



DISSERTATION

Improving Accuracy and Robustness of Localisation and Mapping Algorithms Using Vision and Inertial Sensors

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften unter der Leitung von

A.o.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Institut für Automatisierungs- and Regelungstechnik (E376)

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von
DIPL.-ING. PETER GEMEINER

e0427244
Toryská 10
82107 Bratislava, Slowakei

Wien, im April 2010

Abstract

Robotics is a fascinating scientific discipline, which has connections to electronics, mechanics and software algorithms. The ultimate goal of robotics is to create autonomous intelligent robots capable of solving various tasks. Despite long term efforts, the first autonomous robots, with limited intelligence, appeared only in the 20th century. Today, robots are successfully helping humans in many areas, usually where high precision is demanded, more dangerous working conditions can be expected, or long-term distant work is needed. Due to the complexity of the tasks and the working environment, robotics remains an active research topic.

One of the main tasks of a mobile robot is to know its own position accurately. This self-localising in space is a priori condition for the navigation from its current position to a defined place. In environments, where no map has been given, the navigation to the new place is difficult, because the robot has to solve two parallel tasks.

These two parallel tasks, commonly known as the Simultaneous Localisation and Mapping (SLAM), attracted considerable attention within the scientific community over the past forty years. Due to the concentrated effort, several advanced algorithms have been developed, and nowadays it is generally considered that these algorithms are well-matured. In recent years, the attention of the community has focused mainly on various SLAM applications, which are using different kinds of sensors or mapping large environments.

In this thesis, we present three different contributions to the SLAM research topic. Our contributions aim to improve the robustness, speed or accuracy of SLAM through enhanced vision or a fusion of vision and inertial sensors.

The first contribution presents a fusion of vision and inertial sensors, which provides a robust solution to the localisation and mapping problem. The main idea is that we can use the complementary characteristics of these sensors in an efficient way. The involved algorithms have the same task as SLAM, but are working on different principles.

The second contribution introduces a high-speed camera to the SLAM framework, and discusses a more complex motion model. This improvement for the mobile robot increases the ability to move more rapidly in space.

The last contribution involves the possibility of using a Time-of-Flight (ToF) sensor. The standard vision camera does not provide direct scene depth measurements. The standard camera has to move in order to build a three-dimensional map of the unknown environment. Our result shows how the ToF camera direct scene depth measurements enhance the accuracy of the final map.

Zusammenfassung

Robotik ist eine faszinierende wissenschaftliche Disziplin, die Verbindungen zur Elektronik, der Mechanik und der Informatik hat. Die Hauptaufgabe der Robotik ist, autonome intelligente Roboter zu schaffen, die dazu fähig sind, verschiedenste Aufgaben zu lösen. Trotz langfristiger Anstrengung erschienen die ersten autonomen Roboter - zwar mit beschränkter Intelligenz - erst im 20. Jahrhundert. Heute helfen Roboter den Menschen erfolgreich in vielen Gebieten, z.B. wo hohe Präzision gefordert wird, gefährliche Arbeitsbedingungen erwartet werden, oder Arbeiten an entfernten Orten durchgeführt werden müssen. Wegen der Komplexität der Aufgaben und des Arbeitsumfeldes bleibt Robotik ein aktives Forschungsthema.

Eine der Hauptaufgaben eines beweglichen Roboters ist seine eigene Position genau feststellen zu können. Dieses Selbstlokalisieren im Raum ist a priori Bedingung für die Navigation von seiner gegenwärtigen Position hin zu einem definierten Platz. In Umgebungen, wo keine Karte gegeben worden ist, ist die Navigation zum neuen Platz schwierig, weil der Roboter zwei parallele Aufgaben lösen muss.

Diese zwei parallelen Aufgaben, allgemein bekannt als “Simultaneous Localisation and Mapping” (SLAM), zogen beträchtliche Aufmerksamkeit innerhalb der wissenschaftlichen Gemeinschaft im Laufe der letzten vierzig Jahre an sich. Wegen der fokussierten Bemühung sind mehrere fortschrittliche Algorithmen entwickelt worden, und heutzutage wird es allgemein betrachtet, dass diese Algorithmen nun ausgereift sind. In den letzten Jahren hat sich die Aufmerksamkeit der Gemeinschaft hauptsächlich auf verschiedene SLAM Anwendungen konzentriert, die mehrere Arten von Sensoren verwenden oder grosse Umgebungen darstellen.

In dieser Doktorarbeit präsentieren wir drei verschiedene Beiträge zum SLAM Forschungsthema. Unsere Beiträge haben das Ziel, die Robustheit, die Geschwindigkeit, oder die Genauigkeit des SLAM durch die verbesserte Vision, oder Fusion der Vision und Trägheitssensoren zu verbessern.

Der erste Beitrag präsentiert eine Fusion der Vision und Trägheitssensoren, der eine robuste Lösung zur Lokalisierung und Mapping Problem zur Verfügung stellt. Die beteiligten Algorithmen haben dieselbe Aufgabe wie SLAM, aber arbeiten an verschiedenen Grundsätzen.

Der zweite Beitrag führt eine Hochgeschwindigkeitkamera ins SLAM ein, und erklärt ein erweitertes Bewegungsmodell. Diese Verbesserung vergrößert die Fähigkeit für den beweglichen Roboter, sich schneller im Raum zu lokalisieren.

Der letzte Beitrag ist mit der Möglichkeit verbunden, einen “Time-of-Flight” (ToF) Sensor zu verwenden. Die standard Visionkamera stellt keine Tiefenmessungen für die Szene bereit. Unser Ergebnis zeigt, wie die Verwendung der Tiefenmessungen der ToF Kamera die Genauigkeit der resultierenden Karte erhöht.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Markus Vincze for supporting me over the years, and for giving me so much freedom to explore. His detailed review of my thesis, writing tips and general advices have kept me honest about all the details.

I would like to address my next thank to Dr. Andrew Davison, who was my second supervisor and the member of the thesis committee. His feedbacks throughout the years were very helpful and motivated my research work.

I would like to thank my many friends and colleagues at the Technical University of Vienna for their advices and help.

Last but not least, I would like to thank my family and my fiancé for their continuous encouragement and support.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Problem Declaration	3
1.3	Contribution Description	4
2	Description of Computer Vision Algorithms and Probabilistic Methods	6
2.1	Detecting and Matching Image Features	6
2.1.1	Harris Corners	7
2.1.2	Shi-Tomasi Features	8
2.1.3	Features from Accelerated Segment Test	8
2.1.4	Scale-Invariant Feature Transform	8
2.1.5	Matching Image Features	9
2.1.6	Random Sample Consensus	10
2.2	Single and Multiple View Geometry	10
2.2.1	Representing Camera Pose	11
2.2.2	Representing Camera Orientation	12
2.2.3	Camera Geometry	14
2.2.4	Lens Distortions	17
2.2.4.1	Radial Distortion	17
2.2.4.2	Tangential Distortion	18
2.2.4.3	Swaminathan's Model	19
2.2.5	Camera Calibration	20
2.2.5.1	The Image of the Absolute Conic	21
2.2.5.2	Calibration Conic	22
2.2.6	Pose Computation	23
2.2.7	Linear Triangulation	23
2.3	Probabilistic Techniques	25
2.3.1	The Problem of Dynamic State Estimation	26
2.3.2	Kalman Filter	26
2.3.3	Example: Tracking	27
2.3.4	Extended Kalman Filter	28
2.3.5	Unscented Kalman Filter	29
2.3.6	Particle Filter	31

3	Motion and Structure Estimation by Fusion of Vision and Inertial Data	33
3.1	Related Work	34
3.2	Contribution Description	36
3.3	Inertial Measurement Unit	36
3.3.1	Our Inertial Measurement Unit - MT9B	37
3.3.2	Example: Integration vs. Estimation of Attitude	39
3.4	Motion and Structure Estimation	41
3.4.1	Measurements	41
3.4.1.1	Visual Measurements	42
3.4.1.2	IMU Measurements	42
3.4.2	Modelling Camera Motion	42
3.4.2.1	Pose Calculation	43
3.4.2.2	Motion Model	43
3.4.2.3	Multi-Rate Filters	45
3.4.3	Mapping Scene Structure	46
3.4.3.1	Structure Estimation Algorithm	46
3.4.3.2	Pose-Recovery	48
3.4.3.3	Model Update	48
3.5	Experimental Results	49
3.5.1	Experiments Using a Mobile Platform	49
3.5.1.1	Experimental Setup	50
3.5.1.2	Motion Evaluation	51
3.5.1.3	Structure Evaluation	53
3.5.1.4	Run-time Evaluation	55
3.5.2	Pose-Recovery Experiments Using a Robotic Arm	56
3.5.2.1	Experimental Setup	57
3.5.2.2	Motion Evaluation	57
3.5.2.3	Structure Evaluation	59
3.6	Conclusion	59
4	High-Speed Vision SLAM	61
4.1	Related Work	63
4.2	Increasing Localisation Robustness	64
4.3	Contribution Description	64
4.4	MonoSLAM	65
4.4.1	Example: MonoSLAM	66
4.4.2	Probabilistic Description	68
4.4.3	Motion Representation	69
4.4.3.1	Constant Velocity Motion Model	70
4.4.4	Mapping	72
4.4.4.1	Computing Map Updates	72
4.4.4.2	Feature Initialisation	72
4.4.4.3	Additional Modules and Extensions	73
4.5	Improving Localisation Robustness	76

4.5.1	Dense Visual Information	76
4.5.2	Constant Acceleration Motion Model	76
4.5.3	Motion Model Parameter	80
4.5.3.1	Example: Motion Model Simulation	81
4.5.4	Skipping Vision Information	81
4.6	Experimental Results	83
4.6.1	Setup	83
4.6.2	Ground-truth Computation	84
4.6.2.1	Ground Truth Accuracy	85
4.6.3	Evaluations	86
4.6.3.1	Evaluation of Localisation Accuracy	86
4.6.3.2	Mapping Evaluations	88
4.7	Conclusion	89
5	Good Corners for Structure and Motion Recovery	92
5.1	Related Work	93
5.2	Contribution	93
5.3	Bundle Adjustment	94
5.3.1	The Perspective-Projective Camera Model	94
5.3.2	Iterative Non-linear Optimisation	94
5.3.3	Example: Simulated Bundle Adjustment	97
5.4	Corner Selection Scheme	97
5.4.1	Structure Tensor	99
5.5	Experimental Results	100
5.5.1	ToF Camera	100
5.5.2	Static Depth Noise Modelling	102
5.5.3	Feature Extraction	102
5.5.3.1	2D Corner Extraction	102
5.5.3.2	3D Corner Extraction	103
5.5.3.3	Planar Corner Extraction	106
5.6	Conclusion	106
5.6.1	Future Work	107
6	Conclusion	108
	Bibliography	110

Chapter 1

Introduction

Robotics is a challenging research field, closely related to Artificial Intelligence (AI), which attracts the attention of the general public. The reason for this is that in many areas an intelligent autonomous robot is needed to assist or substitute humans. The goal of robotics is to build robots capable of intelligent behavior, which involves perception, reasoning, learning, communicating, and acting in complex environments [76].

Currently, there are already robots working in different fields, but with limited intelligence, usually controlled by humans or pre-programmed commands. Off-the-shelf intelligent robots capable of solving autonomously different tasks in an indoor or an outdoor environment are still not available. Due to the fact that the robot consists of mechanical, electrical and software algorithms, and the complexity of the environment, its development is still a research topic for many scientific institutes worldwide.

The general robot decision and acting scheme as depicted in Fig. 1.1 (ref e.g. Siegwart and Nourbakhsh in [90]), has four main modules: *perception* responsible for acquiring sensor data, *localisation and mapping* estimating the current position, *cognition and path planning* deciding on the next action and *action* executing the selected action. The main challenge in robotics has been localisation and mapping, which processes the perceived data, estimates the pose and map of the scene, and outputs the results to the cognition and path planning module.

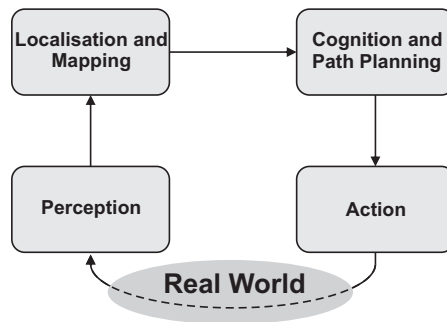


Figure 1.1: Robot's architecture.

The parallel localisation and mapping algorithm has been already used in several autonomous robotic platforms. We present shortly three examples of prototypes applications, which use these algorithms and have been introduced recently.

One of the most successful stories in robotics is the Darpa Grand Challenge competition [98]. This prize competition for driverless cars, sponsored by the Defense Advanced Research Projects Agency (DARPA), attracted considerable attention in the robotics community. The success is due to the localisation and mapping algorithms, which helped the driverless cars to accurately navigate in a desert or a small town.

Recently introduced robots copying the human appearance and behavior commonly known as humanoids, are a field of interest for many researchers worldwide. Due to their possible general use in home, or office environment, they are a very appealing research platform, and many universities and private companies develop humanoids. Many of them have a navigation module [95], which helps them to localise and map in indoor environments.

The localization and mapping algorithms helps autonomous vehicles to navigate in environments, where no global position can be received. This environments include outer space and earth's places e.g. mines or oceans. An interesting application, introduced by Eustice et al. in [35], presented successful localisation in the surrounding of the RMS Titanic while mapping the wreck.

1.1 Motivation

It is only after several decades of research that the localisation and mapping algorithms are generally considered to be well-advanced. However, there are further possibilities on how to improve the state-of-the-art algorithms e.g. mapping larger environments [9] or using a more complex perceptual input.

An overall robot's architecture is depicted in Fig. 1.1. The focus of this work is the robot's module, which enables autonomous finding of desired places in an unknown environment. The module is known as *localisation and mapping*, and is responsible for solving two parallel problems. The first is to maintain the actual robot's position in the three-dimensional space, and the second problem is to estimate the metric volume of the surrounding. These two tasks should help the robot to navigate itself to the desired position while avoiding potential obstacles.

Perfect noiseless sensors would ease these tasks, but this is not the case in real-world scenarios. The localisation and mapping algorithms have to handle noisy measurements, which plays a crucial role in the resulting pose and map estimates. However, any inaccurate estimations further influence the path planning, so it might happen that the robot can not find the desired destination or get lost.

We think that using a combination of different sensors, or using recently introduced more advanced sensors can improve the performance of the state-of-the-art localisation and mapping algorithms. Our motivation is that these efforts should enable the robot to move and react faster, be more robust against pose lost, estimate the metric volume of the scene more accurately or prevent one sensor failure.

1.2 Problem Declaration

A working environment of a robot significantly influences the complexity of its tasks. An industrial robot mounted on a fixed place usually does not have to deal with obstacles, the path to be executed is pre-programmed, and its position can be accessed anytime with high accuracy. The difficulty in robotics is that the advantage of precise navigation is lost when the robot is mobile.

A mobile robot has a completely different environment to the industrial robot, because it has to deal with e.g. moving or occluded obstacles. The main difficulty of a dynamic environment is that position changes of moving obstacles have to be predicted. Successful navigation has paramount importance for a mobile robot while moving to a goal position.

Successful navigation involves, as depicted in Fig. 1.1, a collaboration of these four modules. In the last four decades localisation has been an important topic for the mobile robotics research, and due to this long-term effort significant results were reported. The successful localisation algorithms developed in past years have a common denominator known as the *probabilistic techniques*. These techniques helped to solve the localisation problem with many reported successful practical applications.

The localisation problem can be separated into two categories. The first category represents localisation with a known or given map of the environment. This given map aids the localisation task significantly, and several methods have been proposed [11] on how to solve this problem. The second category introduces localisation without a map of the scene, and the mobile robot has to build the map while moving in the unknown environment. This truly autonomous task is very challenging, because the robot has to solve simultaneously the localisation and mapping task as depicted in Fig. 1.2. In the robotics literature this is known as Simultaneous Localisation and Mapping (SLAM) [30], or it is called *parallel* or *co-occurrent* localisation and mapping.

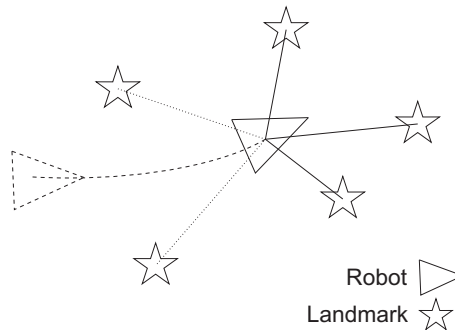


Figure 1.2: The robot has to localise itself while estimating positions of unknown landmarks in the scene.

We think that the most promising sensor for navigation is a single camera, because this sensor provides very rich information about the content of the scene. Humans can process vision information effortlessly, but it proves to be a very difficult task for computers [76].

The problem with using a single camera for navigation is that it can not measure the depth of the environment directly. Due to the perspective-projective nature of a camera the depth information is lost after an image is taken. The only possibility on how to reconstruct the lost depth information is to move with the camera and estimate the motion and structure from several different views. To solve this task in robotics SLAM or Structure from Motion (SfM) algorithm can be applied, and recently several successful examples have been demonstrated (e.g. [77, 56]). Despite having the same goal, the robotics and computer vision communities use different types of algorithms.

In mobile robotics, a real scene can contain dynamic objects, which move along unpredictable trajectories. The robot using a vision SLAM system can encounter situations, when a rapid motion is needed. In such cases, the SLAM system should be able to react appropriately in a short amount of time. This problem can be tackled by fusing low frequency vision information with other high frequency signals or by increasing the frame rate of the camera.

This collision avoidance scenario is one example of a possible application of the vision SLAM system. We think that there are many more industrial possibilities, where such a system can be applied. These possibilities include, e.g., automotive or gaming industries.

Moving in an unknown scene, the SLAM system has to rely on the identification of features in the images received from a single camera. When a feature is wrongly identified as a good corner, and inserted in the map of the scene, it can bias the SLAM estimates. To avoid these bias estimates, a recently introduced sensor can be used. A ToF camera, which provides amplitude and depth data simultaneously. By using both of these measurements the wrong scene features can be detected and pruned.

1.3 Contribution Description

A mobile robot is usually equipped with various sensors based on different physical principles. These sensors should help the robot to navigate in the environment more robustly and safely. We think that for the navigational task, a camera provides one of the richest information content about a new scene. Due to this fact, every contribution of our work is by using a camera as the primary sensor.

Our goal was to improve the robustness, speed or accuracy of SLAM using the camera as the primary sensor. To achieve this goal, we empirically tested and improved state-of-the-art localisation and mapping algorithms. To provide a more comprehensive overview, this thesis is divided into four chapters.

The Chap. 2 provides a brief description of computer vision algorithms and probabilistic techniques, which were used in this work.

In Chap. 3, the benefits of using a fusion of vision and inertial data for structure and motion estimation are described. We empirically proved (e.g. [40]) that the combination of these two sensors is well-suited for a robot, because it is more robust against e.g. temporary linear or rotational accelerations. The introduced structure and motion estimation scheme is considered to be suboptimal, and currently more advanced algorithms have been introduced (e.g. [25]).

The Chap. 4 presents the advantages of using a sensor with higher frame rate while simultaneously localising and mapping in real-time [39]. The main advantage is that the camera movement is more stable and robust against jitter. The other contribution of this part is an extended motion model, which takes advantage of additional second order continuous parameters. Simulated and experimental results proved that using the sensor with a higher frame rate and the extended motion model improves the performance of SLAM framework.

The Chap. 5 introduces a corner selection scheme, which enables it to choose good three-dimensional features [41]. Our selection scheme exploits the amplitude and direct depth measurements of the newly introduced ToF camera. We proved on a simulated SfM example how false corners influence structure and mapping results. Using our proposed scheme false corners can be rejected, and the accuracy of the pose and map results is increased. This increased map accuracy helps the mobile robot in a unknown scene to navigate to a goal position.

Chapter 2

Description of Computer Vision Algorithms and Probabilistic Methods

This chapter provides an overview of computer vision algorithms and probabilistic methods, which were used in this work. In Sec. 2.1, we describe the selected feature detection and matching algorithms. Sec. 2.2 introduces: the possible representations of the camera pose, the geometric properties of a perspective-projective camera, the pose computation and the linear triangulation. Last Sec. 2.3 presents several stochastic methods, which help to estimate the unknown state of a linear or non-linear problem.

2.1 Detecting and Matching Image Features

The first step in computer vision is to create an image of the environment on a two-dimensional array on a photosensitive device. This image is typically formed by a camera through a lens that perceives the scene with a limited field of view. The photocells convert the scene image into a two-dimensional, time-varying matrix with intensity values $I(x, y)$.

As a second step, image pre-processing can be involved, but from the information-theoretical point of view this results in information lost. Sonka et al. in [93] stated that from the information-theoretical point of view: “the best pre-processing is no pre-processing”.

The next step is to select features, which would be suitable for tracking and further scene analysis. One of the most important criterion is that these features have to be distinctive, which allows to match them easily through an image sequence. For this purpose one of the most often selected features are the so called *corners*. In this work, the terms *corner* and *landmark* have an interchangeable meaning.

The existence of a corner in the image is equivalent to the summation of the outer product of the gradients [67] :

$$G(\mathbf{x}) = \sum_{\tilde{\mathbf{x}} \in \mathbf{W}(\mathbf{x})} \nabla I(\tilde{\mathbf{x}}) \nabla I^T(\tilde{\mathbf{x}}) = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \in \mathbf{R}^{2 \times 2}, \quad (2.1)$$

when $G(\mathbf{x})$ is nonsingular. $\mathbf{x} = [x, y]^T$ is the position of a corner in an image. $\nabla I(\tilde{\mathbf{x}})$ is the gradient calculated at $\tilde{\mathbf{x}} = [\tilde{x}, \tilde{y}]^T \in \mathbf{W}(\mathbf{x})$, where $\mathbf{W}(\mathbf{x})$ represents the search window. The eigenvalues λ_1 and λ_2 of $G(\mathbf{x})$ represent the *curvature* of the image surface, and form a rotationally invariant description of G . However, the components of G are calculated using only horizontal and vertical gradients, so they are *not rotationally invariant*.

The two-dimensional matrix $G(\mathbf{x})$ is the so called *structure tensor*. The term *tensor* refers to a representation of an array of data, and the rank describes the number of indices needed to address it. $G(\mathbf{x})$ has a rank that equals two.

As depicted in Fig. 2.1, we can generally consider four cases of detecting corners, where the red window is the detection area, and the black rectangle is a virtual object. In the first position (see Fig. 2.1(a)), image intensity is fairly constant in the whole window and the curvature is little in any direction. The second position (see Fig. 2.1(b)) has little curvature along the edge, but large curvature perpendicular at the edge. The last two positions (see Fig. 2.1(c) and Fig. 2.1(d)) have large curvatures in any direction.

