G. Klein, and D. Murray, *A relative frame representation for fixed-time bundle adjustment in SFM*, in IEEE International Conference on Robotics and Automation, may 2009, pp. 2264 –2269. (cited on page 13)

[57] A. Howard and N. Roy, *The robotics data set repository (radish)*, 2003. (cited on page 120)

[58] G. Huang, A. Rad, and Y. Wong, *A new solution to map dynamic indoor environments*, International Journal of Advanced Robotic Systems, (2008). (cited on page 16)

[59] P. Huber, *Robust Statistics*, Wiley, New York, 1974. (cited on page 45)

[60] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, *KinectFusion: real-time 3d reconstruction and interaction using a moving depth camera*, in Proceedings of the 24th annual ACM Symposium on User interface software and technology, 2011, pp. 559–568. (cited on pages 14 and 55)

[61] P. Jensfelt and H. Christensen, *Pose tracking using laser scanning and minimalistic environmental models*, IEEE Transactions on Robotics and Automation, 17 (2001), pp. 138 –147. (cited on page 72)

[62] S. Julier and J. Uhlmann, *A new extension of the kalman filter to nonlinear systems*, in Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defense Sensing, Simulations and Controls, 1997. (cited on page 69)

[63] G. Klein and D. Murray, *Parallel tracking and mapping for small ar workspaces*, in IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 225 –234. (cited on pages 5, 12, 49, 59, 60, 71, 80, 83, and 114)

[64] M. Klopschitz, A. Irschara, G. Reitmayr, and D. Schmalstieg, *Robust incremental structure from motion*, in 3DPVT10, 2010. (cited on page 39)

[65] K. Konolige and M. Agrawal, *FrameSLAM: From bundle adjustment to real-time visual mapping*, IEEE Transactions on Robotics, 24 (2008), pp. 1066 –1077. (cited on pages 13, 18, and 49)

[66] K. Konolige and J. Bowman, *Towards lifelong visual maps*, in Proceedings of the IEEE/RSJ international Conference on Intelligent Robots and Systems, 2009, pp. 1156–1163. (cited on pages 17, 18, 19, and 111)

[67] B. Kuipers and Y.-T. Byun, *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*, Journal of Robotics and Autonomous Systems, 8 (1991), pp. 47–63. (cited on page 11)

[68] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, *g2o: A general framework for graph optimization*, in Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Shanghai, China, May 2011. (cited on page 153)

[69] P. Lamon, I. Nourbakhsh, B. Jensen, and R. Siegwart, *Deriving and matching image fingerprint sequences for mobile robot localization*, in IEEE International Conference on Robotics and Automation, vol. 2, 2001, pp. 1609 – 1614 vol.2. (cited on page 56)

[70] J. J. Leonard and D. H. Whyte, *Mobile robot localization by tracking geometric beacons*, IEEE Transactions on Robotics and Automation, 7 (1991). (cited on page 11)

[71] J. Levinson and S. Thrun, *Robust vehicle localization in urban environments using probabilistic maps*, in IEEE International Conference on Robotics and Automation, 2010, pp. 4372 –4378. (cited on pages 7, 17, and 19)

[72] G. Lidoris, D. Wollherr, and M. Buss, *Bayesian state estimation and behavior selection for autonomous robotic exploration in dynamic environments*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 1299 –1306. (cited on pages 14, 15, and 19)

[73] K.-H. Lin and C.-C. Wang, *Stereo-based simultaneous localization, mapping and moving object tracking*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 3975 –3980. (cited on pages 15 and 19)

[74] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60 (2004), pp. 91–110. (cited on pages 52, 59, 77, 80, and 101)

[75] C.-P. Lu, G. Hager, and E. Mjolsness, *Fast and globally convergent pose estimation from video images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 610 –622. (cited on page 71)

[76] F. Lu and E. Milios, *Globally consistent range scan alignment for environment mapping*, Autonomous Robots, 4 (1997), pp. 333–349. (cited on page 11)

[77] T. Marks, A. Howard, M. Bajracharya, G. Cottrell, and L. Matthies, *Gamma-SLAM: Using stereo vision and variance grid maps for slam in unstructured environments*, in IEEE International Conference on Robotics and Automation, may 2008, pp. 3717 –3724. (cited on page 12)

[78] D. W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, SIAM Journal on Applied Mathematics, 11 (1963), pp. 431–441. (cited on page 38)

[79] D. Martinec and T. Pajdla, *Robust rotation and translation estimation in multiview reconstruction*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1 –8. (cited on page 39)

[80] D. Marzorati, M. Matteucci, D. Migliore, and D. G. Sorrenti, *On the use of inverse scaling in monocular SLAM*, in IEEE International Conference on Robotics and Automation, 2009, pp. 2030 –2036. (cited on page 52)