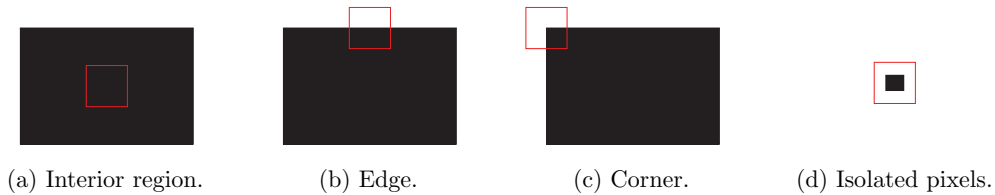


Figure 2.1: Different cases for a feature detector.

2.1.1 Harris Corners

The analysis of the eigenvalues λ_1 and λ_2 of $G(\mathbf{x})$ indicates that these eigenvalues can be divided into three regions as depicted in Fig. 2.2(a).

To classify corners, edges and interior regions, the lines dividing the regions have to be selected. Instead of labeling every single pixel it is suitable to have a single cornerness measure. Harris and Stephens in [44] proposed this cornerness measure:

$$C(G) = \mathbf{det}(G) + \kappa \times \mathbf{trace}^2(G). \quad (2.2)$$

Empirical tests proved that the best performance of this cornerness measure is achieved when $\kappa = 0.04$ as displayed in Fig. 2.2(b).

In the literature the features found by this algorithm are commonly known as *Harris* or *Plessey* corners.

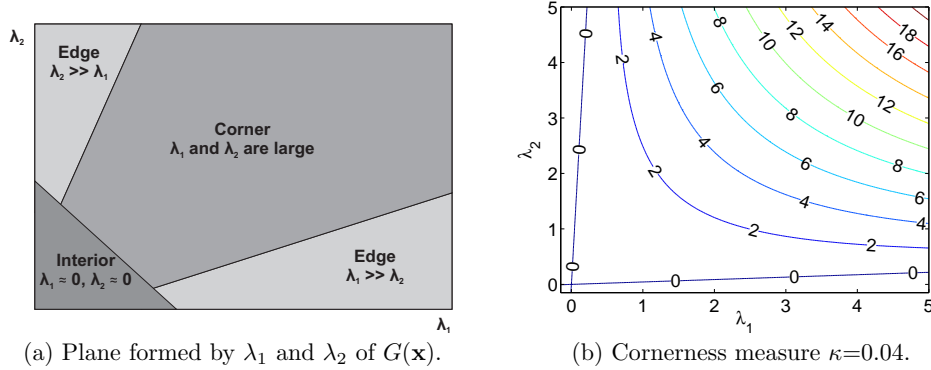


Figure 2.2: Division of eigenvalues space into distinct feature regions.

2.1.2 Shi-Tomasi Features

In [89], an improved algorithm for detecting the so called Shi-Tomasi features is demonstrated. However, except the different cornerness measure the rest of the computation equals to the above-mentioned Harris corner detection (see Sec. 2.1.1). The authors proved that for image patches undergoing affine transformation this is a better measure of the corner strength:

$$\min(\lambda_1, \lambda_2) > \lambda, \quad (2.3)$$

where λ is a predefined threshold.

2.1.3 Features from Accelerated Segment Test

Rosten and Drummond introduced in [86] a detector called Features from Accelerated Segment Test (FAST). As depicted in Fig. 2.3, this feature detector considers contiguous pixels in a Bresenham circle of radius r around a candidate point p . If n , in the Fig. 2.3 n equals 12, contiguous pixels are brighter than the candidate p by at least t , then p is considered to be a feature.

Additionally, a machine learning algorithm is used to discriminate true candidate corners based on entropy measurements. This algorithm creates a decision tree which correctly classifies all corners seen in the training set.

In [86], an efficient detector is generated for n equals 9, which is, according to authors the most reliable FAST detector. According to [86] this detector outperforms Harris or Shi-Tomasi features in speed and accuracy.

2.1.4 Scale-Invariant Feature Transform

Lowe in [65] presented a method for extracting distinctive image features from image sequences. These features are invariant to image scale and rotation, and the author called the method Scale-Invariant Feature Transform (SIFT).

Brief summary of the four major steps of SIFT method:

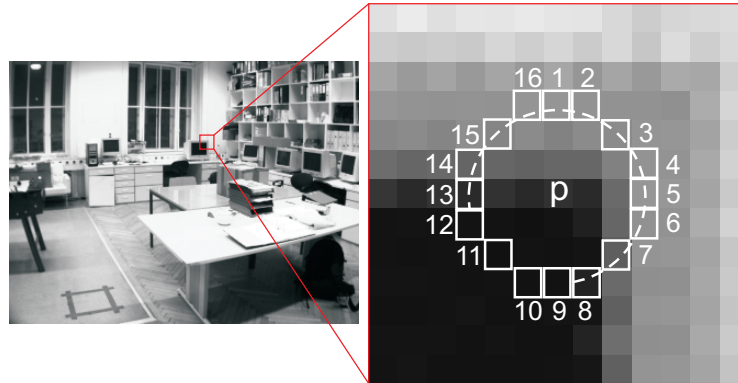


Figure 2.3: FAST 12 point segment test corner detection. The highlighted squares are used for corner detection around a candidate corner p . The arc represents 12 contiguous pixels, which are brighter than p by more than the threshold.

- **Scale-space extrema detection:** For efficient key point localisation in scale space a convolution of the image with Difference-of-Gaussians (DoG) is involved. Searching across all possible scales provides locations invariant to scale changes.
- **Key point localisation:** At each candidate location a three-dimensional quadratic function is fitted. This fitting provides (according to authors) improved matching and stability. The principal curvature (see Eq. 2.1) is computed to eliminate edge features.
- **Orientation assignment:** An orientation histogram is formed based on the local image gradient information. Peaks in this histogram correspond to dominant directions.
- **Key point descriptor:** Is created by computing the magnitude and orientation of the local gradient around the key point location. Followed by weighting with a Gaussian function, and finally the accumulation into orientation histograms is performed.

Due to the provided robust matching, SIFT features are an interesting and popular method. However, this method has a higher computational burden, and we did not use it in this work.

2.1.5 Matching Image Features

Image matching is a fundamental concept of many problems in computer vision, including feature tracking, or scene metric model computation. In this section a simple, but widely used matching criterion is explained.

A way to compute the similarity between two patches is the normalised Sum-of-Squared-Distances (SSD), or also known as normalised sum-of-squared-differences:

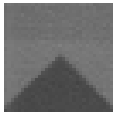
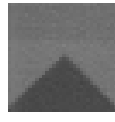
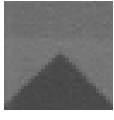
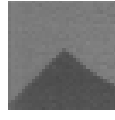
Template		
Image		
normalised SSD $\tilde{S}(x, y)$	0.0026	0.0057

Table 2.1: An example of matching two similar regions around a candidate corner using the normalised SSD criterion.

$$\tilde{S}(x, y) = \frac{\sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} [T(x', y') - I(x + x', y + y')]^2}{\sqrt{\sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} T(x', y')^2 \sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} I(x + x', y + y')^2}}. \quad (2.4)$$

The smaller is the value of $\tilde{S}(x, y)$ at a particular pixel, the more similarity exists between the template T and the image I in the neighborhood of that pixel. An example is displayed in Tab. 2.1.

Other methods to match the template patch with the image include e.g. Normalised Cross-Correlation (NCC). However, the computation of these criteria is simple comparing to the selection of suitable matches. As depicted in Tab. 2.1, two similar patches produce very similar matching results. The challenging task is to select a suitable *threshold*, which would distinguish corresponding features. Finding a corresponding set of features is very important for reconstructing the metric map of the scene from multiple views.

The mentioned matching criteria have poor performance when the lighting conditions change, and are not rotationally invariant. However, they have been used in real-time localisation and mapping systems due to the low computational load.

2.1.6 Random Sample Consensus

Random Sample Consensus (RANSAC) [36] is an opposing technique to least squares methods. Instead of considering as many measurements as possible, RANSAC uses a small subset of data. When possible, this small subset is subsequently extended with further consistent data.

As depicted in Fig. 2.4, the main advantage of RANSAC is that it can efficiently reject outliers from a data set. This popular property is widely used for feature matching, when two sets of candidates are matched, implying geometric constraints [45].

2.2 Single and Multiple View Geometry

Detection and matching of scene features, as described in the previous section, is an obligatory initial step for subsequent vision algorithms. These algorithms can e.g.

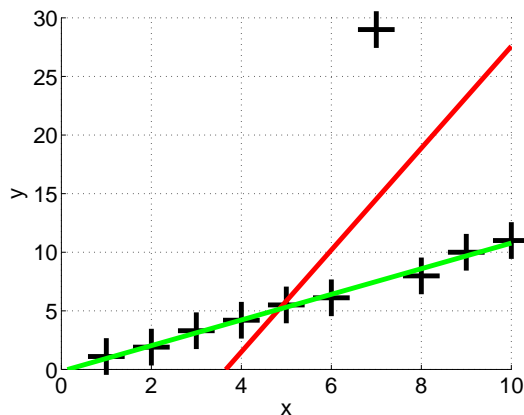


Figure 2.4: Fitting a line in the two-dimensional space using RANSAC and least squares method. RANSAC fitted green line had rejected the outlier point, but least squares fitted red line had considered all points.

compute the orientation and position of the camera based on a known scene object. They can recover unknown scene metric information etc. In both mentioned algorithms, a camera model and motion representation are needed.

For the camera orientation and position computation, a single view of the scene and a known object model is provided. When an instance of the known object can be found in the image, the pose can be computed as presented in the section.

Recovering the scene map is a more challenging task, this is because the perspective-projective camera can not measure depth. At least two different views are needed, where the camera position and orientation is provided.

In this thesis, we have been using several algorithms based on a single or multiple views, and this section is a brief summary of them in five sub-sections. Firstly, an overview of the motion representation is presented. Secondly, a camera internal model is introduced. Thirdly, a camera calibration procedure is demonstrated. Fourthly, a camera position and orientation computation is suggested. Finally, a linear triangulation procedure based on multiple views is presented.

2.2.1 Representing Camera Pose

Three-dimensional Euclidean space can be represented globally by a Cartesian coordinate frame [67], where every point is identified by $\mathbf{X} = [X, Y, Z]^T$.

In this work different Cartesian coordinate systems have been used, and an example containing two frames is defined as follows:

$$\mathbf{X}^W = \mathbf{R}^{WC} \mathbf{X}^C + \mathbf{t}^{WC}, \quad (2.5)$$

where \mathbf{X}^C is a point in the *camera* frame. \mathbf{R}^{WC} is the camera orientation represented by a rotation matrix, and \mathbf{t}^{WC} is the translational vector. This orientation and translation define the geometric relation between the camera and the *world* coordinate system. In computer vision, this relation is commonly known as the *camera pose* [88],

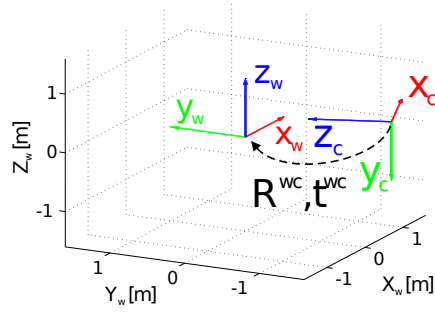


Figure 2.5: The geometric relation between the world and camera coordination systems.

or camera rigid body motion [67]. As displayed in Fig. 2.5, when the pose is known every point \mathbf{X}^C in the camera frame can be transformed to \mathbf{X}^W in the world frame.

Every camera pose has an inverse representation, and point \mathbf{X}^W can be easily transformed to the camera frame:

$$\begin{aligned}\mathbf{X}^C &= (\mathbf{R}^{WC})^T \mathbf{X}^W - (\mathbf{R}^{WC})^T \mathbf{t}^{WC}, \\ \mathbf{X}^C &= \mathbf{R}^{CW} \mathbf{X}^W - \mathbf{R}^{CW} \mathbf{t}^{WC}, \\ \mathbf{X}^C &= \mathbf{R}^{CW} \mathbf{X}^W + \mathbf{t}^{CW}.\end{aligned}\tag{2.6}$$

In this work, a perspective-projective camera has been used. This introduces different geometric properties, as included in the Euclidean space. In Euclidean space e.g. two parallel lines do not intersect, but in projective space an intersection exists.

However, we need to calculate the transformation between two frames as easily as in the Cartesian frame. The solution is to use *homogeneous* coordinates. The homogeneous coordinates of a point \mathbf{X} with length n are usually written as:

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} X_x \\ X_y \\ X_z \\ 1 \end{bmatrix},\tag{2.7}$$

where the new length equals $n + 1$.

Using homogeneous coordinates Eq. 2.6 can be formulated as:

$$\begin{bmatrix} \mathbf{X}^C \\ 1 \end{bmatrix} = \begin{bmatrix} X_x^C \\ X_y^C \\ X_z^C \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{CW} & \mathbf{t}^{CW} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}^W \\ 1 \end{bmatrix}.\tag{2.8}$$

2.2.2 Representing Camera Orientation

The orientation of the camera pose can be expressed using different mathematical representations. Possible representations are: rotation matrices, Euler angles, quaternions,

etc. As described in Eq. 2.9, a typical orientation representation is based on three parameters. The disadvantage of computing orientation using three parameters in three-dimensional (3D) space is that singularities occur.

The orientations \mathbf{R}^{WC} and \mathbf{R}^{CW} are represented using orthogonal rotation matrices, which is a part of the *rotation group* $SO(3)$ [67]. This matrix expresses a compound sequence of orientations following e.g. the *Roll-Pitch-Yaw* or the Euler angle $\mathbf{Z}-\mathbf{Y}-\mathbf{X}$ convention

$$\mathbf{R}^{WC} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

where ψ is yaw, ϕ is pitch and θ is roll. These angles are a specific sequence of Euler angles often used in robotics or aerospace applications.

We can read the rotation sequence in Eq.2.9 in two ways. If we start from right, the rotations are referred to the world coordinate frame. Starting from the left side causes the orientations to be represented in the camera coordinate system.

An interesting example, where the singularities in 3D orientation computation can be demonstrated, is the *gimbal lock* problem. As depicted in Fig. 2.7, this problem occurs when e.g. an aircraft pitches up 90° so the roll and yaw axes become parallel. When using three parameters (see Eq. 2.9) to represent the orientation, singularities in the calculations occur.

One of the popular orientation representations called quaternions, introduced by Sir William Rowan Hamilton in 1843, solved this problem in 3D space robustly and efficiently as displayed in Fig. 2.8. Similarly as a two-dimensional complex number can be represented as a point in a plane, a four-dimensional quaternion can be displayed as a point in space.

A nice overview of quaternion algebra and analysis is presented in e.g. [18, 58, 101].

The basic multiplication rule, carved on the Brougham Bridge in Dublin, is defined as:

$$i^2 = j^2 = k^2 = ijk = -1, \quad (2.10)$$

where i, j, k are three orthogonal unit spatial vectors [18].

The common description of a normalised quaternion is:

$$\mathbf{q} = (q_0, q_x, q_y, q_z), \quad (2.11)$$

where q_0 is the *real* part of the quaternion. (q_x, q_y, q_z) is the *vector* part of the quaternion.

Since quaternions contain four parameters they can be visualised by projecting three out of four components into a 3D subspace. As depicted in Fig. 2.6, a visualisation of these quaternions is called *quaternion map*. These maps are obtained by connecting a set of quaternions to form a smooth curve, for example.

The error in quaternions is calculated as a sum of distances between the truth (\mathbf{q}_i^{gt}) and measured quaternions (\mathbf{q}_i^m):

$$\sum_{i=1}^n d(\mathbf{q}_i^{gt}, \mathbf{q}_i^{*m}). \quad (2.12)$$

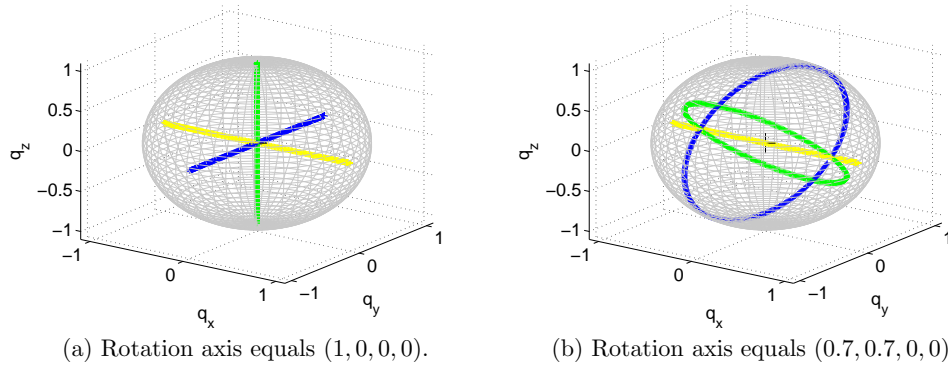


Figure 2.6: Quaternion maps computed for different rotation sequences, and depicted in 3D subspace. These independent rotations about three orthogonal axes can be identified by their colours: x is yellow, y is blue and z is green.

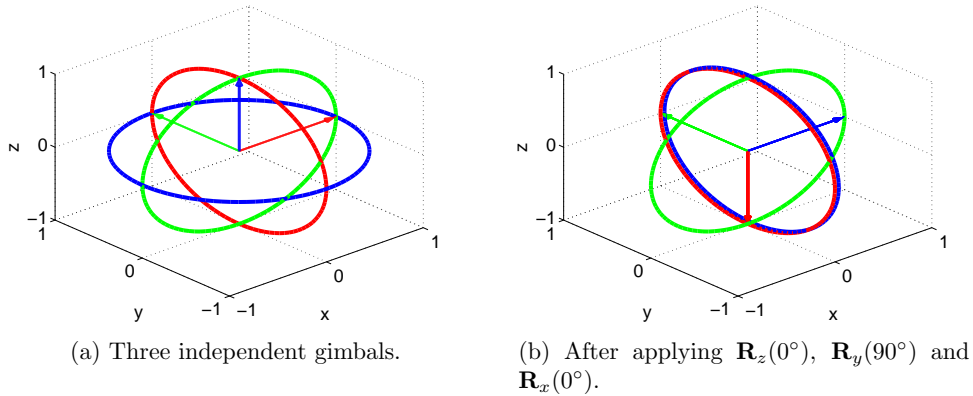


Figure 2.7: Three orthogonal gimbals (2.7(a)) rotated using the sequence $\mathbf{R}_z(0^\circ)$, $\mathbf{R}_y(90^\circ)$ and $\mathbf{R}_x(0^\circ)$ result in the so called gimbal lock situation (2.7(b)). This means that one degree of freedom is lost, because two gimbals become parallel.

2.2.3 Camera Geometry

In this sub-section, the most general camera model (widely used in computer vision and robotics) is explained. As depicted in Fig. 2.9, this model is commonly known as *pinhole* or *perspective-projective* model [45]. The basic feature of this model is that it projects 3D points $\mathbf{X} = [X, Y, Z]$ on the *image plane*. The centre of the projection is the origin of the Cartesian coordinate system and the image plane is assumed to be at $Z = f$.

As displayed in Fig. 2.9, a 3D point is mapped on the image plane using these equations:

$$x = -f \frac{X}{Z}, \quad y = -f \frac{Y}{Z}, \quad (2.13)$$

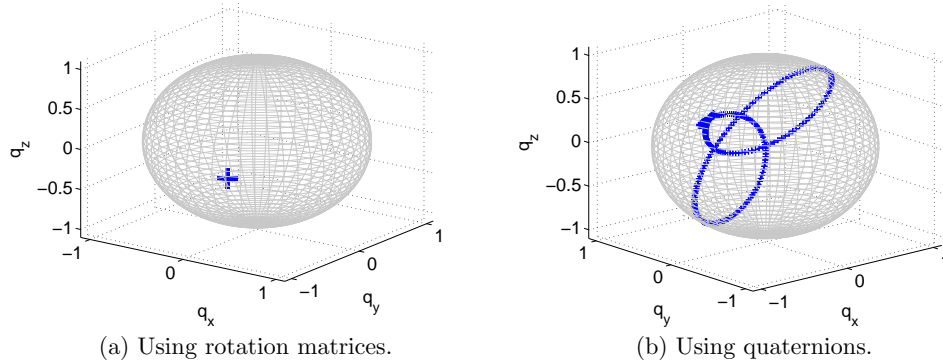


Figure 2.8: Comparison of orientation computation, using the rotation matrix and quaternion representation, when the gimbal lock occurs. Both results should form an identical closed curve. However, the computation of orientation using rotation matrices contains a singularity, and any subsequent rotation around x or z is ignored.

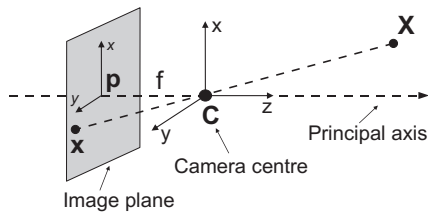


Figure 2.9: Pinhole camera model. The centre of the projection or optical centre is \mathbf{C} , \mathbf{P} is the principal point, \mathbf{X} is the 3D point to be projected and \mathbf{x} is the result of the projection.

and the projected two-dimensional point is represented by $\mathbf{x} = [x, y]$.

However, a more convenient representation would be to assume to localise the camera centre \mathbf{C} behind the image plane. This case is known as *frontal* pinhole imaging model [67]:

$$x = f \frac{X}{Z}, \quad y = f \frac{Y}{Z}. \quad (2.14)$$

As defined in the Eq. 2.13 and Eq. 2.14, due to the projection of a 3D point on the image plane is the depth dimension lost. As depicted in Fig. 2.9, this lost dimension can not be recovered from a single view, because infinitely many 3D points can project on a single two-dimensional point.

In homogeneous coordinates, see 2.2.1, can be the Eq. 2.14 rewritten as:

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ f & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.15)$$

In the Eq. 2.13 and Eq. 2.14 it is assumed that the origin of coordinates of the image plane is at the principal point \mathbf{P} . However, in the practice \mathbf{P} is usually not exactly in the origin of the coordinates, and the exact position needs to be estimated.

When a precise position of \mathbf{P} is known, this not invertible mapping is valid:

$$\lambda \mathbf{x} = \begin{bmatrix} fx + zp_x \\ fy + zp_y \\ z \end{bmatrix} = \begin{bmatrix} f & p_x & 0 \\ f & p_y & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.16)$$

where $\mathbf{P} = [p_x, p_y]$ and λ is the unknown scale.

The focal length f and the coordinates of the principal point p_x and p_y are commonly known as *intrinsic* camera parameters. In practice, their exact values are usually unknown and have to be estimated. This process of camera parameters estimation is known as *calibration* [113], and the result can have e.g. a matrix form:

$$\mathbf{K} = \begin{bmatrix} f & p_x \\ f & p_y \\ 1 & 1 \end{bmatrix}. \quad (2.17)$$

Then the projection in the Eq. 2.16 can be rewritten as:

$$\lambda \mathbf{x} = \mathbf{K}[\mathbf{I}|0]\mathbf{X}. \quad (2.18)$$

The projected point \mathbf{x} , see Eq. 2.14, is localised in the image plane in the *normalised coordinate system* [67]. However, we need the representation of this point in the pixel values. After multiplying with a scaling factor, this projected point in the normalised system can be transferred to the camera system (x_{cam}, y_{cam}) . As depicted in Fig. 2.10, when the coordinates of the principal point are added to the point in the camera system, we receive a point in the pixel representation.

In pinhole cameras, there is the possibility of having non-square pixels due to the difference between the horizontal and vertical resolution. The solution to this is to introduce two different scaling factors m_x and m_y . To represent \mathbf{K} in terms of pixel dimensions the focal lengths α_x and α_y are suggested:

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & p_x \\ & \alpha_y & p_y \\ & & 1 \end{bmatrix}, \quad (2.19)$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$. Another intrinsic parameter is the skew s , which models the possibility of the angle between x and y pixels. However, skew s equals zero for most cameras.

The previously defined relation between the world and camera coordinate systems, see Eq. 2.8, allows it to project a 3D world point on the image plane. The rotation \mathbf{R}^{CW} and translation \mathbf{t}^{CW} between these two coordinate systems are called *extrinsic* camera parameters.

The projection using intrinsic and extrinsic camera parameters can be written as:

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{R}^{CW}[\mathbf{I}|\mathbf{t}^{CW}]\mathbf{X}. \quad (2.20)$$

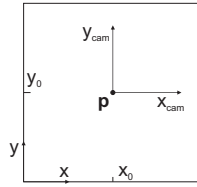


Figure 2.10: Two possible representations of the coordinate system in the image plane. The first (x_{cam}, y_{cam}) is known as the camera system. The second (x, y) is called the image coordinate system.

2.2.4 Lens Distortions

In practice, considering only previously suggested intrinsic and extrinsic camera parameters is not sufficient. The camera lens usually introduces additional non-linear distortions, which can significantly influence the image measurement process. These distortions typically increase with the decreasing quality, and price of the lens.

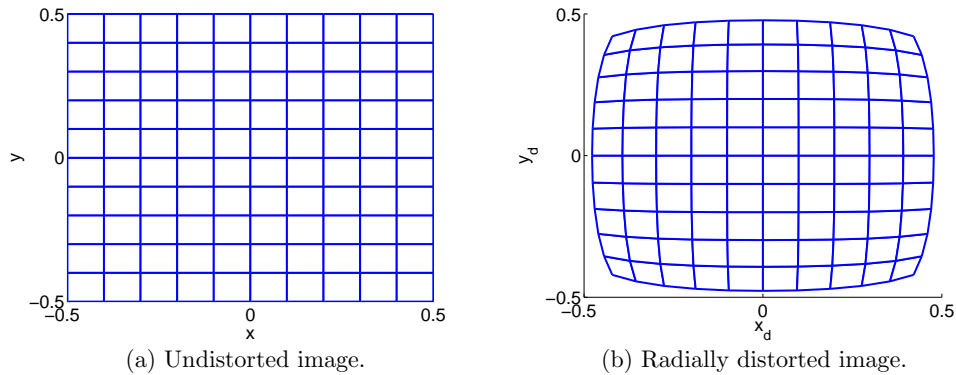


Figure 2.11: An ideal image (a) deformed by the radial distortions (b).

2.2.4.1 Radial Distortion

The most important lens deviation is generally a radial distortion (see Fig. 2.11). This distortion can be modelled as follows:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = L(r) \begin{bmatrix} x - p_x \\ y - p_y \end{bmatrix}, \quad (2.21)$$

where x and y are the coordinates of a point in the ideal image, p_x and p_y are the coordinates of the principal point, x_d and y_d are the coordinates of the distorted point, and r equals:

$$r = \sqrt{(x - p_x)^2 + (y - p_y)^2}. \quad (2.22)$$

The function L can contain arbitrarily many radial parameters κ :



Figure 2.12: On the left side a distorted image taken with a wide-angle lens (2.1mm - horizontal angle is 81°) is displayed. On the right side an undistorted image is presented.

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \dots, \quad (2.23)$$

but in practice typically two or three radial coefficients are used.

Approaches of how to estimate these radial coefficients have been introduced by e.g. Zhang in [113] or by Hartley and Zisserman in [45].

2.2.4.2 Tangential Distortion

The image distortion can be approximated with more complex models, where all except radial additional tangential parameters are implemented. An example of this extended distortion model is presented in Fig. 2.12. The image taken by a camera with a wide-angle lens is undistorted using two radial ($\kappa_1 = -0.3178$, $\kappa_2 = 0.0915$) and two tangential ($\kappa_3 = 0.0004$, $\kappa_4 = -0.0009$) coefficients.

The tangential distortion is added to the radially distorted coordinates (x_d, y_d) :

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} \kappa_3 a_1 + \kappa_4 a_2 \\ \kappa_3 a_3 + \kappa_4 a_1 \end{bmatrix} \quad (2.24)$$

where a_1 , a_2 and a_3 equal:

$$\begin{aligned} a_1 &= 2xy \\ a_2 &= r^2 + 2x^2 \\ a_3 &= r^2 + 2y^2. \end{aligned} \quad (2.25)$$

r is defined in Eq. 2.22, and (x, y) are the coordinates of a point in the ideal image.

A comparison between radial and tangential distortion is displayed in Fig. 2.13(b). A distorted image (see Fig. 2.13(a)) is an abstraction of a real example as displayed in e.g. Fig. 2.12. In Fig. 2.13(c), only undistortion using radial coefficients is presented. In Fig. 2.13(d), undistortion considering both radial and tangential coefficients is introduced.

For further details please refer to e.g. [12].

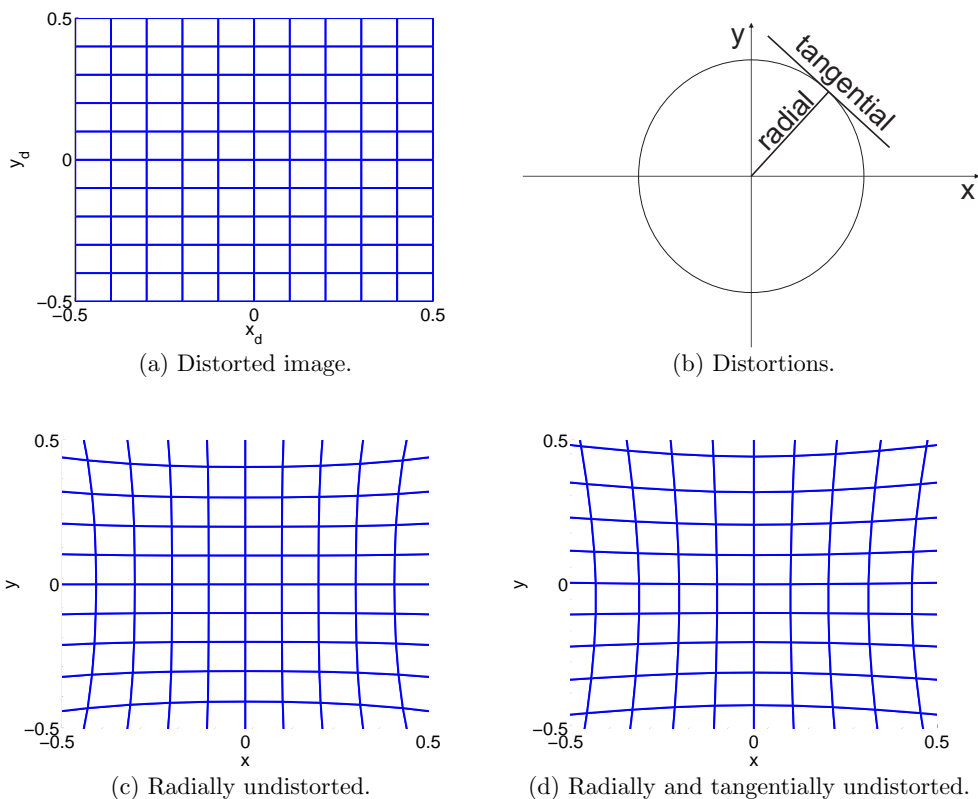


Figure 2.13: Image undistortion using two types of parameters.

2.2.4.3 Swaminathan's Model

To allow proper compensation of the lens inaccuracies, the radial and tangential parameters introduced previously have to be modeled carefully and precisely. The main problem of these complex models containing radial and tangential parameters, is that there exist no closed-form inverse representation.

When calculating the projection of a 3D point on the image plane this inverse representation is not needed. However, when computing the re-projected two-dimensional position of a point on the image plane, this inverse form is necessary. This re-projection is commonly known as *back-projection* [45].

The back-projection has this form:

$$\begin{bmatrix} X & Y & Z \end{bmatrix} = \lambda \begin{bmatrix} x & y & 1 \end{bmatrix}^T, \quad (2.26)$$

where λ is the unknown scale, X , Y and Z are the coordinates of an unknown 3D point, and x and y are coordinates of a known point on the image plane.

Despite efforts to model the lens inaccuracies precisely, in some applications only one parameter distortion model proved to be satisfactory [27]. This one parameter is the radial distortion, and the model has been introduced by Swaminathan and Nayar in [96].

When substituting L in Eq. 2.23 with:

$$L(r) = \frac{1}{\sqrt{1 + 2\kappa_1 r^2}}, \quad (2.27)$$

the other undistorting equations can be found in Sec. 2.2.4.1.

For the back-projection the inversion of L in Eq. 2.27 is needed:

$$L(r) = \frac{1}{\sqrt{1 - 2\kappa_1 r^2}}. \quad (2.28)$$

An example of undistorting image points considering two different κ_1 coefficients is depicted in Fig. 2.14.

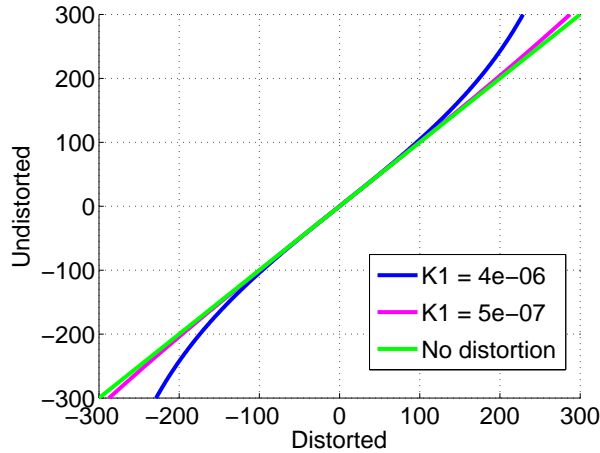


Figure 2.14: Undistorting image points using Swaminathan’s approximated model.

2.2.5 Camera Calibration

The intrinsic and extrinsic camera parameters described in the above-mentioned subsections are usually unknown, and have to be estimated. This camera parameters estimation process is called *camera calibration*.

As presented by Zhang in [113], there are three possible ways to calibrate a camera:

- *photogrammetric calibration* - using a precisely known 3D object,
- *self-calibration* - no calibration object needed and
- *others* - calibrating from pure rotation, or using *vanishing points* (see Fig. 2.15).

In this thesis, a calibration algorithm [12], based on the work of Zhang in [113], has been used. This algorithm lies between the photogrammetric and self-calibration approach, because it uses a two-dimensional metric instead of 3D information.

The contribution of Zhang’s method [113], is that it provides a closed-form solution to the calibration problem using only a planar pattern. The basic idea is depicted in Fig. 2.16. This calibration algorithm needs at least two different scene views with at least three points in each view.

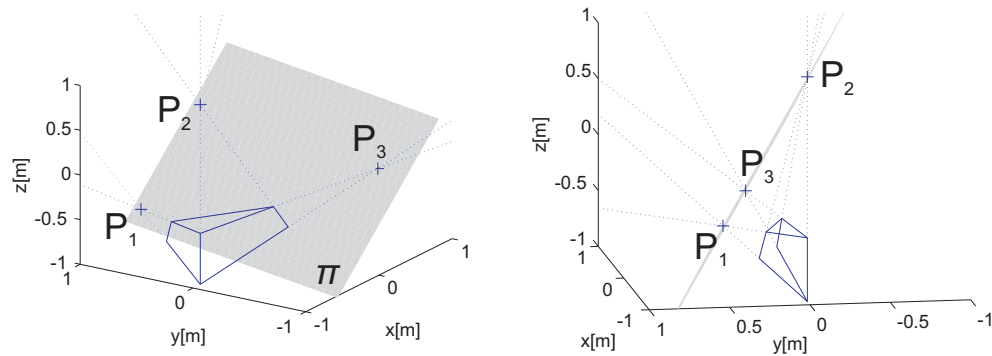


Figure 2.15: The points P_1 , P_2 and P_3 are commonly known as vanishing points. The plane fitted through these three vanishing points is called as the plane at infinity. Two displayed images represent the same example from different viewpoints.

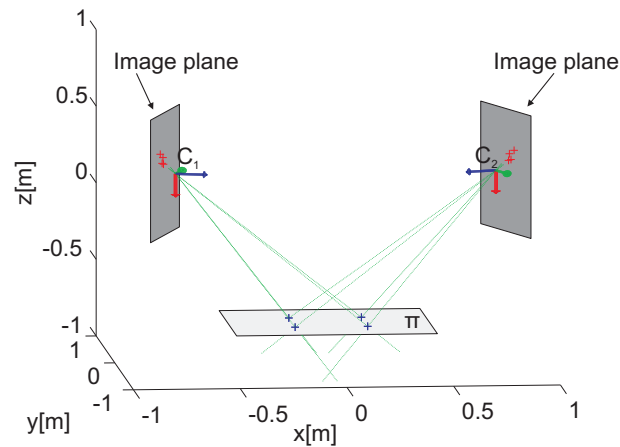


Figure 2.16: If the camera centre moves, then the images are in general not related by a projective transformation, unless all space points are coplanar [45].

2.2.5.1 The Image of the Absolute Conic

An important concept in the camera calibration literature is the *Absolute Conic* (AC) (see Fig. 2.17) lying in the *plane at infinity* (see Fig. 2.15). The projection of AC in an image is called *Image of the Absolute Conic* (IAC), and is constant under Euclidean transformation. This means that for constant intrinsic camera parameters the relative position of IAC is constant even when the camera is moving. The concepts of AC and IAC are abstract and imaginary, but an approximated description is displayed in Fig. 2.17.

In [45], a simple calibration example is introduced, where three planar squares provide sufficient constraints to compute \mathbf{K} . This simple calibration uses the idea of a conic ω , which contains the calibration matrix \mathbf{K} .

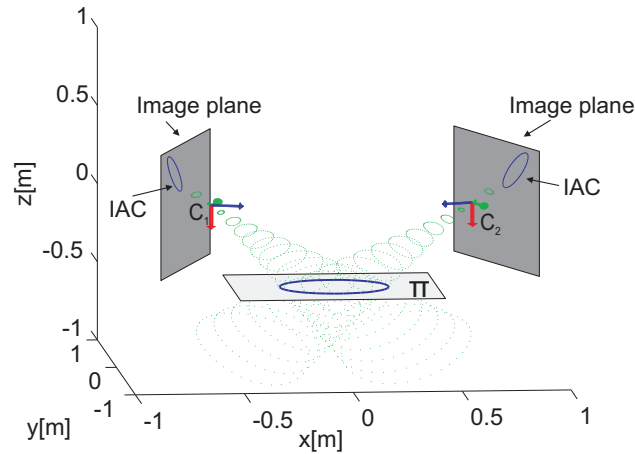


Figure 2.17: Absolute Conic is lying in the plane at infinity (π). Two projections of AC from different viewpoints have constant relative positions. These projections of AC are commonly known as IAC.

2.2.5.2 Calibration Conic

For visualisation purposes it is suitable to consider a different conic called *calibration conic* (see Fig. 2.18):

$$C = \mathbf{K}^{-T} \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix} \mathbf{K}^{-1}, \quad (2.29)$$

where \mathbf{K} is defined in Eq. 2.19.

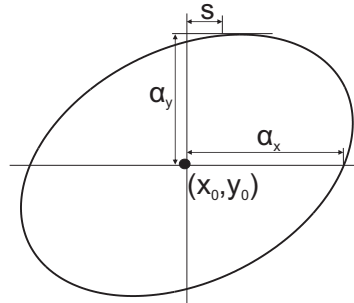


Figure 2.18: Reading the intrinsic camera parameters (see Sec. 2.2.3) from a calibration conic.

The camera intrinsic parameters are easily read from the calibration conic. The principal point \mathbf{P} is the centre of the conic, and the scales (α_x , α_y) are easily identified. The skew parameter s can be read as depicted in Fig. 2.18.

Please, for further details about the camera calibration refer to [45].

2.2.6 Pose Computation

The process of computing the position and orientation of a given calibrated camera from a known 3D scene or an object is called *pose computation* [66]. In this thesis, only techniques based on point-based computation has been used.

Point-based methods compute the pose using 3D points and the corresponding image projections. These methods are commonly known as the *Perspective-n-Point* (PnP) problem [36].

Depending of the number of points n , PnP problem can have the following number of solutions [36]:

- $n = 3$: theoretically, the P3P problem admits eight possible solutions. However, Fischler and Bolles remark in [36] that in fact four solutions exist.
- $n = 4$: when the four points are coplanar then P4P problem admits a unique solution [66].

For an extended overview of the PnP problem please refer to [36] and [66].

In this work, the pose computation is based on the algorithm of Lu et al. [66], which is robust against noise or possible outliers.

The starting point for this algorithm is the *object-space* collinearity error vector:

$$e_i = (\mathbf{I} - \mathbf{V}_i)(\mathbf{R}\mathbf{X}_i + \mathbf{t}), \quad (2.30)$$

where \mathbf{V}_i is the observed line-of-sight projection matrix. This matrix projects the point orthogonally to the line of sight defined by the image point \mathbf{x}_i :

$$\mathbf{V}_i = \frac{\mathbf{x}_i \mathbf{x}_i^T}{\mathbf{x}_i^T \mathbf{x}_i}. \quad (2.31)$$

The goal is to minimise the sum of the squared error:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|(\mathbf{I} - \mathbf{V}_i)(\mathbf{R}\mathbf{X}_i + \mathbf{t})\|^2, \quad (2.32)$$

which can not be solved in closed-form. However, it can be computed using an iteration algorithm, and an example is depicted in Fig. 2.19.

Schweighofer and Pinz presented in [88] an improved version of Lu et al. algorithm. They empirically proved that the algorithm of Lu et al. introduces pose ambiguity for planar objects, and proposed to refine the output.

In this thesis, we have been using the algorithm of Schweighofer and Pinz [88], where the reference object is e.g. an artificial planar target displayed in Fig. 2.20.

2.2.7 Linear Triangulation

In computer vision, given two cameras and the corresponding image point pairs, the computation of the coordinates of an unknown 3D point is called *triangulation*.

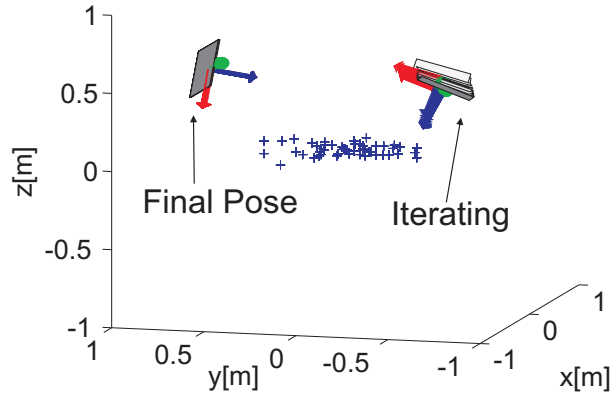


Figure 2.19: Using an iterating algorithm to find the camera pose.

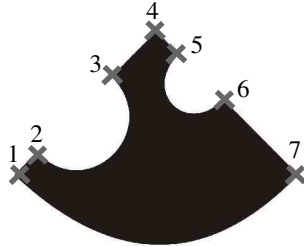


Figure 2.20: The camera pose is initialized using this object. The 3D positions of corner features (1-7), and the correspondence between 3D word and 2D image plane positions are assumed to be known [88].

A simple linear triangulation algorithm can be found in [45]. Having e.g. two cameras \mathbf{P} and \mathbf{P}' , where $\mathbf{P} = \mathbf{K}\mathbf{R}^{CW}[\mathbf{I}|\mathbf{t}^{CW}]$ (see Eq. 2.20), the projections ($\mathbf{x} = [x, y]$ and $\mathbf{x}' = [x', y']$) of an unknown point \mathbf{X} :

$$\begin{aligned} \mathbf{x} &= \mathbf{P}\mathbf{X}, \\ \mathbf{x}' &= \mathbf{P}'\mathbf{X}. \end{aligned} \tag{2.33}$$

are measured in the cameras' image planes.

The coordinates of \mathbf{X} can be calculated as a solution (see Fig. 2.21) of an equation of the form $\mathbf{A}\mathbf{X} = 0$, where:

$$\mathbf{A} = \begin{bmatrix} x\mathbf{p}^{3T} & - & \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} & - & \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} & - & \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} & - & \mathbf{p}'^{2T} \end{bmatrix}. \tag{2.34}$$

\mathbf{p}^{iT} and \mathbf{p}'^{iT} is the i^{th} row of the matrix \mathbf{P} , and \mathbf{P}' respectively.

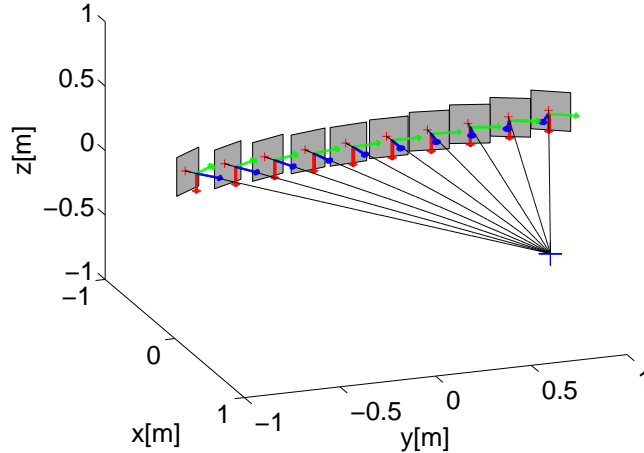


Figure 2.21: Triangulation using ten identical cameras.