[81] J. Matas, O. Chum, M. Urban, and T. Pajdla, *Robust wide baseline stereo from maximally stable extremal regions*, in British Machine Vision Conference, 2002. (cited on page 52)

[82] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, *A constant time efficient stereo SLAM system*, in British Machine Vision Conference, 2009. (cited on page 13)

[83] I. Miller and M. Campbell, *Rao-blackwellized particle filtering for mapping dynamic environments*, in IEEE International Conference on Robotics and Automation, 2007, pp. 3862 –3869. (cited on pages 15 and 19)

[84] N. Mitsou and C. Tzafestas, *Temporal occupancy grid for mobile robot dynamic environment mapping*, in Mediterranean Conference on Control Automation, june 2007, pp. 1 –8. (cited on page 17)

[85] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, *FastSLAM: A factored solution to the simultaneous localization and mapping problem*, in In Proceedings of the AAAI National Conference on Artificial Intelligence, 2002, pp. 593–598. (cited on pages 11 and 12)

[86] M. Montemerlo, S. Thrun, and W. Whittaker, *Conditional particle filters for simultaneous mobile robot localization and people-tracking*, in IEEE International Conference on Robotics and Automation, 2002, pp. 695 – 701. (cited on pages 14, 15, and 19)

[87] J. Montiel, J. Civera, and A. Davison, *Unified inverse depth parametrization for monocular SLAM*, in Proceedings of Robotics: Science and Systems, 2006. (cited on pages 12 and 52)

[88] H. Moravec and A. Elfes, *High resolution maps from wide angle sonar*, in IEEE International Conference on Robotics and Automation, vol. 2, 1985, pp. 116 – 121. (cited on pages 11 and 60)

[89] F. Moreno-Noguer, V. Lepetit, and P. Fua, *Accurate non-iterative O(n) solution to the pnp problem*, in IEEE International Conference on Computer Vision, 2007. (cited on pages 13 and 71)

[90] F. Mosteller and J. W. Tukey, *Data analysis and regression : a second course in statistics*, Addison-Wesley, 1977. (cited on page 46)

[91] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, *3D reconstruction of complex structures with bundle adjustment: an incremental approach*, in IEEE International Conference on Robotics and Automation, 2006, pp. 3055 –3061. (cited on page 12)

[92] R. Newcombe and A. Davison, *Live dense reconstruction with a single moving camera*, in IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1498 –1505. (cited on pages 13, 14, 55, and 59)

[93] R. A. Newcombe, S. Lovegrove, and A. J. Davison, *DTAM: Dense tracking and mapping in real-time*, in Intrenational Conference on Computer Vision, 2011. (cited on pages 13, 14, and 49)

[94] V. Nguyen, A. Harati, A. Martinelli, R. Siegwart, and N. Tomatis, *Orthogonal SLAM: a step toward lightweight indoor autonomous navigation*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, oct. 2006, pp. 5007 –5012. (cited on page 59)

[95] V. Nguyen, A. Harati, and R. Siegwart, *A lightweight SLAM algorithm using orthogonal planes for indoor mobile robotics*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 658 –663. (cited on pages 54, 55, and 59)

[96] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, *A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 1929 – 1934. (cited on page 55)

[97] D. Nister and H. Stewenius, *Scalable recognition with a vocabulary tree*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 2161 – 2168. (cited on pages 74, 75, 76, and 83)

[98] I. R. Nourbakhsh, A. Soto, J. Bobenage, S. Grange, R. Meyer, and R. Lutz, *An effective mobile robot educator with a full-time job*, Artificial Intelligence, 114 (1999), pp. 95–124. (cited on pages 62 and 63)

[99] C. F. Olson, L. Matthies, M. Schoppers, and M. W. Maimone, *Rover navigation using stereo ego-motion*, Robotics and Autonomous Systems, 43 (2003), pp. 215–229. (cited on page 68)

[100] L. Paz, P. Jensfelt, J. Tardos, and J. Neira, *EKF SLAM updates in O(n) with divide and conquer SLAM*, in IEEE International Conference on Robotics and Automation, 2007, pp. 1657 –1663. (cited on page 12)

[101] L. Paz, P. Pinies, J. Tardos, and J. Neira, *Large-scale 6-DOF SLAM with stereo-in-hand*, IEEE Transactions on Robotics, 24 (2008), pp. 946 –957. (cited on page 12)

[102] H. Pfister, M. Zwicker, J. V. Baar, and M. H. Gross, *Surfels: surface elements as rendering primitives*, in Annual Conference on Computer Graphics, 2000, pp. 335–342. (cited on pages 14 and 49)

[103] K. Pirker, M. Rüther, and H. Bischof, *Histogram of oriented cameras - a new descriptor for visual SLAM in dynamic environments*, in Proceedings of British Machine Vision Conference, 2010. (cited on page 19)