2.3 Probabilistic Techniques

One of the most important issues of using sensors in the real world are the inaccurate and uncertain observations. In robotics for example, the desired goal of a mobile robot is to move to a pre-defined location. Any inaccurate sensor measurement disturbs the position state estimate, for example this causes the robot to not reach the final place.

Estimating the behavior of a dynamic system by assuming noisy measurements is one of the fundamental problems of an interdisciplinary field called *control theory*. “**Estimation** can be viewed as the *process of selecting a point from a continuous space* - the best estimate” [7]. In the control theory, this process is also known as *filtering*, and it is the key idea of eliminating noise in this thesis.

To find the best estimate of the dynamic system, a carefully designed equation plant has to be introduced. These state estimation equations have roots in the *probability theory*, and contain:

- an imperfect initial estimate,
- disturbances of its dynamics - process noise,
- only some components are observable and
- the sensor measurements are corrupted by noise.

In control theory literature (see e.g. [7]) are the first two items encompassed in the *process* model and the second two items in the *measurement* model of the system. The so called recursive prediction-update behavior, using these two models, is an essential part of all *Bayes* filters introduced below [97].

2.3.1 The Problem of Dynamic State Estimation

The problem of *linear* dynamic state estimation is:

$$\begin{aligned}\mathbf{x}(k) &= \mathbf{F}(k-1)\mathbf{x}(k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1) + \nu(k-1) \\ \mathbf{z}(k) &= \mathbf{H}(k)\mathbf{x}(k) + \mathbf{w}(k)\end{aligned}\quad (2.35)$$

similar to the *non-linear* dynamic state estimation:

$$\begin{aligned}\mathbf{x}(k) &= f[k-1, \mathbf{x}(k-1), \mathbf{u}(k-1)] + \nu(k-1) \\ \mathbf{z}(k) &= h[k, \mathbf{x}(k)] + \mathbf{w}(k)\end{aligned}\quad (2.36)$$

where the goal is to estimate the *true* state $\mathbf{x}(k)$ corrupted by unknown zero-mean white Gaussian noise $\nu(k-1)$. The state is estimated using observed unprecise measurements disturbed by unknown zero-mean white Gaussian noise $\mathbf{w}(k)$.

$\mathbf{F}(k-1)$, $\mathbf{G}(k-1)$ and $\mathbf{H}(k)$ are known as possibly time-varying matrices. f is the known non-linear process and h is the known non-linear measurement function.

2.3.2 Kalman Filter

In control theory, one of the most important algorithms for estimating a quantity of interest from inaccurate and uncertain measurements is *Kalman filter* (KF). Due to the suitability for discrete-time systems and easy implementation is KF widely used in computer vision and robotics.

If the initial state, process noise and measurement noise are *Gaussian* and mutually independent, and the evolution of the system state and measurements can be described with *linear* time-varying difference equations, then KF is an *optimal* Minimum Mean Square Error (MMSE) estimator. If the initial state, process or measurement noise are *not* Gaussian, then KF is the *linear* MMSE estimator [7].

The equation plant of KF has two distinct phases. The first phase is **Prediction** (Process model):

$$\begin{aligned}\hat{\mathbf{x}}(k|k-1) &= \mathbf{F}(k-1)\hat{\mathbf{x}}(k-1|k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1) \\ \mathbf{P}(k|k-1) &= \mathbf{F}(k-1)\mathbf{P}(k-1|k-1)\mathbf{F}^T(k-1) + \mathbf{Q}(k-1),\end{aligned}\quad (2.37)$$

where $\hat{\mathbf{x}}$ is the state vector. $\mathbf{F}(k-1)$, $\mathbf{G}(k-1)$ and $\mathbf{Q}(k-1)$ are assumed to be known and possibly time-varying. The initial state $\hat{\mathbf{x}}(0)$, generally unknown, is modeled as a random variable. $\mathbf{u}(k-1)$ is known input vector, $\nu(k-1)$ is the process noise with covariance $\mathbf{Q}(k-1)$. $\mathbf{F}(k-1)$ is the state transition matrix and $\mathbf{G}(k-1)$ is the input matrix. The covariance associated with the estimate $\hat{\mathbf{x}}$ is known as \mathbf{P} .

The second phase is **Update** (Measurement model):

$$\begin{aligned}\tilde{\mathbf{y}}(k) &= \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1) \\ \mathbf{S}(k) &= \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^T(k) + \mathbf{R}(k) \\ \mathbf{K}(k) &= \mathbf{P}(k|k-1)\mathbf{H}(k)^T\mathbf{S}(k)^{-1} \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)\tilde{\mathbf{y}}(k) \\ \mathbf{P}(k|k) &= (\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k))\mathbf{P}(k|k-1),\end{aligned}\quad (2.38)$$

where $\mathbf{z}(k)$ is the measurement vector. This measurement vector is uncorrelated from the estimation error, that means they are *orthogonal*. $\mathbf{H}(k)$ is the possibly time-varying output matrix. The matrix $\mathbf{K}(k)$ is the Kalman gain, and $\tilde{\mathbf{y}}(k)$ is called *innovation* or measurement residual. $\mathbf{R}(k)$ is the possibly time-varying measurement noise matrix. The two noise matrices $\mathbf{Q}(k-1)$ and $\mathbf{R}(k)$ are assumed to be *mutually independent*.

2.3.3 Example: Tracking

The use of KF is demonstrated on a kinematic motion model revealing the major issues encountered in estimation.

Consider a continuous state-space system:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\tilde{\mathbf{v}}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{p} & \tilde{\mathbf{v}} \\ 0 & 1 \\ 0 & 0 \end{bmatrix}} \begin{bmatrix} \mathbf{p} \\ \tilde{\mathbf{v}} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tilde{\mathbf{a}} \quad (2.39)$$

where \mathbf{x} is the state vector of the system. The state vector contains the position \mathbf{p} and velocity vector $\tilde{\mathbf{v}}$ of a target object. The acceleration of the target $\tilde{\mathbf{a}}$ is unknown, and represents the system noise.

When we transform this one-dimensional continuous state-space system to a two-dimensional discrete state-space system the transition matrix \mathbf{F} has this form:

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.40)$$

Not only do we observe the position of the target, but we estimate the velocity as well:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (2.41)$$

The process noise covariance matrix equals:

$$\mathbf{Q} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \quad (2.42)$$

and this matrix can be adjusted as introduced e.g. in [7].

The measurement noise covariance matrix describes how accurate the new observations are:

$$\mathbf{R} = \begin{bmatrix} 0.75 & 0 \\ 0 & 0.75 \end{bmatrix}, \quad (2.43)$$

and these values can be estimated by analysing a sequence of measurements.

We have an imperfect estimate of the initial state vector $\hat{\mathbf{x}}(0)$:

$$\hat{\mathbf{x}}(0) = [0 \ 0 \ 3 \ 3]. \quad (2.44)$$

The exact initial covariance is not known, so we approximate the initial values $\mathbf{P}(0)$:

$$\mathbf{P}(0) = \begin{bmatrix} 8 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \quad (2.45)$$

The true state \mathbf{x} of the target object is corrupted by the system noise ν , and only an estimate $\hat{\mathbf{x}}$ of the true state can be computed (see the left side of Fig. 2.22).

When a new position measurement is available the state of the target object is updated. The state update depends on the computed gain \mathbf{K} , which can be:

- large, if the state is inaccurate and the measurement is accurate or
- small, if the state is accurate and the measurement is inaccurate.

In this example the gain is large, and the response to the new measurements is rapid as displayed on the right side of Fig. 2.22.

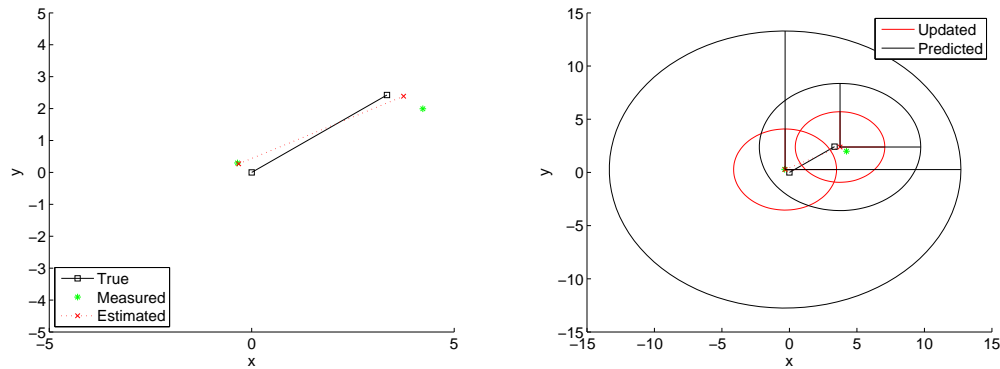


Figure 2.22: Tracking a target object. The state vector encompasses the position and velocity of the target. Left is the true, observed and estimated position. Right is the predicted and updated covariance of the position.

2.3.4 Extended Kalman Filter

In many real applications non-linearities occur, and KF can not be considered to be *optimal*. In this work, kinematic models have been used, which contain e.g. rotations. To track the state vector of an object undergoing a non-linear transformation a *suboptimal* technique has to be applied.

For this purpose, the well-known *Extended Kalman Filter* (EKF) (see [7] or [97]) can be applied. The idea of EKF is based on the *linearisation* of the non-linear process and measurement equations. In this thesis, a first-order EKF based on first order series expansion has been used.

Prediction (Process model):

$$\begin{aligned}\hat{\mathbf{x}}(k|k-1) &= f[k-1, \hat{\mathbf{x}}(k-1|k-1), \mathbf{u}(k-1)] \\ \mathbf{P}(k|k-1) &= \mathbf{F}(k-1)\mathbf{P}(k-1|k-1)\mathbf{F}(k-1)^T + \mathbf{Q}(k),\end{aligned}\quad (2.46)$$

where f is the non-linear process function and $\mathbf{F}(k-1)$ equals:

$$\mathbf{F}(k-1) = \left. \frac{\partial f(k-1)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(k-1|k-1)}.\quad (2.47)$$

Update (Measurement model):

$$\begin{aligned}\tilde{\mathbf{y}}(k) &= \mathbf{z}(k) - h[k, \hat{\mathbf{x}}(k|k-1)] + \mathbf{w}(k-1) \\ \mathbf{S}(k) &= \mathbf{H}(k)\mathbf{P}(k|k-1)\mathbf{H}^T(k) + \mathbf{R}(k)\end{aligned}\quad (2.48)$$

$$\mathbf{K}(k) = \mathbf{P}(k|k-1)\mathbf{H}^T(k)\mathbf{S}^{-1}(k)\quad (2.49)$$

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)\tilde{\mathbf{y}}(k)\quad (2.50)$$

$$\mathbf{P}(k|k) = (\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k))\mathbf{P}(k|k-1),\quad (2.51)$$

where h is the non-linear measurement function and $\mathbf{H}(k)$ equals:

$$\mathbf{H}(k) = \left. \frac{\partial h(k)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(k|k-1)}.\quad (2.52)$$

The other matrices and vectors have the same meaning as explained in Sec. 2.3.2.

2.3.5 Unscented Kalman Filter

An alternative algorithm to EKF is known as Unscented Kalman Filter (UKF), summarised by Julier and Uhlmann in [52].

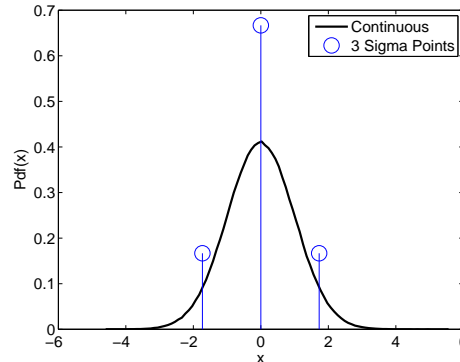


Figure 2.23: Representation of the standard normal distribution and its discrete approximation.

The basic idea of UKF is stated in [51]: “We use the intuition that it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation.”

UKF introduces deterministic sampling of the probability distribution function, which is depicted in Fig. 2.23. The *deterministically* selected *sigma* points are computed as follows:

$$\begin{aligned}\mathcal{X}_0(k-1|k-1) &= \hat{\mathbf{x}}(k-1|k-1) , \\ \mathcal{X}_i(k-1|k-1) &= \hat{\mathbf{x}}(k-1|k-1) + (\sqrt{(n+\kappa) \mathbf{P}(k-1|k-1)})_i , \\ \mathcal{X}_{i+n}(k-1|k-1) &= \hat{\mathbf{x}}(k-1|k-1) - (\sqrt{(n+\kappa) \mathbf{P}(k-1|k-1)})_i ,\end{aligned}\quad (2.53)$$

$$\begin{aligned}W_0 &= \frac{\kappa}{(n+\kappa)} , \\ W_i &= \frac{1}{2(n+\kappa)} , \\ W_{i+n} &= \frac{1}{2(n+\kappa)} ,\end{aligned}\quad (2.54)$$

where $\kappa \in \mathfrak{R}$, $\sqrt{(n+\kappa) \mathbf{P}(k|j)}_i$ is the i^{th} row of the matrix square root. The W_i is the weight associated with the i^{th} row, and assuming a Gaussian distribution $\kappa = 3 - n$.

The **predicted** sigma points are defined below:

$$\mathcal{X}^i(k|k-1) = f[\mathcal{X}^i(k-1|k-1), \mathbf{u}(k-1), k-1].\quad (2.55)$$

Predicted mean and covariance is computed as:

$$\hat{\mathbf{x}}(k|k-1) = \sum_{i=0}^{2n} W_i \mathcal{X}_i(k|k-1),\quad (2.56)$$

$$\mathbf{P}(k|k-1) = \sum_{i=0}^{2n} W_i [\mathcal{X}_i(k|k-1) - \hat{\mathbf{x}}(k|k-1)][\mathcal{X}_i(k|k-1) - \hat{\mathbf{x}}(k|k-1)]^T.\quad (2.57)$$

Instantiate each of the prediction points through the observation model:

$$\mathcal{Y}_i(k|k-1) = h[\mathcal{X}_i(k|k-1)] + \mathcal{V}_i(k).\quad (2.58)$$

The **predicted** observation is calculated by:

$$\tilde{\mathbf{y}}(k) = \sum_{i=0}^{2n} W_i \mathcal{Y}_i(k|k-1).\quad (2.59)$$

Finally, **updates** of mean and covariance are computed as follow:

$$\begin{aligned}\mathbf{K}(k) &= \mathbf{P}_{xy}(k|k-1) \mathbf{P}_{yy}^{-1}(k|k-1), \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)[\mathbf{z}(k) - \tilde{\mathbf{y}}(k)], \\ \mathbf{P}(k|k) &= \mathbf{P}(k|k-1) - \mathbf{K}(k) \mathbf{P}_{yy}(k|k-1) \mathbf{K}^T(k),\end{aligned}$$

where

$$\mathbf{P}_{yy}(k|k-1) = \sum_{i=0}^{2n} W_i [\mathcal{Y}_i(k|k-1) - \tilde{\mathbf{y}}(k)] [\mathcal{Y}_i(k|k-1) - \tilde{\mathbf{y}}(k)]^T, \quad (2.60)$$

and

$$\mathbf{P}_{xy}(k|k-1) = \sum_{i=0}^{2n} W_i [\mathcal{X}_i(k|k-1) - \hat{\mathbf{x}}(k|k-1)] [\mathcal{Y}_i(k|k-1) - \tilde{\mathbf{y}}(k)]^T. \quad (2.61)$$

2.3.6 Particle Filter

Another algorithm, which offers an alternative to EKF and UKF, is called Particle Filter (PF) or sequential Monte-Carlo methods [97]. The key idea of PF is to represent the posterior probability density function by samples drawn from this posterior. This idea is similar to UKF, but in PF the samples are drawn *randomly*. The next major difference between PF and other non-linear filters (EKF and UKF) is that the posterior distribution is approximated and non-parametric. This advantage enables PF to represent different probability distributions, and not only e.g. Gaussians.

The PF algorithm contains following steps:

1. generation of initial particles (see Fig. 2.24(a) at time equals 0),
2. calculating prediction (see the red crosses in Fig. 2.24(a)),

$$\hat{\mathbf{x}}(k|k-1) = f[k-1, \hat{\mathbf{x}}(k-1|k-1), \mathbf{u}(k-1)] \quad (2.62)$$

3. computation of weights,

$$\omega_i = \frac{1}{\sqrt{\sigma}} e^{-\frac{1}{2} \frac{1}{\sqrt{\sigma}} ((\mathbf{z}(k) - \tilde{\mathbf{y}}(k))^2)} \quad (2.63)$$

where

$$\tilde{\mathbf{y}}(k) = h[k, \hat{\mathbf{x}}(k|k-1)] \quad (2.64)$$

followed by normalisation,

$$\omega_i = \frac{\omega_i}{\sum_{j=1}^M \omega_j} \quad (2.65)$$

4. selection step (see the blue lines in Fig. 2.24(a)), which encompasses resampling algorithm (e.g. residual resampling algorithm [60]), and selecting particles with resampled indices.

As depicted in Fig. 2.24(b), with sufficient particles is PF more accurate than EKF or UKF. However, in real-time applications, a large amount of particles can decrease the performance of the system.

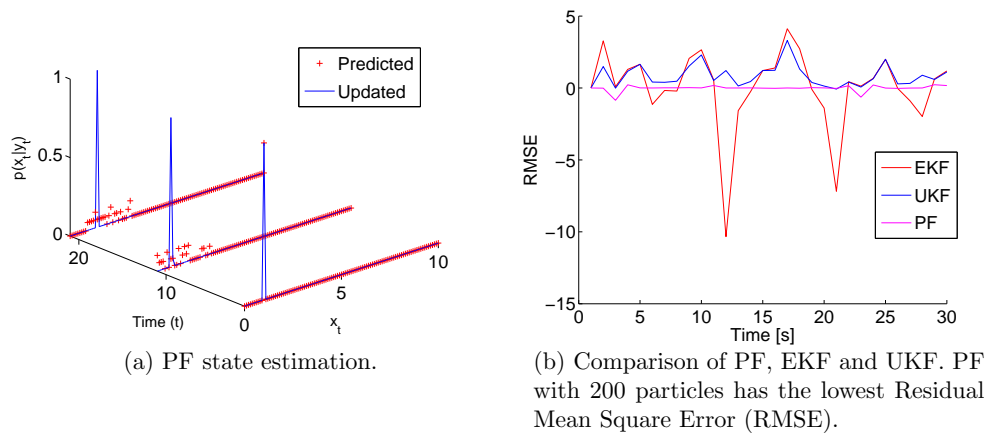


Figure 2.24: Particle filter state estimation and performance evaluation.

Motion and Structure Estimation by Fusion of Vision and Inertial Data

Humans can easily and rapidly move forward into a new environment while acquiring a model of it. As stated by e.g. Lobo in [61], the perceived retinal image of a scene is stabilised by a part of the human inner ear called the vestibular system. This vestibular system measures tilt and angular acceleration, and as with other biological systems, it has inspired the development of technical prototypes with similar functions commonly known as inertial sensors.

The inertial sensors, namely three orthogonal accelerometers and three orthogonal gyroscopes, are typically integrated in one Inertial Measurement Unit (IMU). The accelerometer measures translational acceleration, or in other words it senses the deviation from freefall. The development of this device can be tracked back to the work of Schuler in [87]. The gyroscope measures angular velocity, helping to maintain the orientation. The development of the first gyroscope prototype started in the beginning of the 20th century [24].

The rationale behind fusing vision and inertial data is their complementary characteristics. On one hand, a vision system provides a precise and long-term estimation of the ego-motion and its environment. However, it lacks the ability to cope with fast and unpredicted motions that may blur features in the image, or that may lead to failure to detect the features at all. This problem is particularly relevant when simultaneously estimating the pose and the structure. The structure estimation is the prerequisite to pose estimation and vice versa. On the other hand, inertial sensors provide good signals during rapid motions, but suffer from biased estimations, due to the dual integration performed to compute the pose from the acceleration readings.

While the complementarity of visual and inertial data suggests a combination of these two cues, the following problems need to be solved to provide a technical solution. First, any dynamics in the scene such as occlusions or foreground motion make it difficult to decide what is the fixed and unmovable structure or background. Second, the continuous change of illumination and appearance with viewpoint change, may lead

to a loss of *correspondence* of visual features. Fast camera motion poses a similar problem, especially rotation, which results in motion blur and makes feature matching difficult to nearly impossible. And third, technical solutions employ iterative algorithms to handle non-linear models.

The problems that arise due to occlusions or foreground motion in the scene can be tackled using different approaches. Firstly, a dense map of the environment can be recovered, where the assumption is that only a certain amount of features can be occluded at once. Secondly, a rather sparse map of the scene is reconstructed, and in the case of occlusions a camera pose reconstruction algorithm is employed. Thirdly, a sensor fusion approach is involved, where e.g. the visual data failure can be substituted temporarily with inertial measurements, which helps to estimate the ego-motion.

The problem caused by continuous change of illumination or fast camera motion is challenging in the computer vision field. Firstly, most computer vision algorithms assume that the illumination of the scene is fairly constant during the whole experiment. Secondly, the fast camera motion typically introduces motion blur, which can be generally avoided by executing only smooth slow movements.

This chapter is divided into six sections. A detailed overview of state-of-the-art is given in Sec. 3.1. The previous work and our contribution are outlined in Sec. 3.2. The principle of our IMU is explained in Sec. 3.3. Sec. 3.4 presents the motion and structure estimation algorithms. The experimental results are introduced in Sec. 3.5 and the conclusion in Sec. 3.6.

3.1 Related Work

In [5], one of the first papers tackling the sensor fusion problem is introduced, where a combination of a radar and an IMU is presented. As explained by Babister in [5], the advantage of this fusion is that the radar provides long-term stability and the IMU has little high-frequency noise. Several decades later a seminal paper, presented by Viéville and Faugeras [102], motivated to use the fusion of vision and inertial sensor in mobile robotics. In the work of Viéville and Faugeras, the first example of using an IMU on a robotic platform is mentioned, and a calibration of these inertial sensors is presented.

In the work of Lobo and Dias the advantages of using an IMU and a camera have been extensively studied. In [63], a pose estimation algorithm is proposed to integrate measurements from a stereo camera system and the IMU for mobile robots. Accelerometer signals have been low pass filtered to compute the gravity as a vertical reference. Further investigations are carried on in [1], where an algorithm is presented to estimate the intrinsic IMU parameters as scale, bias and axis alignment. Also to estimate the relative orientation between the single camera and the IMU. In [62], a robot equipped with a stereo camera and low-cost inertial sensors computes the *vertical reference*, fixates using an active stereo vision system on a ground plane, and obtains the three-dimensional model of it. In [64], a further extension of this work is presented, where using the estimated vertical reference stereo depth map can be aligned, and vertical or horizontal features can be extracted. Recently, Lobo summarised these papers in [61].

An interesting work presented by Diel et al. in [28] describes a vision-aided iner-

tial navigation without augmenting scene features directly. Instead *stochastic epipolar constraints* over a broad baseline in time and space using an omnidirectional camera have been proposed, which proved to have long-term stability. A probabilistic Epipolar Constraint filter (EPC), based on Kalman filter, performs the data fusion.

Corke introduced in [22] an architecture of a low-cost, light-weight inertial and visual sensing system for a small-scale autonomous helicopter. The vision system consists of a stereo camera, which provides measurements to compute features' correspondences. After these correspondences are found, optical flow is calculated, and a first-order complementary filter is involved to fuse the inertially and visually derived velocity information.

In [83], a rotation estimation algorithm is presented, where line correspondences and IMU data have been combined. The mathematical formulation of orientation is based directly on the group of rotation matrices, and simulated experimental results proving the convergence of the algorithm are provided.

Huster et al. presented several papers on estimating the relative position of an Autonomous Underground Vehicle (AUV) by fusing the vision and inertial measurements. In [47], a free floating vehicle capable of autonomous tasks in a two-dimensional world has been proposed. In [46], an extension of the previous algorithm to estimate relative position, velocity, attitude and inertial sensors' biases in a three-dimensional world has been introduced. In their paper, a modified Kalman filter has been used, where all non-linearities have been transferred to the state dynamics. In [48], the robustness of the algorithm has been successfully tested in the presence of significant process noise.

Klein and Murray in [55] introduced a tightly coupled inertial-vision tracking system. The authors used only three rate gyroscopes without considering accelerations. However, the angular velocities helped to predict the pose of known three-dimensional models, and this increases the system's robustness. For the sensor fusion a simple Kalman Filter is proposed.

Except for robotics the fusion of vision and inertial data has been an interesting topic in Augmented Reality (AR) as well. The AR community has been using this sensor fusion to build faster and more reliable trackers to overlay the three-dimensional information more precisely.

Naimark and Foxlin presented in [74] a prototype hybrid vision-inertial self-tracker capable of tracking artificial landmarks in indoor environments. As fiducial landmarks, two-dimensional barcodes have been used, and the paper discusses the robust localisation of these landmarks. Later the authors introduced in [37] a small, wearable and low-cost product having the same functions as the prototype.

Ribo et al. in [84] presented so called *hybrid* tracking for augmented reality applications, which is based on the fusion of vision and inertial information. Their system is built on the Dintree algorithm, which deals with multiple hypotheses simultaneously. Using the five best hypothesis a set of Kalman filters is repeatedly updated. A new object pose is computed with the help of the maximum likelihood scheme. To fuse current object pose with the inertial measurements Extended Kalman Filter (EKF) is applied. The experiments with a mobile AR kit prototype demonstrated successful tracking in simple outdoor environments.

In [13], a stereo camera combined with an IMU is used to track the head mo-

tion. Two parallel EKF filters update their states independently only when new measurements are available. The head motion filter is fusing information provided by accelerometers, gyroscopes and camera measurements. The structure filter computes the three-dimensional position of five natural landmarks.

In [112], a hybrid inertial and vision tracking approach for AR is proposed, where the complementary characteristic of both devices is tackled. This work is extended in [111], where 6DOF EKF with two separate processing channels is presented. These two processing channels process the high-speed gyroscope and low-rate vision measurements independently, which helps to compensate for each other weaknesses.

For an extended overview of the fusion of vision and inertial information please refer to the survey presented in [23].

3.2 Contribution Description

One of the first papers, which inspired our work is [111]. The difference with our work is that the EKF fusion filter is using vision and only gyroscopic measurements, while the goal is to design a more stable and robust tracking system. An interesting idea of their work is the independent updating of inertial or vision information.

Chroust and Vincze in [19] demonstrated an extension of this idea, where also accelerometers have been used. They concentrated their efforts on motion estimation, and the structure algorithm has been proposed only conceptually. Armesto et al. in [2] continued this research, and experimentally tested different types of motion filters.

Our work, recently summarised in [40], presented in this chapter continues with the afore mentioned efforts, but the goal is to develop a system capable of robust real-time motion estimation in an unknown environment. The structure estimation is realised as an independent algorithm, so our system consists of two loosely coupled statistical filters.

The contribution of our work is an extension of afore mentioned research, and we summarise it as follows:

- implementation of the work of Chroust and Vincze [19] and Armesto et al. [2],
- extending this implementation with a structure estimation algorithm and
- real-time experiments.

3.3 Inertial Measurement Unit

This section addresses the problem of position and orientation estimation using accelerometers and gyro inertial sensors. These inertial sensors provide acceleration and rates (attitude changes over a time interval - known as angular velocity). To provide three components of acceleration and rate measurements, accelerometers and gyros are typically mounted in orthogonal triad clusters enclosed within an IMU.

In many modern navigational systems the inertial sensors are attached rigidly to the body of the host vehicle. This rigid or “strapped down” configuration has several benefits: lower cost, light weight and greater reliability compared with equivalent platform system [99, 15].

The main disadvantage of an IMU is that the new position and orientation computation is based solely on the previous values. This disadvantage causes the estimation errors to cumulate. Estimating current position only from previous values is known as *dead-reckoning*.

To overcome this difficulty, IMU is usually combined with a sensor, which is able to perceive absolute position and orientation. Two additional interesting examples, to those presented in Sec. 3.1, where an IMU supports estimation of the position and orientation:

- in humanoid robotics in combination with a single camera [95] or
- in land vehicle applications using an additional GPS sensor [29].

3.3.1 Our Inertial Measurement Unit - MT9B

In the last 25 years, the demand in the automotive industry initiated an intensive research in Micro-Machined Electromechanical System (MEMS) sensors [99]. The reason was that these MEMS sensors have several advantages: small size, low weight, low power consumption, short start-up time, inexpensive to produce, etc. Except automotive industry are these MEMS features attractive in robotics, for example.

In this work we have been using MT9B unit (see Fig. 3.2(a)), which contains MEMS sensors. The output of these sensors includes:

- orthogonal triad accelerometer readings,
- orthogonal triad gyro measurements,
- output of the temperature sensor placed in the middle of the electronics and
- orthogonal triad magnetic field strength readings in atomic units (a.u.).

However, only the accelerometer and gyro measurements have been used in this work. A brief explanation of the MEMS accelerometer and gyro measurement principle:

- MEMS *accelerometer* detects the change in the capacitance gap (see Fig. 3.1(a)) between the proof mass and the substrate (out-of-plane), or measures the change in capacitance across the comb fingers (in-plane) [99]. MT9B has two in-plane accelerometers in the horizontal plane and one out-of-plane accelerometer in the vertical plane.
- MEMS *gyroscope* relies on detecting a force acting on a mass that is subject to linear vibratory motion. The resulted Coriolis force is measured. As depicted in Fig. 3.1(b), this Coriolis force is perpendicular to the axis of linear motion and input rotation [99].

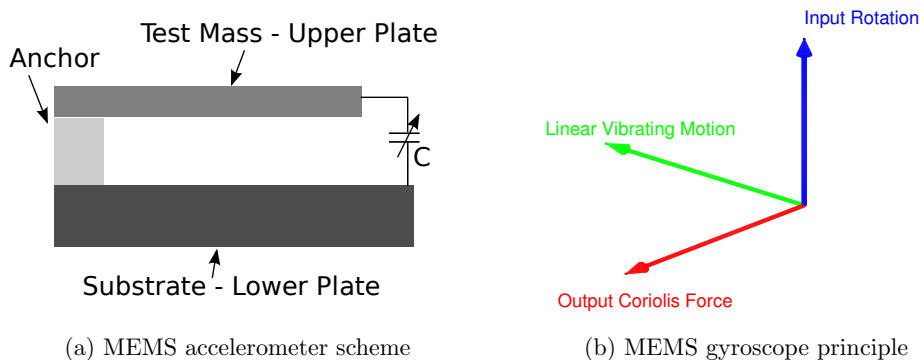


Figure 3.1: Basic construction scheme of an out-of-plane MEMS accelerometer, and the MEMS gyro measurement principle.

The manufacturer of MT9B [109] assumes that the enclosed accelerometers and gyros are calibrated, when these three constants are known: *bias*, *scale factor* and *misalignment*. The sensor model of MT9B IMU is formulated as:

$$\mathbf{s} = \mathbf{K}_T(\mathbf{u} - \mathbf{b}_T), \quad (3.1)$$

where the gain matrix \mathbf{K}_T contains the scale factor and misalignment elements. \mathbf{b}_T is the bias vector. The gain matrix \mathbf{K}_T and bias vector \mathbf{b}_T are the unknown calibration quantities. The calibration data is used to relate the sampled digital voltages, \mathbf{u} , (unsigned integers from the 16 bit Analog-Digital-Converter) streaming from the IMU to the respective physical quantity, \mathbf{s} [109].

We have been using a calibrated MT9B, where the \mathbf{K}_T and \mathbf{b}_T have been provided by the manufacturer.

	Accelerometer	Gyroscope
Full scale	$\pm 20 \frac{m}{s^2} (2g)$	$\pm 900 \frac{deg}{s^2}$
Bandwidth	50Hz	30Hz

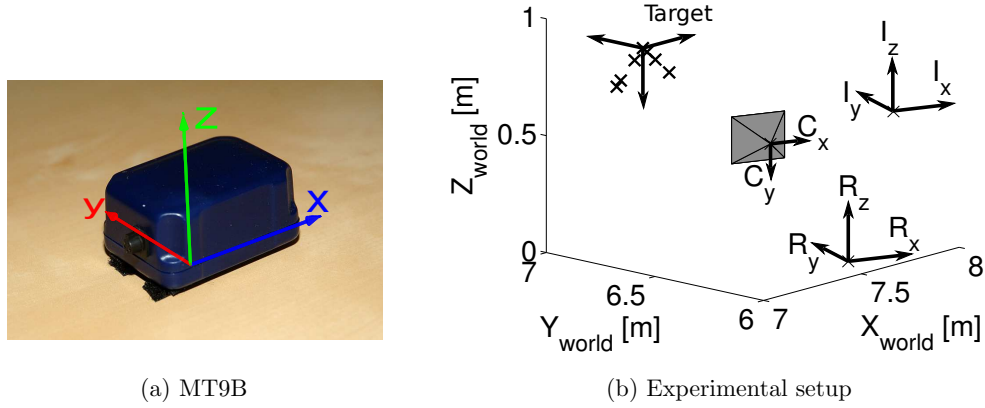
Table 3.1: This MT9B configuration has been used.

The manufacturer stated in [109] that:

- bias stability is 1 for accelerometer and 0.02 for gyro (in units per °C),
- scale factor is 0.2 for accelerometer and 0.03 for gyro (in % per °C) and
- misalignment is 0.1° for accelerometer and gyro.

We have been streaming MT9B output at 100Hz, but this does not mean that the sensors can sense at such a high sampling rate. The bandwidths of the IMU sensors have lower sampling rates, and a standard configuration of the manufacturer is tabled in Tab. 3.1.

In this thesis the MT9B IMU has been attached rigidly to the body of a single camera. The relation between the coordination frames of the IMU and the camera is



(a) MT9B

(b) Experimental setup

Figure 3.2: The experimental setup contains the robotic ("R") platform, the sensor rig and the initial target("T"). The sensor rig includes a MT9B IMU ("I") and a single camera ("C"). The seven initial crosses of the target object are additionally superimposed on the target frame.

calibrated [19]. An overview of the robot, target, IMU, and single camera coordination frames, is depicted in Fig. 3.2(b).

For further details about MT9B please refer to [109].

3.3.2 Example: Integration vs. Estimation of Attitude

As mentioned in Sec. 3.3, estimation of new attitude (θ_i) using a gyro is based solely on the last attitude (θ_{i-1}), and current angular velocity measurement (θ_m):

$$\theta_i = \theta_{i-1} + \int \dot{\theta}_m dt. \quad (3.2)$$

However, a more suitable way of computing the current attitude is *estimating*, e.g. using EKF (see Sec. 2.3.4), instead of *integrating* the last attitude. In the following example, we demonstrate how the non-linear probabilistic algorithm improves state estimation using noisy measurements.

The state $\hat{\mathbf{x}}$ of the EKF has two elements:

$$\hat{\mathbf{x}} = [\theta \quad \dot{\theta}_b], \quad (3.3)$$

where θ is the estimated angle, and $\dot{\theta}_b$ is the angular velocity bias.

An unbiased $\dot{\theta}$ is computed as:

$$\dot{\theta} = \dot{\theta}_m - \dot{\theta}_b. \quad (3.4)$$

The state prediction is computed as follows:

$$\hat{\mathbf{x}}(k|k-1) = \hat{\mathbf{x}}(k-1|k-1) + [\dot{\theta} dt \quad , \quad 0]. \quad (3.5)$$

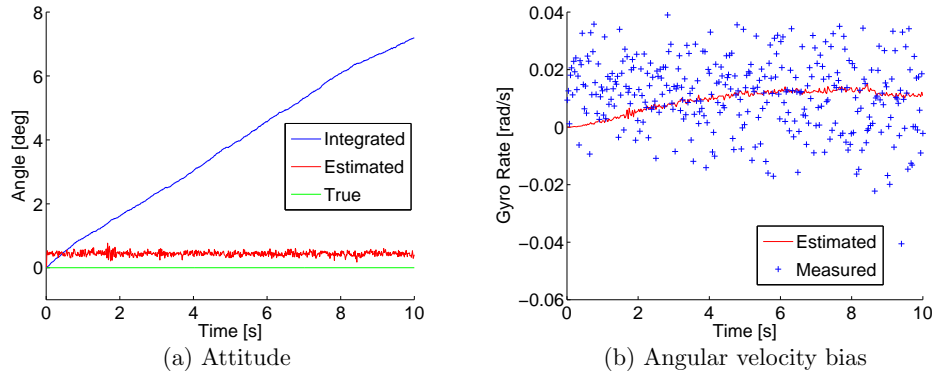


Figure 3.3: Left is the comparison of integration vs. estimation of attitude when no external force has been applied. Right is an additional overview of measured angular velocities and estimated angular velocity bias.

The system matrix equals:

$$\mathbf{F}(k-1) = \begin{bmatrix} \frac{\partial \dot{\theta}}{\partial \theta} & \frac{\partial \dot{\theta}}{\partial \theta_b} \\ \frac{\partial \dot{\alpha}}{\partial \theta} & \frac{\partial \dot{\alpha}}{\partial \theta_b} \end{bmatrix}, \quad (3.6)$$

where α is angular acceleration bias, and equals:

$$\alpha = \frac{\partial \dot{\theta}_b}{\partial t}. \quad (3.7)$$

The $\mathbf{F}(k-1)$ as a result of differentiating:

$$\mathbf{F}(k-1) = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \quad (3.8)$$

The input matrix:

$$\mathbf{H}(k) = [1 \ 0]. \quad (3.9)$$

The noise matrices $\mathbf{Q}(k)$ and $\mathbf{R}(k)$ have the following values:

$$\mathbf{Q}(k) = \begin{bmatrix} 0.003 & 0 \\ 0 & 0.001 \end{bmatrix}, \quad (3.10)$$

$$\mathbf{R}(k) = [0.00011]. \quad (3.11)$$

Additionally, this attitude estimation in x-direction is corrected using Y (a_y) and Z (a_z) axis accelerometer readings:

$$\mathbf{z}(k) = \tan^{-1}(a_z, a_y). \quad (3.12)$$

As depicted in Fig. 3.3(a), estimating attitude using EKF and accelerometer readings outperforms integration significantly. Another advantage of EKF is that it estimates additionally the angular velocity bias (see Fig. 3.3(b)).

A similar example of attitude estimation for mobile robots is presented in [8].

3.4 Motion and Structure Estimation

To explain the details of the structure and motion estimation, this section comprises of three parts. Firstly, the extraction of visual and inertial measurements is explained in Sec. 3.4.1. Secondly, the camera motion estimation is described in Sec. 3.4.2. Thirdly, the structure estimation algorithm is presented in Sec. 3.4.3.

Before introducing the structure and motion algorithms, these are the general assumptions under which we developed them:

- using monocular images only,
- accurate, well-calibrated sensors and
- known initial pose of the target.

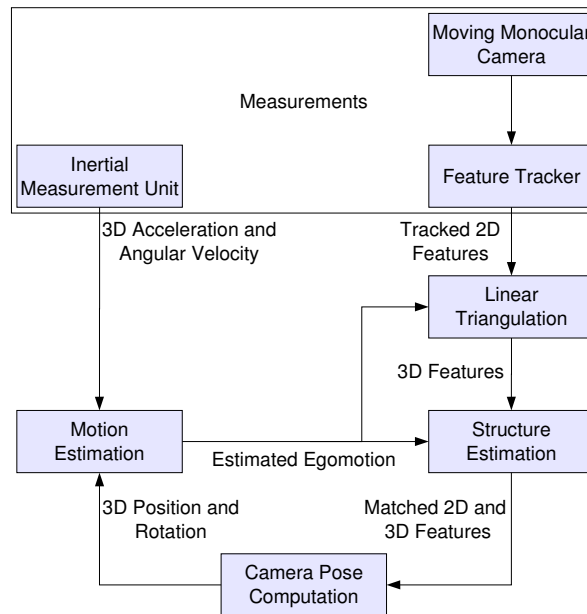


Figure 3.4: Flow chart for the motion and structure estimation.

3.4.1 Measurements

The basic idea of this chapter is to fuse vision and inertial information for structure and motion estimation. To perform this fusion we need to process visual images from a single camera, and read the output from an IMU (see Sec. 3.3).

3.4.1.1 Visual Measurements

The vision input comes from a moving monocular camera, where in each image frame corner features are detected. For extraction and tracking (see Fig. 3.4) of corners, we have been using these two very similar approaches:

1. For estimating new scene 3D landmarks in Sec. 3.4.3.1 we used a feature tracker implementation [114] based on the well-known Shi-Tomasi features (see Sec. 2.1.2). This tracker is a stand-alone module, which tracks features independently on the last estimated motion state.

For a good compromise of fast tracking and visibility from different viewpoints we detect features in 7x7 pixel patches. From a sequence of images the output are the 2D positions of the matched features. The correspondence errors of the tracked features are not found in the tracking step. If a correspondence error occurs it can be detected using the structure estimation procedure, and tracking of the erroneous feature is stopped (for details refer to Sec. 3.4.3).

2. In the experiments described in Sec. 3.5.2, we detect new scene features using the Harris corner detector (see Sec. 2.1.1). The problem of features matching between two consecutive frames is tackled with the fundamental matrix computation [45], and consequent RANSAC (see Sec. 2.1.6) robust fitting.

The extracted 2D features from at least two different images are used to obtain the 3D position of this feature by linear triangulation (see Sec. 2.2.7). The prerequisite for linear triangulation is to know the camera pose for each of the images used.

3.4.1.2 IMU Measurements

Inertial measurements angular velocity and linear acceleration have been used. Our IMU, which delivers acceleration in three orthogonal coordinate directions and the three angular velocities about these directions, is presented in Sec. 3.3.1.

3.4.2 Modelling Camera Motion

The input to this function block (see Fig. 3.4) are measurements from the IMU, and the pose estimate from the vision system. The measurements of the inertial system are not synchronized with the measurements of the vision system, and it is not assumed that both sensors have a fixed sampling rate. The prediction from one time frame to the next is performed with the same equation independent of the type of the measurement, but the dependencies between system state and measurement are totally different. The idea used in this work is based on the multi-rate Kalman filter motion estimation [19, 2]. Whenever there is a new measurement it updates either the inertial or visual estimates, and is then fused in motion state estimation.

3.4.2.1 Pose Calculation

As depicted in Fig. 3.4, camera pose calculation is the link between structure estimation and motion estimation: a given structure allows it to calculate the camera pose, from the camera pose the egomotion is estimated, and finally, due to motion new features are detected and added to the model in the structure estimation frame.

However, the pose calculation suffers from many errors. The most important impact on the pose calculation is the distortion (up to 10mm) introduced by the printer, when printing the image of the target. The known 3D points of the target are precise up to $1\mu\text{m}$, and it is very difficult to measure the printed version with such accuracy. The next error affecting the pose is the camera calibration [12]. The last source of pose evaluation errors is the sub-pixel accuracy of the corner feature extraction. To calculate the pose we use the algorithm described in Sec. 2.2.6, which proved to be sufficiently robust.

The calculated pose has been used only for the motion estimation. The estimated motion has not been used for the tracking of the features' 2D positions.

As initial target, an artificial planar object (see Fig. 2.20) with seven known features is chosen [14]. The correspondences of these features are given, that means the relationships between 3D world positions and 2D positions in the image plane are known. This object has properties, which are invariant to perspective transformation. It is assumed that in the first image frame of every sequence all seven features are visible.

3.4.2.2 Motion Model

The camera motion estimation in 3D space is treated as a non-linear dynamic state estimation problem, and this problem is described in Eq. 2.36.

The state vector $\mathbf{x}(k)$, the measurement vector $\mathbf{y}(k)$, the system noise $\nu(k)$ and the measurement noise $\mathbf{w}(k)$ are defined as follows:

$$\begin{aligned}\mathbf{x}(k) &= [\mathbf{p}^T \ \vec{v}^T \ \vec{a}^T \ \mathbf{b}^T \ \mathbf{q}^T \ \vec{\omega}^T]_k^T, \\ \mathbf{y}(k) &= [\vec{a}_m^T \ \vec{\omega}_m^T \ \mathbf{p}_m^T \ \mathbf{q}_m^T]_k^T, \\ \nu(k) &= [j^T \ \vec{\alpha}^T \ \mathbf{b}'^T]_k^T, \\ \mathbf{w}(k) &= [\vec{w}_{a_m}^T \ \vec{w}_{\omega_m}^T \ \vec{w}_{p_m}^T \ \vec{w}_{q_m}^T]_k^T,\end{aligned}$$

where the state vector is represented with positions $\mathbf{p}(k)$, velocities $\vec{v}(k)$, accelerations $\vec{a}(k)$, biases of the acceleration measurements $\mathbf{b}(k)$, quaternions (see Sec. 2.2.2) $\mathbf{q}(k)$, and angular velocities $\vec{\omega}(k)$. The measurement vector is formed with acceleration $\vec{a}_m(k)$ and angular velocity $\vec{\omega}_m(k)$ measurements from the IMU and position $\mathbf{p}_m(k)$, and quaternion $\mathbf{q}_m(k)$ measurements from the vision system. The system noise is composed of: jerks $\vec{j}(k)$, angular accelerations $\vec{\alpha}(k)$ and velocity biases $\mathbf{b}'(k)$.