[104] ——, *CD SLAM - continuous localization and mapping in a dynamic world*, in Proc. IEEE/RSJ Conference on Intelligent Robots and Systems, 9 2011. (cited on page 19)

[105] K. Pirker, M. Rüther, G. Schweighofer, H. Bischof, and H. Mayer, *An omnidirectional time-of-flight camera and its application to indoor SLAM*, in IEEE Proceedings of International Conference on Control, Automation, Robotics and Vision, 2010. (cited on page 48)

[106] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles, *Detailed real-time urban 3D reconstruction from video*, International Journal of Computer Vision, 78 (2008), pp. 143–167. (cited on page 39)

[107] J. M. Porta, J. J. Verbeek, and B. J. A. Kröse, *Active appearance-based robot localization using stereo vision*, Auton. Robots, 18 (2005), pp. 59 – 80. (cited on page 49)

[108] V. Pratt, *Direct least-squares fitting of algebraic surfaces*, in Conference on Computer graphics and interactive techniques, 1987, pp. 145–152. (cited on page 55)

[109] E. Rosten and T. Drummond, *Machine learning for high-speed corner detection*, in European Conference on Computer Vision, 2006, pp. 430–443. (cited on page 51)

[110] P. Rybski, F. Zacharias, and J.-F. Lett, *Using visual features to build topological maps of indoor environments*, in IEEE International Conference on Robotics and Automation, vol. 1, 2003, pp. 850 – 855. (cited on page 62)

[111] D. Sabatta, *Vision-based topological map building and localisation using persistent features*, in Robotics and Mechatronics Symposium, 2008. (cited on page 62)

[112] D. Scharstein and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, International Journal of Computer Vision, 47 (2002), pp. 7–42. (cited on page 53)

[113] G. Schindler, M. Brown, and R. Szeliski, *City-scale location recognition*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1 –7. (cited on page 76)

[114] G. Sibley, C. Mei, I. Reid, and P. Newman, *Adaptive relative bundle adjustment*, in Robotics Science and Systems, 2009. (cited on page 13)

[115] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Bradford Company, Scituate, MA, USA, 2004. (cited on page 51)

[116] J. Sivic and A. Zisserman, *Video google: a text retrieval approach to object matching in videos*, in IEEE International Conference on Computer Vision, 2003, pp. 1470 –1477. (cited on pages 74, 75, and 76)

[117] I. Skrypnyk and D. Lowe, *Scene modelling, recognition and tracking with invariant image features*, in International Symposium on Mixed and Augmented Reality, 2004, pp. 110 – 119. (cited on page 71)

[118] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, *The new college vision and laser data set*, The International Journal of Robotics Research, 28 (2009), pp. 595–599. (cited on page 120)

[119] R. C. Smith and P. Cheeseman, *On the representation and estimation of spatial uncertainty*, International Journal of Robotic Research, (1985). (cited on page 11)

[120] S. M. Smith and J. M. Brady, *Susan - a new approach to low level image processing*, International Journal of Computer Vision, 23 (1995), pp. 45–78. (cited on page 51)

[121] N. Snavely, I. Simon, M. Goesele, R. Szeliski, and S. Seitz, *Scene reconstruction and visualization from community photo collections*, Proceedings of the IEEE, 98 (2010), pp. 1370 –1390. (cited on page 39)

[122] J. Solà, *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach.*, PhD thesis, Institut National Polytechnique de Toulouse, 2007. (cited on page 15)

[123] ——, *Consistency of the monocular EKF-SLAM algorithm for three different landmark parametrizations*, in IEEE International Conference on Robotics and Automation, 2010, pp. 3513 –3518. (cited on page 52)

[124] C. Stachniss and W. Burgard, *Mobile robot mapping and localization in non-static environments*, in International Conference on Artificial Intelligence, 2005, pp. 1324–1329. (cited on page 17)

[125] C. Stachniss, G. Grisetti, D. Hähnel, and W. Burgard, *Improved rao-blackwellized mapping by adaptive sampling and active loop-closure*, in Workshop on Self-Organization of AdaptiVE behavior (SOAVE), 2004, pp. 1–15. (cited on page 120)

[126] H. Strasdat, A. Davison, J. Montiel, and K. Konolige, *Double window optimisation for constant time visual SLAM*, in IEEE International Conference on Computer Vision, 2011. (cited on page 13)

[127] H. Strasdat, J. M. M. Montiel, and A. Davison, *Scale drift-aware large scale monocular SLAM*, in Proceedings of Robotics: Science and Systems, Zaragoza, Spain, June 2010. (cited on pages 13, 71, 111, and 114)

[128] H. Strasdat, J. M. M. Montiel, and A. J. Davison, *Real-time monocular SLAM: Why filter?*, in International Conference on Robotics and Automation, 2010, pp. 2657–2664. (cited on pages 4, 5, 76, and 77)