The non-linear dynamic Eq. 2.36 can be separated in two parts:

$$\mathbf{x}_t(k) = f_t[k-1, \mathbf{x}(k-1), \mathbf{u}(k-1)] + \nu(k-1), \quad (3.13)$$

$$\mathbf{x}_r(k) = f_r[k-1, \mathbf{x}(k-1), \mathbf{u}(k-1)] + \nu(k-1). \quad (3.14)$$

The translational part of the estimated state vector $\hat{\mathbf{x}}(k)$ equals:

$$\hat{\mathbf{x}}_t(k) = [\mathbf{p}^T \ \bar{\mathbf{v}}^T \ \bar{\mathbf{a}}^T \ \mathbf{b}^T]_k^T, \quad (3.15)$$

and the rotational part:

$$\hat{\mathbf{x}}_r(k) = [\mathbf{q}^T \ \bar{\boldsymbol{\omega}}^T]_k^T. \quad (3.16)$$

The translational dynamic equations have the form:

$$\mathbf{p}(k) = \mathbf{p}(k-1) + T \bar{\mathbf{v}}(k-1) + \frac{T^2}{2} \bar{\mathbf{a}}(k-1) + \frac{T^3}{6} \bar{\mathbf{j}}(k-1), \quad (3.17)$$

$$\bar{\mathbf{v}}(k) = \bar{\mathbf{v}}(k-1) + T \bar{\mathbf{a}}(k-1) + \frac{T^2}{2} \bar{\mathbf{j}}(k-1), \quad (3.18)$$

$$\begin{aligned} \bar{\mathbf{a}}(k) &= \bar{\mathbf{a}}(k-1) + T \bar{\mathbf{a}}'(k-1) = \\ &= \bar{\mathbf{a}}(k-1) + T(\bar{\mathbf{j}}(k-1) + \bar{\boldsymbol{\alpha}}(k-1) \times \bar{\mathbf{v}}(k-1)) + \\ &\quad + T(\bar{\boldsymbol{\omega}}(k-1) \times \bar{\mathbf{a}}(k-1)), \end{aligned} \quad (3.19)$$

$$\mathbf{b}(k) = \mathbf{b}(k-1) + T \mathbf{b}'(k-1), \quad (3.20)$$

where acceleration consists of two parts:

$$\bar{\mathbf{a}}(k) = \bar{\mathbf{a}}_t(k) + \bar{\boldsymbol{\omega}}(k) \times \bar{\mathbf{v}}(k), \quad (3.21)$$

where $\bar{\mathbf{a}}_t(k)$ is the tangential and $\bar{\boldsymbol{\omega}}(k) \times \bar{\mathbf{v}}(k)$ is the centripetal acceleration.

The translational part of the state vector $\hat{\mathbf{x}}(k)$ is composed of Eq. 3.17-3.20 and can be computed as:

$$\begin{aligned} \hat{\mathbf{x}}_t(k) &= \begin{bmatrix} \mathbf{I}_{3 \times 3} & T\mathbf{I}_{3 \times 3} & \frac{T^2}{2}\mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & T\mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \bar{\mathbf{v}} \\ \bar{\mathbf{a}} \\ \mathbf{b} \end{bmatrix}_{k-1} + \\ &\quad + \begin{bmatrix} \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \\ T \bar{\boldsymbol{\omega}} \times \bar{\mathbf{a}} \\ \mathbf{0}_{3 \times 3} \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{T^3}{6} \mathbf{I}_{3 \times 3} \\ \frac{T^2}{2} \mathbf{I}_{3 \times 3} \\ T \mathbf{I}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \end{bmatrix} \bar{\mathbf{j}}(k-1) + \\ &\quad + \begin{bmatrix} \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \\ T \bar{\boldsymbol{\alpha}} \times \bar{\mathbf{v}} \\ \mathbf{0}_{3 \times 3} \end{bmatrix}_{k-1} + \begin{bmatrix} \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} \\ T \mathbf{I}_{3 \times 3} \end{bmatrix} \mathbf{b}'(k-1). \end{aligned} \quad (3.22)$$

The rotational dynamic equations have the form:

$$\begin{aligned} \mathbf{q}(k) &= \mathbf{q}(k|k-1) \otimes \mathbf{q}(k-1) \\ &= \exp\left(\frac{\Delta\theta}{2}\right) \otimes \mathbf{q}(k), \end{aligned} \quad (3.23)$$

$$\bar{\boldsymbol{\omega}}(k) = \mathbf{I}_{3 \times 3} \bar{\boldsymbol{\omega}}(k-1) + T \mathbf{I}_{3 \times 3} \bar{\boldsymbol{\alpha}}(k-1), \quad (3.24)$$

where Eq. 3.23 consider that the rotation is composed of the angular velocity and acceleration:

$$\Delta\theta = \vec{\omega} T + \frac{1}{2} \vec{\alpha} T^2 . \quad (3.25)$$

The multiplication of two quaternions $(\vec{\alpha}, \beta)$, represented by \otimes , is defined as in [18]

$$\begin{aligned} \tilde{\alpha} \otimes \beta &= (a_0 + a) \otimes (b_0 + b) \\ &= a_0 b_0 + a_0 b + b_0 a - ab + a \times b , \end{aligned}$$

where each quaternion has a scalar (e.g. a_0) and a vector (e.g. a) part.

The Eq. 3.23 can be expanded as mentioned in [101]:

$$\begin{aligned} \mathbf{q}(k) &= \mathbf{q}(k-1) \cos\left(\frac{\|\Delta\theta\|}{2}\right) + \Omega(q) \vec{\omega}(k) T \frac{\sin\left(\frac{\|\Delta\theta\|}{2}\right)}{\|\Delta\theta\|} + \\ &\quad + \Omega(q) \vec{\alpha}(k) \frac{T^2}{4} \sin\left(\frac{\|\Delta\theta\|}{2}\right) , \end{aligned} \quad (3.26)$$

where

$$\Omega(q) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix} .$$

3.4.2.3 Multi-Rate Filters

In our application, the vision system is sampled at approximately 25Hz and the IMU is sampled at 100Hz in order to sense fast motions. Therefore multi-rate fusion techniques are required to improve the estimations, and to exploit the complementary characteristic properties of each sensor.

The main drawback of EKF (see Sec. 2.3.4) is the need of linearisation, which can produce unstable filter performance and is usually non-trivial. Therefore Julier and Uhlmann [51] decided to approximate a Gaussian distribution, rather than to approximate an arbitrary non-linear function. They introduced UKF, for more details please refer to Sec. 2.3.5.

The multi-rate explanation of EKF and UKF can be found in [2]. The idea behind the multi-rate fusion is to build up a size-varying measurement vector, containing only those measurements that have been sampled at each time instant. In this sense, the measurement vector is given by:

$$\begin{aligned} \mathbf{y}(k) &= [\vec{a}_m^T \quad \vec{\omega}_m^T \quad \mathbf{p}_m^T \quad \mathbf{q}_m^T]^T \text{ iff data from both sensors are sampled,} \\ &= [\vec{a}_m^T \quad \vec{\omega}_m^T]^T \text{ iff inertial data are available,} \\ &= [\mathbf{p}_m^T \quad \mathbf{q}_m^T]^T \text{ iff vision data are available,} \end{aligned} \quad (3.27)$$

where no data are sampled, then an update step can not be executed and only the predicted pose can be used for the next iteration.

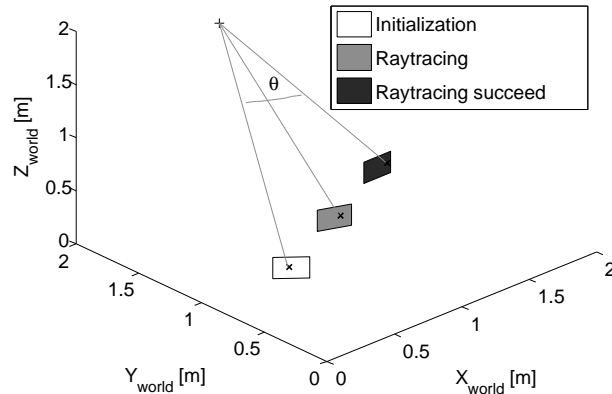


Figure 3.5: For two 2D vision measurements, rays from the image plane are projected in the 3D space (raytracing) and the view angle (θ) between them is measured. If the view angle is satisfactory, the algorithm proceeds.

3.4.3 Mapping Scene Structure

To present the mapping of the scene structure, this section comprises three parts. Firstly, the structure estimation algorithm is introduced. Secondly, the pose-recovery algorithm is explained. Thirdly, updating of the scene model is presented.

3.4.3.1 Structure Estimation Algorithm

The goal of the structure estimation algorithm (see Alg. 3.1) is to estimate the natural landmarks in 3D space. These natural landmarks represent the 3D positions of the extracted corner features (see Sec. 3.4.1.1). To achieve this goal, every natural landmark is estimated in two steps. The first step is the triangulation from several images, which delivers the initial 3D position. In the second step, the landmark's initial 3D position is further estimated with the EKF (see Sec. 2.3.4).

To estimate the 3D position of a new feature in world coordinates the pose of the camera in world coordinates must be estimated (see Sec. 3.4.2.1). The next step is the calculation of the 3D position of each single feature. At least two measurements in successive images from different viewpoints are necessary to calculate the 3D position using triangulation (see Sec. 2.2.7). Finally, to estimate the position of a feature more reliably, a Kalman filter (see Sec. 2.3.2) is used to smooth the calculation. Each new feature gets its own EKF to keep the computational burden small. The multiple instances of the same EKF for each corner feature are called a *filter bank*. If the position estimation of a feature is reliable, then it is inserted in the model of the scene to enlarge the model for an increase of the robustness of the pose estimation. The overall procedure is outlined in Alg. 3.1.

In this work, it is assumed that at the beginning of the experiment, the 3D positions

```

Require: 2D feature is found
Ensure: EKF for this feature is uninitialized
1: while feature is visible do
2:   track this feature
3:   if feature is not in scene model then
4:     if EKF not initialized then
5:       calculate the view angle (see Fig. 3.5)
6:       if view angle > threshold (see Sec. 3.4.3.3) then
7:         linear triangulation for this feature
8:         initialize EKF for this feature
9:       end if
10:    else
11:      update the state with EKF
12:      if EKF converging (see Sec. 3.4.3.3) then
13:        if baseline shift > threshold (see Sec. 3.4.3.3) then
14:          insert this feature to the scene model
15:        end if
16:      end if
17:      if iterations > threshold (see Sec. 3.4.3.3) then
18:        terminate feature estimation
19:      end if
20:    end if
21:    else
22:      if covariance < threshold (see Sec. 3.4.3.3) then
23:        update the state with EKF
24:      end if
25:      use this feature for pose estimation
26:    end if
27: end while

```

Algorithm 3.1: Structure estimation pseudocode.

of the seven points of the initial target (see Fig. 2.20) are known. The calculation of the 3D position of a new feature point can be done if at least two 2D measures are available, together with the camera pose of both measures. Then, for two 2D vision measurements, rays from the image plane are projected in the 3D space (raytracing - see Fig. 3.5) and the view angle between them is calculated. If the view angle is satisfactory, the algorithm proceeds. Next, the calculation of the 3D position is evaluated and it is based on the linear triangulation (see Sec. 2.2.7). It is important for a good triangulation to have a large baseline, which is certainly not true for frame-to-frame measurements, but can be achieved over time by employing a triangulating from multiple image frames. The calculation of this baseline threshold is given in Sec. 3.4.3.3. The output of the triangulation is then used as an initial estimate for EKF.

The EKF is used to continually smooth the 3D position of a feature. Smoothing is necessary to cope with the inaccuracies of the camera pose computation and inaccurate

feature detection. The accuracy of camera pose computation depends on the features that are used. If using only the seven corner features of the initializing mask, the pose calculation is accurate. Any new feature has a corresponding uncertainty when added to the model, and, hence, pose calculation might slightly jump if a new feature is added. As a consequence, the EKF smoothing helps to reduce this effect, which improves the filter update by using the known 3D features and the new 2D feature measurements. The accuracy of the feature measurement for the 2D corners is about one pixel and can only be eliminated in the image using computationally expensive sub-pixel fitting. Hence, the EKF smoothing is welcome and serves the same purpose as explained above.

The non-linear state-space description for the filter process can be described as in Sec. 2.3.4, where $\mathbf{x}(k)$ describes the state of the process, in this case the 3D position x , y and z , the measurements in the camera image plane (u, v) , f is the dynamic description of the system, $\nu(k-1)$ is the process noise, h is the mapping of the state to the output and $\mathbf{w}(k)$ is the measurement noise.

3.4.3.2 Pose-Recovery

When due to the fast motion no corners are extracted from an image or no correspondence can be found between consecutive frames or RANSAC (see Sec. 2.1.6) algorithm failed because of insufficient count of putative matches, then this pose-recovery algorithm is used:

- 1: Back-project known 3D model points with the last estimated pose on the image plane.
- 2: Search for corners in threshold-bounded windows around the back-projected (ref Sec. 2.2.4.3) points.
- 3: If no corners are found, continue to estimate pose from inertial data, and go back to the first step.

Algorithm 3.2: Pseudocode of the pose-recovery algorithm.

3.4.3.3 Model Update

As mentioned in Alg. 3.1, there are four key decisions for updating the model of the environment.

Raytracing Threshold

First a raytracing threshold is introduced. In this work, only one camera has been used, and there is no previous knowledge about the baseline of two consecutive frames. Therefore, every extracted feature from the image plane is treated separately. The raytracing (see Fig. 3.5) threshold is represented by a view angle. This view angle is calculated from two frames. The first frame is the one, where the feature appears for the first time, and then every next frame is taken into account until a pre-defined view angle is reached.

EKF Convergence Thresholds

The next threshold is the EKF convergence condition, which is represented by three thresholds. First is the Euclidean distance between last and previous state estimates ($\mathbf{x}(k)$) in the feature 3D position. The second threshold is the norm of the 3×3 covariance matrix, which contains the information about the uncertainty of the feature in 3D space. The third is a condition, which stops, when a maximum count of iterations (across multiple images) has been reached. It means that the initial state is inaccurate or the filter is diverging. In this case, the estimation of this feature starts again from raytracing.

Baseline Threshold

Before inserting the newly estimated 3D feature into the model of the scene, the motion drift is evaluated. A pose hypothesis is computed from the last model of the scene with the new feature inserted. If the Euclidean norm of the difference between the pose hypothesis and the last estimated pose is above a threshold, then return to raytracing. Otherwise, insert the new feature (3D position) into the model of the environment. This procedure assures that features not fitting to the model are rejected. The reason can be that the baseline was too short for a good 3D estimate or correspondence in the tracking step failed.

Smoothing Condition

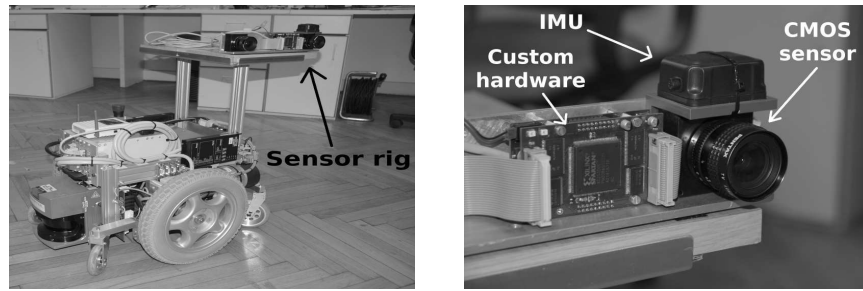
The structure estimation is correlated with the pose computation (see Fig. 3.4), and once a new feature has been inserted into the model of the scene, it is used for subsequent pose calculation. To enable a more accurate pose computation, all features (except initial - see Fig. 2.20) in the model are estimated as long as the norm of the feature 3D position uncertainty (3×3 covariance matrix) lies above a threshold. If the norm of the feature 3D position uncertainty is below the threshold, the estimation process stops.

3.5 Experimental Results

The presented motion and structure estimation algorithms have been tested using two different experimental setups. The first experiment tested the robustness and performance of our algorithms when the sensor rig was mounted on a custom mobile platform. The second experiment aimed at testing the sensor fusion complementary characteristic and accuracy of the proposed motion and structure approaches, where the camera and IMU were attached to an industrial robotic arm.

3.5.1 Experiments Using a Mobile Platform

A set of offline experiments has been devised to demonstrate the performance and robustness of our motion and structure algorithms (see Sec. 3.4) [40, 42]. These algorithms operate with calibrated sensors where the intrinsic parameters are assumed known. The comparison with the true state is introduced for ego-motion, and also for structure of the scene. The true motion state has been compared with the robotic platform poses, and the correctness of 3D feature positions has been evaluated in comparison to their true distance in the 3D scene.



(a) The robotic platform from Blue-Botics with mounted sensor rig.

(b) Sensor rig consists of the Inertial Measurement Unit (IMU), two CMOS vision sensors (only one of them is being used) and the custom hardware.

Figure 3.6: Sensor rig mounted on a mobile platform.

The assumptions under which these experiments have been conducted are:

- well-calibrated sensors,
- known initial object (see Fig. 2.20) and
- constant lightning conditions.

To detect features in the image sequence, an implementation of the Kanade-Lucas-Tomasi feature tracker [114] has been used. The details about this tracker are explained in Sec. 3.4.1.1.

3.5.1.1 Experimental Setup

As depicted in Fig. 3.2(b), the coordinate frames of this experimental setup consists of: the robotic platform (see Fig. 3.6(a)), one CMOS vision sensor and a XSens MT9B IMU (see Fig. 3.6(b)). These sensors are mounted on the robotic platform, which can perform only planar motions.

When calibrating the camera, the radial and tangential lens distortions were considered (see Sec. 2.2.4). The calibration software used was introduced by Bouguet in [12]. For more details about the MT9B calibration, please refer to Sec. 3.3.1.

The robotic platform that has been used in the experiments is a prototype of the MOVEMENT [71] system's Mobile Platform, designed and manufactured by Blue-Botics. It features a differential drive with two driven wheels in the middle, two castor wheels in the back and two castor wheels on springs in the front. The platform's sensor system comprises wheel encoders (odometry) and a SICK laser ranger in the front. There are two modes of motion control: avoid mode and bump mode. Avoid mode is intended for autonomous motion of the platform to a destination specified by the user. Using its laser ranger and a map, the platform continuously corrects its odometry data and avoids obstacles. However, in this mode the motion (i.e. the speed) is entirely controlled by the platform's onboard navigation algorithm in response to the sensor data. In bump mode the user can specify the translation and rotation speed of the

platform, but with the downside that its onboard controller makes no use of the laser ranger, neither for obstacle avoidance nor for correcting the odometry data. In our experiments all motions have been controlled in bump mode.

3.5.1.2 Motion Evaluation

The robotic platform performed motions including translation and rotation as given in Tab. 3.2. The translation to the target includes changing the scale of the initial object and changing the velocity. The rotation includes: loss of the initial target and the system moves into a completely new environment. Each of the motion runs is about 18s long. For longer runs in an unknown environment more new landmarks are assumed to be inserted in the map of the scene. These new landmarks have larger uncertainties in their 3D positions, and due to this the accuracy in pose computation is decreasing. This results in increased motion drift.

The offline performance of two multi-rate motion filters (EKF and UKF) is depicted in Fig. 3.7 and in Fig. 3.8. When the model of the scene has been updated by adding new landmarks, we refer to it as 'with updated model', while adding no new features is called 'without updated model'. The mean square errors over the complete motions of EKF/UKF to ground truth are given in Tabs. 3.3 and 3.4. The Fig. 3.7 and Fig. 3.8 indicate that without an update, model uncertainty increases the further the motion moves away from the initial pose (compare the increase of p_y and q_x). When returning to the initial position, accuracy is similar for both cases. The graphs also indicate that the performance of EKF and UKF are very similar.

Type of motion	Translation		Rotation	
	length [m]	average velocity [m/s]	length [°]	velocity [°/s]
translation + rotation	1.13	0.1	12.72	2.70
	1.28	0.1/0.2	10.54	2.19
	1.11	0.1	54.16	11.32

Table 3.2: Performed motion patterns.

	RMSE in position [m]		
	x	y	z
EKF	0.0409	0.0100	0.0242
UKF	0.0603	0.0065	0.0177

Table 3.3: Residual mean square error (RMSE) in position estimation when comparing the motion states of EKF and UKF with the ground truths.

In all experiments, the process noise ($\mathbf{Q}(k-1)$) and the measurement noise ($\mathbf{R}(k)$) have the same values for the multi-rate EKF and UKF. To find the optimal values for these noise matrices Armesto et al. [3] performed different movements with various

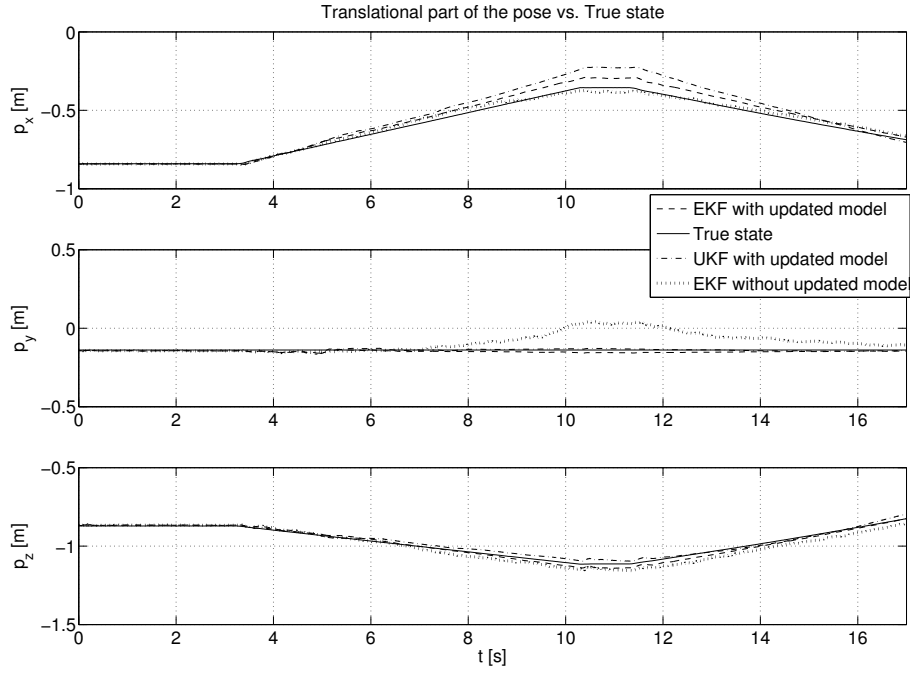


Figure 3.7: The translational part of motion estimation using multi-rate EKF and UKF.

	RMSE in orientation			
	q_0	q_x	q_y	q_z
EKF	0.0033	0.0175	0.0124	0.0147
UKF	0.0057	0.0170	0.0205	0.0133

Table 3.4: Residual mean square error (RMSE) in orientation estimation when comparing the motion states of EKF and UKF with the ground truths.

velocities, and the error in motion estimation using EKF or UKF has been analyzed. The values obtained are

$$\mathbf{Q}(k-1) = \begin{bmatrix} 7.447 \cdot 10^{-1} \mathbf{I}_{3 \times 3} & & & \\ & 3.8 \cdot 10^{-1} \mathbf{I}_{3 \times 3} & & \\ & & 1.9 \cdot 10^{-7} \mathbf{I}_{3 \times 3} & \\ & & & \end{bmatrix},$$

$$\mathbf{R}(k) = \begin{bmatrix} 1.0 \cdot 10^{-3} \mathbf{I}_{3 \times 3} & & & \\ & 1.0 \cdot 10^{-4} \mathbf{I}_{3 \times 3} & & \\ & & 1.0 \cdot 10^{-7} \mathbf{I}_{3 \times 3} & \\ & & & 1.0 \cdot 10^{-6} \mathbf{I}_{4 \times 4} \end{bmatrix}.$$

The benefit of the fusion of inertial and vision data is that the inertial sensors provide more information about rapid motions, but the drawback is that they typically

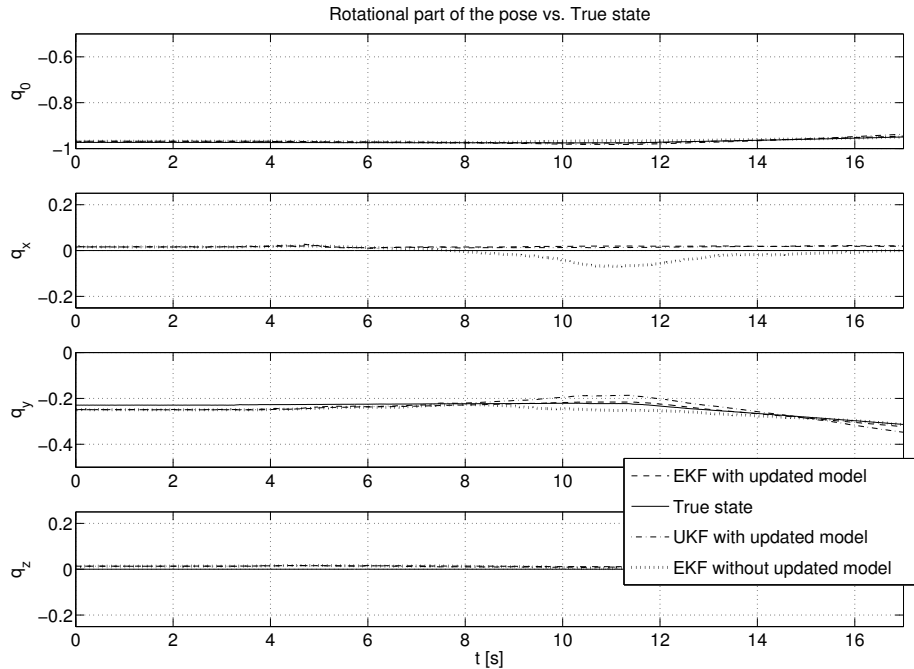


Figure 3.8: The rotational part of motion estimation using multi-rate EKF and UKF.

suffer from drift. In this work, the calibrated XSens MT9B [110] IMU has been used, and the gyroscope drift rate of this IMU is up to $60^\circ/\text{h}$. When estimating the position only from acceleration readings a bias of 0.1m/s has been measured [19]. However, the drift rate from gyroscope and the bias from accelerometer are compensated by EKF or UKF using the vision data.

3.5.1.3 Structure Evaluation

The structure of the unknown environment is measured by hand, and these measurements are taken as the true states for the estimated 3D positions. The comparison of some rigid 3D features is tabled in Tab. 3.5. From the table it is clear that features closer to the camera image plane are more accurate than more distant ones (e.g. ID 66). Some of the tabled features are highlighted on the left side in Fig. 3.9. The possible danger for long-term tracking are features like the one with ID 78, because they are not true 3D points. These outliers in corner tracking can slide in each image frame for several reasons (moving objects, occlusion boundaries, specular reflections, etc.). If the model has been updated with a non existing 3D feature, then it can cause a motion drift.

Fig. 3.10 depicts all estimated 3D positions of features with superimposed uncertainties (ellipsoids). The uncertainty of the feature mainly depends on its distance from the camera, duration in the model (due to the smoothing process) and if the feature is stable in 3D space.

ID	$ X_x - \bar{X}_x $	$ X_y - \bar{X}_y $	$ X_z - \bar{X}_z $	$\ X - \bar{X}\ $
52	0.025	0.114	0.006	0.117
53	0.015	0.020	0.011	0.027
59	0.027	0.104	0.013	0.108
66	0.726	0.794	0.553	1.210
107	0.155	0.131	0.009	0.203
108	0.159	0.135	0.017	0.209
109	0.743	0.015	0.015	0.743
Σ	1.849	1.312	0.624	

Table 3.5: Errors in structure estimation, where 3D points have been measured and estimated in the world coordinate frame in [m]. First item in the header row is the identification number of a feature (ID). Other items represent Euclidean norms between true values (X) and estimated 3D positions (\bar{X}). The corresponding image points are depicted in Fig. 3.9.

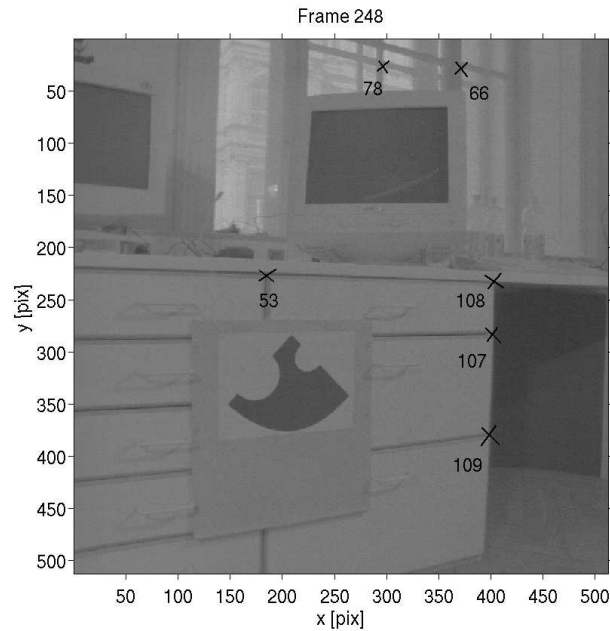


Figure 3.9: This image frame depicts only those 2D features, whose corresponding 3D positions can be initialised using the raytracing procedure (see Sec. 3.4.3.3).

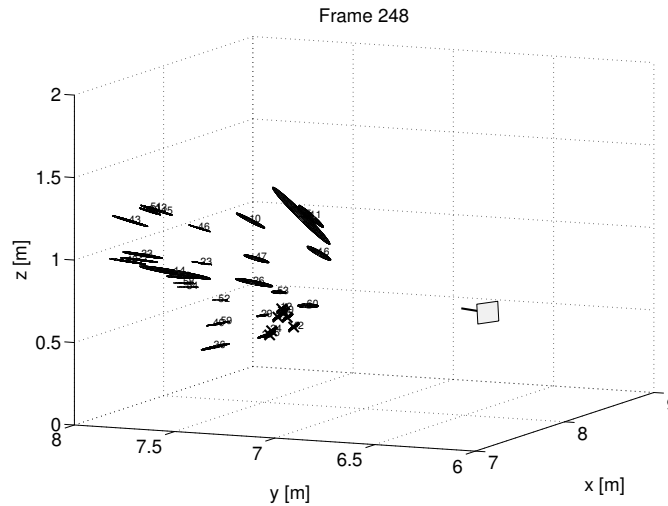


Figure 3.10: This is the virtual 3D space with all estimated 3D positions of features found up to the current frame. The uncertainties are represented by superimposed ellipsoids, and the camera coordinate frame is plotted additionally.

The thresholds for updating the model of the environment are as follows:

- Raytracing threshold (see Sec. 3.4.3.3) - The angle between two frames has to be greater than 1° .
- EKF convergence threshold (see Sec. 3.4.3.3) - The Euclidean distance between the last and previous estimate is expected to be less than $1.0 \cdot 10^{-2}$. The norm of the covariance matrix has to be below $1.0 \cdot 10^{-1}$. The maximum count of iterations is set to 75.
- Pose discrepancy (see Sec. 3.4.3.3) - The pose difference with and without the new estimated feature has to be below $1.0 \cdot 10^{-2}$ in translation (Euclidean distance between translation vectors) and in rotation (absolute difference in RPY angles) it can not be more than $3.0 \cdot 10^{-1}$.

The system can also move into an unknown environment performing a faster rotation ($11.32^\circ/\text{s}$ - third row of Tab. 3.2). The last image frame from this sequence is depicted in Fig. 3.11. The results of the pose estimation are similar to Fig. 3.7 and Fig. 3.8, and the error is once again similar as it was for the slower motions. The performance of EKF and UKF have been found to be similar. Further experiments with higher velocities need to be conducted, but require other hardware to obtain ground truth.

3.5.1.4 Run-time Evaluation

On a 3.0GHz Pentium processor, the minimum processing time needed for simultaneous motion and structure estimation using vision and inertial data is typically about 45ms. For inertial measurements only, about 5ms are needed. The performance of the motion estimation using EKF is about ten times faster than using UKF. This difference in

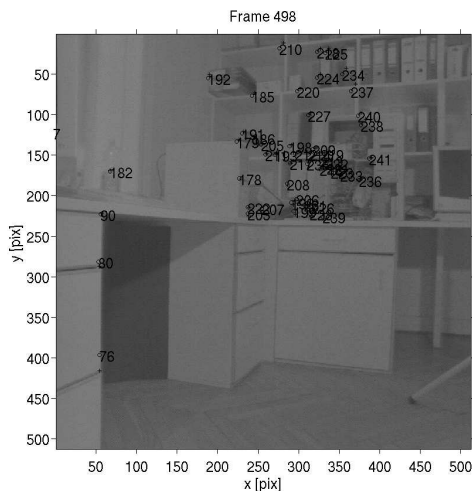


Figure 3.11: This is the last image frame of the first motion from Table 3.2. The initial artificial object is out of view, and the pose calculation is based only on new features. The back-projected 3D feature positions are depicted as circles, and the corner measurements are displayed as crosses.

Image loading (512×512 pixels at 25Hz)	2ms
Corner tracking (see Sec. 2.1.2)	10ms
Pose calculation (see Sec. 2.2.6)	10-15ms
Acquiring inertial data	1ms
Motion estimation (UKF - see Section 2.3.4 and Sec. 3.4.2.3)	2ms
Motion estimation (EKF - see Section 2.3.5 and Sec. 3.4.2.3)	0.2ms
Structure estimation (see Sec. 3.4.3)	10ms
Visualization	5ms
Total	45ms

Table 3.6: Computational time for the steps of simultaneous motion and structure estimation.

performance between EKF and UKF is similar to the one presented in [94]. The time needed in each step is given in Tab. 3.6.

3.5.2 Pose-Recovery Experiments Using a Robotic Arm

In these experiments, we performed fast motions with the KUKA KR/15 robot to recapture the features from the inertial data, and in Sec. 3.4.3.2 we call this pose-recovery procedure [38]. The estimation of the motion can be compared with the true state, because the sensor head is attached to a robotic arm whose known motion provides ground truth. The involved motion and structure algorithms are explained in Sec. 3.4. The same assumptions as presented in Sec. 3.5.1 are valid for experiments with the robotic arm.

In these experiments, image features have been detected using the Harris corner

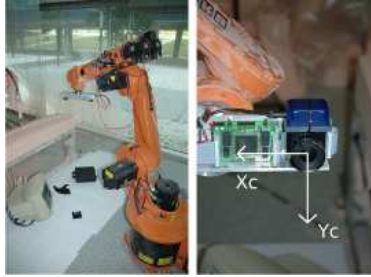


Figure 3.12: Left is the KUKA KR15/2 robot with the mounted sensor rig. Right is the vision sensor with the superimposed Cartesian coordinate system. The blue box on the top of the camera is the IMU unit.

detector. RANSAC has been employed for robust matching of two feature sets between consecutive frames. The feature extraction is explained in Sec. 3.4.1.1.

3.5.2.1 Experimental Setup

This section describes the simulation results with the architecture hardware, as depicted in Fig. 3.12. The custom CMOS camera (developed by the project partner EMT, TU-Graz - [73]) contains: two Fuga1000 1024×1024 vision sensors from FillFactory (used in a 512×512 subsampled mode) and an IMU unit from XSens (MT9-B). This sensor rig is mounted on a KUKA KR15/2 robot.

3.5.2.2 Motion Evaluation

Vision typically fails when in fast motion, so pose-recovery has been tested on a parallel translatory motion pattern with four different vision failure situations. The robot was moving from the first position along the world y -axis (the world y -axis is aligned with camera y -axis - see Fig. 3.12) to the final position and then back, during the beginning of the return an acceleration was introduced. The desired velocity for losing vision measurements with CMOS camera running at 20Hz from approximately 1m distance to the object is about 2m/s, but the problem with the robot acceleration is that it needs some distance to achieve 2m/s. The minimum distance for losing vision frames was empirically set to 0.1m and then 0.15m, 0.2m and 0.3m distances were tested (see Fig. 3.13).

Every tested motion pattern has been a translation along the y -axis. In Fig. 3.13 is the overall comparison of all motions. On the left side is the ideal motion pattern and on the right side is the comparison of all motions with the estimated y -translational part.

The average vision failure distance (0.2m) is depicted in Fig. 3.14. Due to the fast motion for three frames the feature extraction failed.

In figures 3.13 and 3.14 the pose calculation is precise enough to recover the pose.

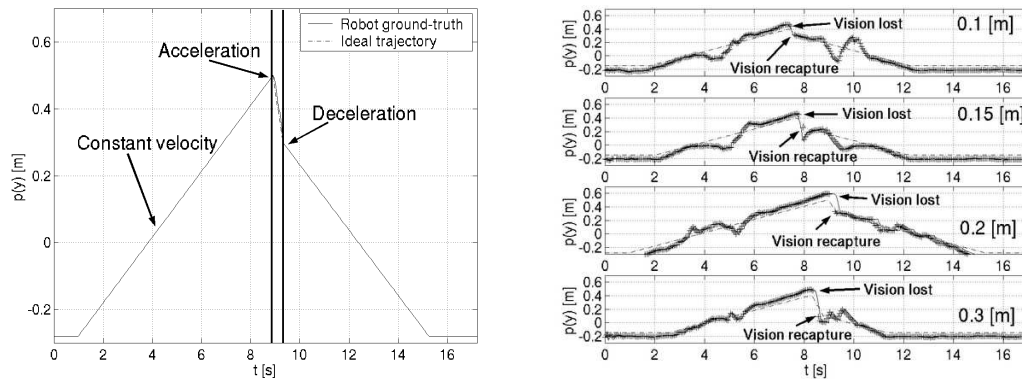


Figure 3.13: Left is the ideal motion pattern, while focused on the y -translational part of the pose. Right are the different vision failure situations. The estimated pose is plotted with a solid line, while ground-truth is displayed with dash dotted line, and vision measurements are represented with superimposed crosses.

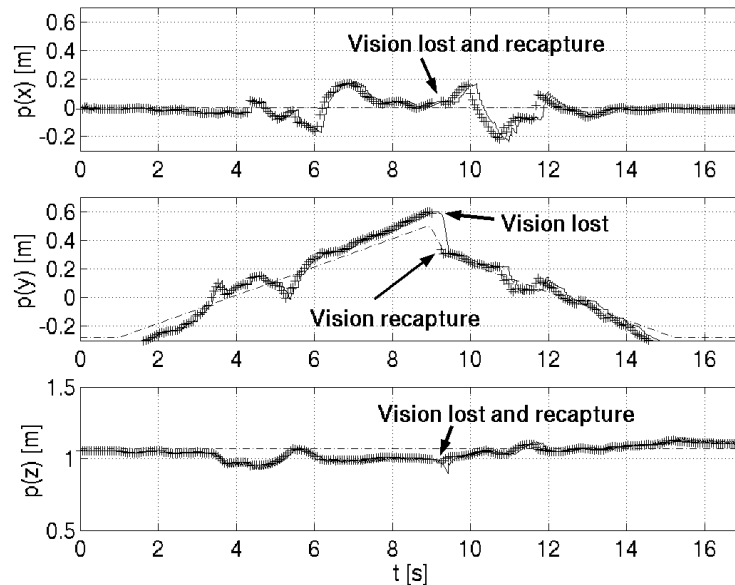


Figure 3.14: Successful camera pose-recover after losing three frames, when the robot was moving approximately 2m/s for 0.2m on the way back. The estimated translational part of the pose is plotted with a solid line, while ground-truth is displayed with dash dotted line, and vision measurements are represented with superimposed crosses.

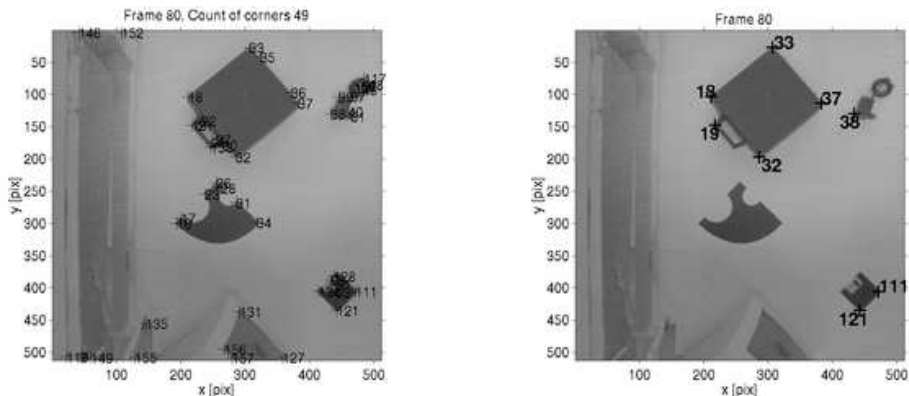


Figure 3.15: Left is the image with all estimated landmarks and right are highlighted only landmarks with very low or no change in their estimated 3D positions.

3.5.2.3 Structure Evaluation

To prove the structure estimation algorithm a sparse set of landmarks is estimated. These landmarks represent 3D positions of extracted corners (see Fig. 3.15). The errors between estimated 3D positions and the ground-truth 3D positions (measured with the KUKA robot ¹) are tabled in Tab. 3.7. To compare, which direction has the largest discrepancy, the Euclidean distances are calculated separately for x, y and z direction, then the Euclidean distances for two points in 3D space are calculated. Next the Mahalanobis distance (σ is the covariance matrix of the estimated landmark) is evaluated, and additionally all errors are calculated.

From Tab. 3.7 it is clear that the calculated error is largest in z and smallest in x direction. This result corresponds to the fact that the largest distance from the target was in z and smallest in x direction. The Euclidean distance of two points should diminish for every new point, because the system is smoothing the estimated pose. This is not true, due to the discrepancies in the motion estimation (see the end of Sec. 3.5.2.2). In opposite the Mahalanobis distance is diminishing, but this should not be taken as a measure for the distance calculation in this case. The Mahalanobis distance indicates that the system believes, that the new points (with higher IDs) are evaluated more precisely.

3.6 Conclusion

The presented motion and structure estimation framework has the ability to estimate the egomotion of the sensor rig by fusion of vision and inertial data, which has been tested on robotic and less smooth handheld motions. For generating the ground-truth values a mobile platform and a robotic arm have been used. However, the platform is able to perform only planar motion, while the system can estimate 6DOF motion. Free 6DOF motion is estimated in the case of handheld and robotic arm motion. In the case of handheld motion ground truth is not available. The estimation techniques use

¹The calibration error of the KUKA KR15/2 robot is less than 1[mm].

ID	$\ X_x - \bar{X}_x\ $	$\ X_y - \bar{X}_y\ $	$\ X_z - \bar{X}_z\ $	$\ X - \bar{X}\ $	$((X - \bar{X})^T \sigma^{-1} (X - \bar{X}))^{\frac{1}{2}}$
18	6.53e-3	1.246e-2	7.869e-2	7.993e-2	2.8e-2
19	1.44e-2	2.48e-2	1.19e-1	1.23e-1	1.35e-2
33	5.7e-4	1.33e-2	8.27e-2	8.38e-2	2.9e-3
37	1.14e-2	1.41e-2	4.99e-2	5.31e-2	1.9e-3
38	5.4e-3	4.6e-3	2.02e-2	2.15e-2	8.19e-4
111	1.05e-2	1.44e-2	6.78e-2	7.01e-2	2.3e-3
121	3.6e-3	3.58e-2	6.0e-2	7.0e-2	2.1e-3
Σ	5.24e-2	1.195e-1	4.783e-1	5.014e-1	5.15e-2

Table 3.7: Errors in structure estimation, while 3D points have been measured and estimated in [m].

the well-known EKF and UKF, both filters are used in the multi-rate sampling scheme, because of the asynchronous sampling of the vision and inertial sensor.

As presented in [40], we think that the proposed motion and structure estimation algorithms can help the mobile robot to navigate faster and more robustly. The real-time implementation of those algorithms proved to be able to deliver the pose of the sensor rig even though missing the presence of vision measurements. This useful property can be used in the case of performing rotations with the mobile platform.

However, the main disadvantage of this work is the ignorance of inter-feature correlations. In the structure estimation algorithm, the features are estimated separately and they are correlated only with the camera's pose. The lack of inter-feature correlations is an important drawback [91] and the main reason why we do not consider developing this structure algorithm in the future.

Due to this drawback we decided to substitute our motion and structure algorithms with the SLAM framework. The principle of this framework is explained in the next chapter.

Chapter 4

High-Speed Vision SLAM

In this chapter the motion and structure of an unknown scene is estimated using SLAM. This incremental probabilistic approach is more suitable to recover the geometry of an unknown environment than our previous algorithm (see Sec. 3.4.3). The need to incorporate the inter-features correlations in the probabilistic framework is argued by Smith and Cheeseman in [91]: “merge independent measurements by the sensor into a single approximate transformation”.