[129] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, D. Cremers, and R. Siegwart, *Towards a benchmark for RGB-D SLAM evaluation*, in Proceedings of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conference, 2011. (cited on page 120)

[130] N. Sünderhauf, K. Konolige, S. Lacroix, P. Protzel, and T. U. Chemnitz, *Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle*, in Tagungsband Autonome Mobile Systeme, 2005. (cited on page 68)

[131] A. Tapus and R. Siegwart, *Incremental robot mapping with fingerprints of places*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 2429 – 2434. (cited on page 62)

[132] S. Thrun, *Learning occupancy grids with forward sensor models*, Autonomous Robots, 15 (2002), pp. 111–127. (cited on page 61)

[133] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press, 2005. (cited on pages 15, 61, and 70)

[134] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, *Simultaneous localization and mapping with sparse extended information filters*, International Journal of Robotics Research, International Journal of Robotics Research (2004). (cited on page 12)

[135] C. Tomasi and T. Kanade, *Shape and motion from image streams under orthography—a factorization method*, in Proc. of the National Academy of Sciences of the United States of America, vol. 90, November 1993, pp. 9795–9802. (cited on page 39)

[136] B. L. Torsten Sattler and L. Kobbelt, *Fast image-based localization using direct 2d-to-3d matching*, in IEEE International Conference on Computer Vision, 2011. (cited on page 5)

[137] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, *Bundle adjustment - a modern synthesis*, in Vision Algorithms: Theory and Practice, LNCS, Springer Verlag, 2000, pp. 298–375. (cited on pages 3 and 42)

[138] R. Tsai, *A versatile camera calibration technique for high accuracy 3d machine vision using off-the-shelf tv cameras and lenses*, IEEE Journal of Robotics and Automation, (1987). (cited on page 33)

[139] J. W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977. (cited on page 45)

[140] I. Ulrich and I. Nourbakhsh, *Appearance-based place recognition for topological localization*, in IEEE International Conference on Robotics and Automation, 2000, pp. 1023 –1029. (cited on page 62)

[141] C.-C. Wang, *Simultaneous Localization, Mapping and Moving Object Tracking*, PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004. (cited on pages 7, 14, and 19)

[142] C.-C. Wang, D. Duggins, J. Gowdy, J. Kozar, R. MacLachlan, C. Mertz, A. Suppe, and C. Thorpe, *Navlab slammot datasets*, May 2004. Carnegie Mellon University. (cited on page 121)

[143] C.-C. Wang and C. Thorpe, *Simultaneous localization and mapping with detection and tracking of moving objects*, in IEEE International Conference on Robotics and Automation, 2002. (cited on pages 7 and 14)

[144] C.-C. Wang, C. Thorpe, M. Hebert, S. Thrun, and H. Durrant-whyte, *Simultaneous localization, mapping and moving object tracking*, International Journal of Robotics Research, (2004). (cited on page 14)

[145] C.-C. Wang, C. Thorpe, and S. Thrun, *Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas*, in IEEE International Conference on Robotics and Automation, 2003, pp. 842 – 849. (cited on pages 14 and 19)

[146] S. Wangsiripitak and D. W. Murray, *Avoiding moving outliers in visual SLAM by tracking moving objects*, in IEEE international conference on Robotics and Automation, 2009, pp. 705–710. (cited on pages 7 and 15)

[147] J. Weingarten and R. Siegwart, *3D SLAM using planar segments*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 3062 –3067. (cited on pages 55 and 56)

[148] G. Welch and G. Bishop, *An introduction to the kalman filter*, tech. rep., University of North Carolina at Chapel Hill, 1995. (cited on page 69)

[149] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, *Anisotropic Huber-L1 optical flow*, in Proceedings of the British Machine Vision Conference (BMVC), London, UK, September 2009. (cited on page 55)

[150] D. Wolf and G. S. Sukhatme, *Online simultaneous localization and mapping in dynamic environments*, in International Conference on Robotics and Automation, 2004, pp. 1301–1306. (cited on page 17)

[151] C. Wu, *SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)*. (cited on pages 77 and 80)

[152] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, *OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems*, in Proceedings of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010. Software available at `http://octomap.sf.net/`. (cited on pages 61 and 64)

[153] M. Yi, S. Steffano, K. Jana, and S. S. Shankar, *An Invitatio to 3-D Vision*, Springer, 2004. (cited on page 22)

[154] Z. Zhang, *Flexible camera calibration by viewing a plane from unknown orientations*, in International Conference on Computer Visio, 1999, pp. 666–673. (cited on page 33)

[155] L. Zhao, S. Huang, L. Yan, and G. Dissanayake, *Parallax angle parametrization for monocular SLAM*, in IEEE International Conference on Robotics and Automation, 2011, pp. 3117 –3124. (cited on page 52)