The success of SLAM and its applications can be considered as a product of well-advanced probabilistic techniques [97], where the basic algorithm is e.g. Kalman filter [54]. Thanks to those techniques a wide range of SLAM algorithms have been introduced, and successfully applied in real scenarios (e.g. [35, 95, 106, 115]). The key idea here is that the position of the robot as well as the features of the map are estimated, which means that we know where they might be with certain probability. This concept is crucial in SLAM and proved to work very well in practice [30].

An overview of SLAM is depicted in Fig. 4.1, and it consists of three parts. The *prediction* part predicts robot’s and scene feature’s position, while receiving information about the control and current positions. The next part *matches* the observed scene and predicted features, which is crucial for later updates. False matches can cause SLAM to collapse, and without a proper re-initialisation the position is lost. The last part is the *map building*, which is responsible for managing scene features.

As input for SLAM different kinds of sensors e.g. laser [106] or sonars [115] can be used. One of the most interesting (in terms of cost, weight, etc.) and challenging sensors is a single perspective-projective camera. When observing the environment with a camera, the depth information of new landmarks can not be directly acquired. To recover this depth information the camera has to move, and observe these landmarks from different viewpoints. While performing motions, the robot with the onboard camera is expected to move slowly, as any abrupt position changes can cause motion blur in the camera images.

The underlying assumption of vision SLAM is that in every two consecutive camera images enough feature matches have to be found. Otherwise, the SLAM framework will most likely diverge, causing the robot to get lost. If one camera image contains motion blur, matching the features becomes a challenging task and difficult to solve in

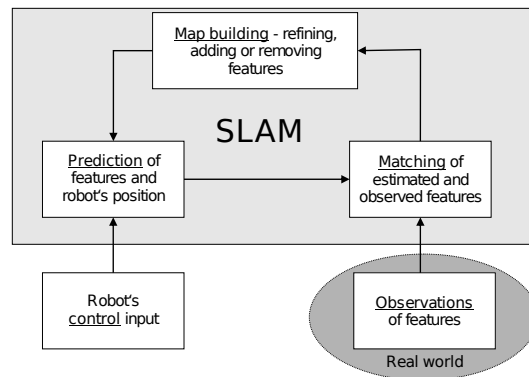


Figure 4.1: SLAM basic scheme.

real-time applications. Algorithms for image de-blurring are well-known [59], but they have a considerable computational burden, which is unsuitable for a real-time vision SLAM application.

A possibility of how to eliminate the motion blur and retain or increase the robot's translational and rotational velocity is to increase the readout speed of the vision sensor. Using a camera with a higher readout speed allows to update the pose of the robot more often. This can prevent the vision SLAM from diverging when more abrupt motion occurs.

The problem of camera motion and unknown environment reconstruction is known in the computer vision community as SfM. The main difference comparing to SLAM framework is that the probabilistic techniques are replaced with non-linear least square estimation algorithms. For a mobile robot real-time is a very important constraint, because it needs to react instantly in some scenarios. Generally the non-linear least square estimation requires a higher computational burden, but recently also real-time SfM frameworks have been introduced (e.g. [56]).

An important difference between SfM and vision SLAM frameworks is how they compensate for the motion drift. SfM frameworks, e.g. [77], track the features from frame to frame, and do not correct their positions after re-observing the same place. However, SLAM algorithms recognise a place that has been seen before, and this is known as the *loop-closing* [6]. Loop-closing is considered to be a data association problem, and SLAM is very fragile to incorrect association of observations [30]. The loop-closing problem is considered to be very difficult, and an overview of possible solutions is provided by Bailey and Durrant-White [6].

Davison et al. introduced the first real-time Monocular SLAM (MonoSLAM) algorithm, recently summarized in [27]. The camera motion estimation and incremental map building are computed within a standard EKF (see Sec. 2.3.4) SLAM framework. An alternative SLAM framework is based on FastSLAM-type particle filter (see Sec. 2.3.6) algorithms [31]. This FastSLAM-type particle filter has the advantage that it can estimate more features keeping the computational costs low, because it requires $O(MN)$ space. M is the count of particles and N is the count of landmarks.

However, MonoSLAM requires $O(N^2)$ space. The disadvantage of the FastSLAM-type particle filter algorithm is that the loop-closing SLAM property is not available, and has to be added using visual appearance-based models [33], for example. Using visual appearance-based models it is possible to identify an image or place in a video sequence.

This chapter consists of six sections. Sec. 4.1 and Sec. 4.2 present the related work. Sec. 4.3 introduces our contribution and Sec. 4.4 explains the MonoSLAM framework. Sec. 4.5 presents the theoretical background of high-speed MonoSLAM and Sec. 4.6 introduces experimental results. Finally, Sec. 4.7 concludes this chapter with an outline of the future work.

4.1 Related Work

The problem of SfM is tackled in the seminal work of Pollefeys et al. [81]. To find an optimal solution the authors presented offline non-linear and linear self-calibration approaches. The proposed self-calibration methods are based on the idea of an absolute conic (see Sec. 2.2.5.1).

In [17], an interesting three-dimensional real-time structure from motion estimation system is described. This system is based on non-linear filtering. The main disadvantage of this approach is the unavoidable presence of drift, which occurs when long motion is performed.

Bosse et al. presented in [10] a SfM system based on EKF recursive estimation. An interesting aspect of their work is that they exploited omni-directional images, and detected vanishing points (see Sec. 2.2.5) and lines. These features were used for motion and structure estimation. In subsequent work Bosse et al. [9] proposed the ATLAS sub-mapping framework, where they used a graph of coordinate frames for mapping large man-made environments.

In the same year, when Davison presented MonoSLAM [25], Nister introduced a system capable of robust real-time SfM estimation [77]. Nister's system is based on a preemptive scoring in RANSAC (see Sec. 2.1.6) and the framework uses the bundle adjustment [100] algorithm to find robust motion and structure estimates.

In [34], Elinas et al. introduced a stereo vision SLAM system based on particle filters, which is particularly suitable for large-scale environments. They used SIFT for feature matching (see Sec 2.1.4). But this computationally intensive matching hinders real-time processing.

Recently, a very interesting real-time parallel tracking and mapping approach was introduced by Klein and Murray in [56]. They proposed splitting the tracking and mapping into two separate tasks, running them in parallel threads. The goal of their approach is to estimate a dominant plane in a small workspace, where virtual characters can be displayed. The result is a system capable of estimating the pose of the camera while mapping thousands of features in real-time.

4.2 Increasing Localisation Robustness

One of the underlying assumptions in MonoSLAM is that the camera is expected to move smoothly. This classical EKF SLAM framework is prone to fail as soon as sudden or erratic camera movements occur, and can not reliably recover when the pose is lost. The smooth motion assumption attracted attention recently, and several authors proposed solutions to this problem.

Williams et al. presented in [108] an additional relocalisation algorithm, which can operate parallel to MonoSLAM, and increases robustness against camera shakes and occlusions. A recent improvement of this algorithm is using a randomised list classifier and RANSAC to determine the pose robustly [107]. This randomised list of classifier are trained when a feature is observed for the first time. The input for the training are 400 synthetically warped views of this feature, so it can be recognised from a variety of views.

To handle erratic camera motion and occlusions, Pupilli and Calway [82] presented a visual SLAM framework based on a FastSLAM-type particle filter. This work tackles mainly the problem of robust camera localisation. Chekhlov et al. [16] extended this SLAM framework to operate over a large range of views using a SIFT-like spatial gradient descriptor.

Another visual SLAM framework based on a FastSLAM-type particle filter introduced by Eade and Drummond [31] can incorporate hundreds of features in real-time. However, the filter needs to be adapted for closing loops over large trajectories.

The monocular SLAM algorithms discussed above [108, 107, 82, 16, 31] use a standard 30Hz camera.

Interesting high-speed applications already enabled to e.g. measure the motion of a waving flag or a flying ball [104]. Komuro and Ishikawa proposed in [57] a method for three-dimensional object tracking from noisy images. The noise typically produced by high-speed cameras can be tackled with proper noise models.

Another way to improve the robustness of tracking and estimating features is to increase the sampling rate of the camera. This has been beautifully demonstrated by Ishii et al. [49]. They introduced a 1ms visual feedback system using massively parallel processing. Since image acquisition time is countered with equally fast image processing (both operate at 1kHz), very fast motion of a bouncing ball can be tracked. However, objects need to be bright in front of dark background.

4.3 Contribution Description

The high-speed cameras used in this work outperform a standard CCD camera because they provide a faster read-out speed, can address independent windows, have little motion blur, and provide more visual information. This increase in visual information reduces the uncertainty in robot localisation and the uncertainties of the 3D positions of the new landmarks.

From the above-mentioned state-of-the-art visual SLAM frameworks we are using in this work the MonoSLAM algorithm [25, 27], because that it is the most convincing

one.

Our idea is to show that MonoSLAM can operate at a frame rate of 200Hz and that this improves the robustness of localisation considerably. An example of how higher frame rate improves the dynamics and system performance of visual tracking is presented in [103]. The intention of this chapter is to show that with a higher frame rate a performance similar to the recent improvements discussed above [108, 107, 82, 16, 31] can be achieved, while additionally all these measures could again be used to even further improve performance.

In MonoSLAM framework, it is assumed that the robot moves with constant velocity. This does not mean that the velocity is constant between two consecutive frames, but it is assumed to be constant on average. When we increase the readout speed of the camera, it should be possible to estimate motion with higher dynamics. However, to estimate such motion an extended motion model should be considered. We introduced additional continuous parameters, which model motion accelerations, and we assume that these accelerations are on average constant. The expected advantage of this constant acceleration motion model, when comparing it to the constant velocity motion model, is a better ability to track a faster motion.

We summarise our contributions as follows:

- introduction of the high-speed camera to MonoSLAM framework and
- presentation of an extended motion model.

We presented this contribution in [39].

4.4 MonoSLAM

The intention of this section is to give a brief overview of MonoSLAM. We divided this overview of functionalities into four sections. Firstly, the functionalities of MonoSLAM are described as an example in Sec. 4.4.1. Secondly, a probabilistic description of EKF MonoSLAM is given in Sec. 4.4.2. Thirdly, the camera pose estimation and the typical motion model description are presented in Sec. 4.4.3. Fourthly, the computation of map updates with recently published additional modules are mentioned in Sec. 4.4.4.

The only sensor input in MonoSLAM used in this work are images from a single perspective-projective camera as displayed in Fig. 4.2(a). After successful initialisation the camera moves into an unknown scene. The output of MonoSLAM are the camera poses and a sparse geometrical model of the scene. Due to the sparse map constraint, the system is capable of real-time performance, which is crucial for robotic applications. Additionally, a correct association of the present to the previously seen features enables MonoSLAM to close the loop.

However, before we start with this example, we have to mention the initial assumptions of MonoSLAM:

- constant lighting conditions,
- calibrated camera,

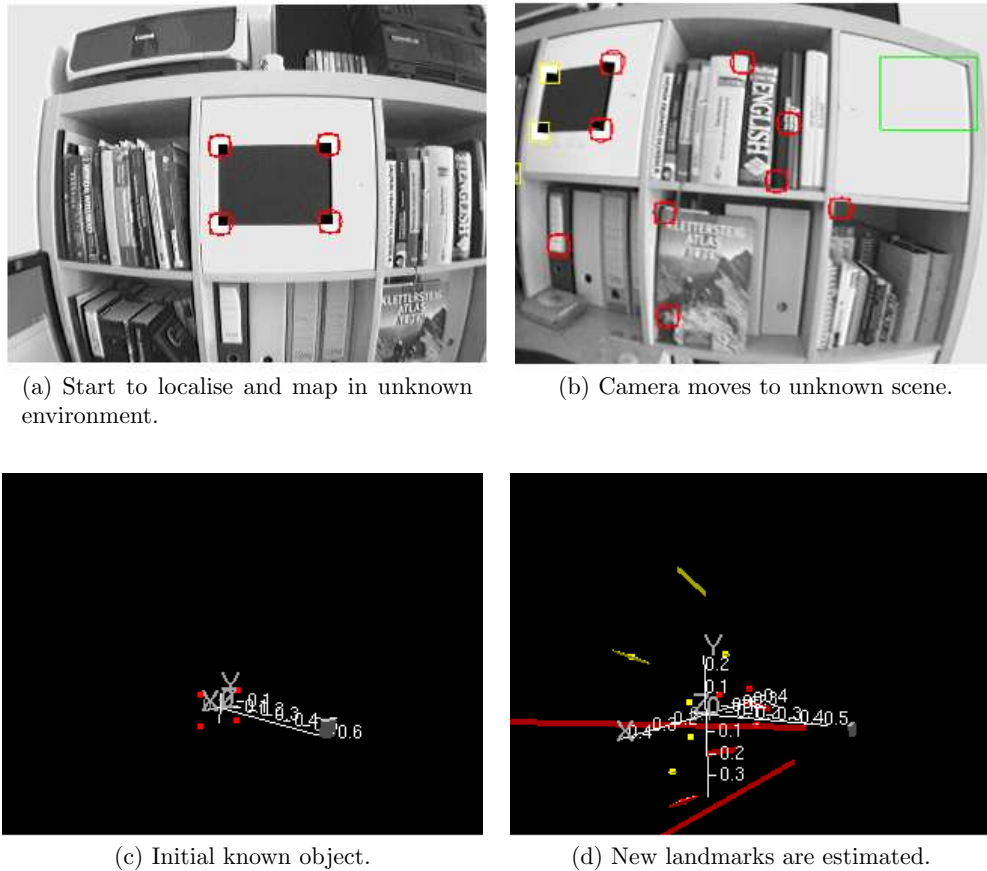


Figure 4.2: The goal of MonoSLAM is to simultaneously localise in an unknown environment and to generate a map from new landmarks. The upper row contains scene images, taken with the camera, and back-projected known features. The bottom row contains the resulting 3D maps, and the camera poses.

- smooth movements,
- rigid scene (no moving objects) and
- scene contains textured objects.

If at least one of these assumptions is not valid, then we can expect erratic MonoSLAM localisation or mapping results. A more detailed description of these assumptions can be found in Sec. 4.6.

4.4.1 Example: MonoSLAM

Using a perspective-projection camera in MonoSLAM has several advantages - the sensor is low-cost, small and light-weight. The drawback is that once a screenshot

of the scene is taken, the 3D information can not be directly observed. To estimate the position of a landmark, the camera has to move and for this several 2D feature positions are required. From these new landmarks a map is generated. These landmarks are subsequently required for the localization step. These steps are described in this subsection as an example where the camera moves in an indoor scene.

In this example, a perspective-projection camera with a wide-angle lens (see Sec. 2.2.4) has been used. Comparing this with a standard lens, the wide-angle lens has the advantage that the features stay longer in the field of view, resulting in a more stable MonoSLAM performance. However, this lens can introduce extremely large distortions. As mentioned in Sec. 2.2.4.3, this can be calibrated using only one radial coefficient. This one radial coefficient is satisfactory for MonoSLAM [27] to model the image distortions.

The initially known object can be placed arbitrarily in the scene, and its position describes the beginning of the world coordination frame (see Sec. 2.2.1). The geometry of the initial black rectangle is known, so the first known 3D positions (see Fig. 4.2(c)) can be re-projected into the image (see Fig. 4.2(a)). When the first features overlay with the initial object, tracking can be initialised.

As depicted in Fig. 4.2(a), after tracking is initialised the camera can start to move in the scene. Due to the read-out speed of the standard camera the camera motion has to be smooth. Any abrupt or shaky movement can cause that MonoSLAM will get lost. When the camera moves in a certain direction, MonoSLAM tries to detect new features in the direction of this motion (see Fig. 4.2(b)) so that these newly detected features do not overlap. This is possible due to the fact that the predicted camera pose is computed. Using the predicted pose and the displacement of projected known features in the image, MonoSLAM tries to found a non-overlapping region.

MonoSLAM is referred to as a *top-down* approach due to this decision strategy. This type of approach together with the usage of EKF puts MonoSLAM into the family of top-down Bayesian framework [27]. In opposition to the top-down approaches are the *bottom-up* methods. The SfM algorithms are based on the bottom-up approaches, for example [79].

As displayed in Fig. 4.2(d), after a new feature is detected using the Shi-Tomasi detector (see Sec. 2.1.2), it is inserted into the map of the scene. When the camera moves further into an unknown scene, and this feature is observable, then its 3D position uncertainty decreases (see Fig. 4.3(c)). After some video frames the initial object gets lost and the count of new features increases (see Fig. 4.3(a)). As the camera continues to move in the unknown space, MonoSLAM estimates the pose and scene structure (see Fig. 4.3(c)). When new features are inserted in the map, the uncertainty of the camera pose grows. Meaning that moving into a new environment without seeing known places increases the camera pose drift.

An important feature of MonoSLAM is the ability to compensate drift when an already known place is repeatedly seen. As depicted in Fig. 4.3(b), seeing the initial part of the scene helps to close the loop. The final scene map is displayed in Fig. 4.3(d), where the landmarks on the cupboard lies correctly in a plane.

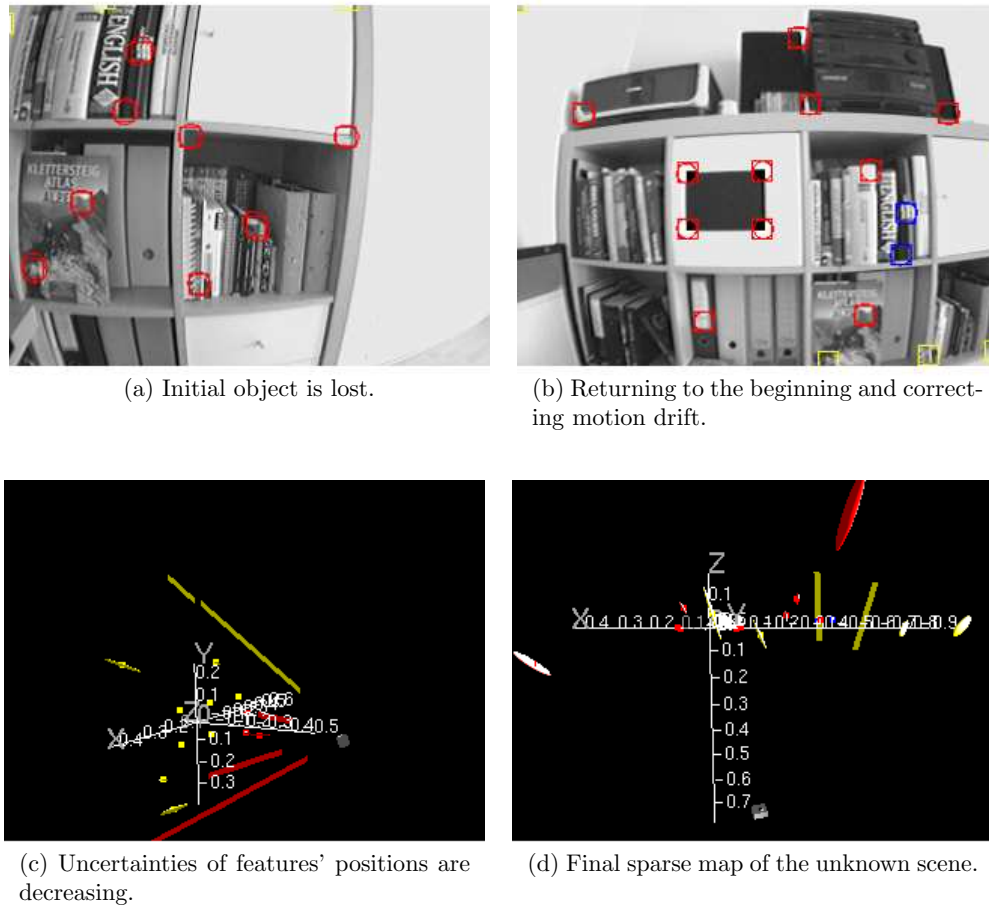


Figure 4.3: Loop-closing is an important MonoSLAM property. The upper row contains scene images, taken with the camera, and back-projected known features. The bottom row contains the resulting 3D maps, and the camera poses.

4.4.2 Probabilistic Description

MonoSLAM is formulated as a system with dynamic state estimation (see Sec. 2.3.1), where EKF is applied to estimate the true state of the system. The equations' notation in this chapter is identical with the description of EKF in Sec. 2.3.4.

The state vector $\hat{\mathbf{x}}$ contains the camera pose ($\hat{\mathbf{x}}_v$) and the 3D positions of estimated scene landmarks:

$$\hat{\mathbf{x}} = [\hat{\mathbf{x}}_v \quad \mathbf{y}_1 \quad \dots \quad \mathbf{y}_i \quad \dots]^T, \quad (4.1)$$

where i is the index of the feature's estimated 3D position \mathbf{y} .

The symmetric covariance matrix \mathbf{P} has the following form:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{P}_{xy_1} & \mathbf{P}_{xy_2} & \cdots \\ \mathbf{P}_{y_1x} & \mathbf{P}_{y_1y_1} & \mathbf{P}_{y_1y_2} & \cdots \\ \mathbf{P}_{y_2x} & \mathbf{P}_{y_2y_1} & \mathbf{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad (4.2)$$

where $\mathbf{P}_{y_1x} = \mathbf{P}_{xy_1}^T$.

The prediction and update steps of EKF estimation procedure are explained in the subsequent subsections.

4.4.3 Motion Representation

This subsection is focused on the mathematical model of the motion representation. The goal of this motion representation is to predict (see Sec. 2.3.4) the camera's next pose. This predicted pose determines, where the searching ellipses (see e.g. Fig. 4.2(b)) will lie in the next image frame.

In this work, the only used sensor in MonoSLAM is a single camera, which can perform an unconstrained motion in 3D space. The vector $\hat{\mathbf{x}}_v$ describes the camera motion state with 13 states, and has the following form:

$$\hat{\mathbf{x}}_v = [\mathbf{r}^W \quad \mathbf{q}^{WC} \quad \vec{v}^W \quad \vec{\omega}^C]^T, \quad (4.3)$$

where $\mathbf{r}_{1 \times 3}^W$ is the position and $\mathbf{q}_{1 \times 4}^{WC}$ is the orientation of the camera in reference to the initial object (see Sec. 2.2.2). Vectors $\vec{v}_{1 \times 3}^W$ and $\vec{\omega}_{1 \times 6}^C$ represent the linear and angular velocities. W is the coordinate frame of the initial object (referred to as world frame) and C is the camera coordinate frame.

The motion of the camera can be represented with different models. Davison, for example, has been using the *constant linear velocity* and the *constant angular velocity* model in his work [27]. The modeling of the camera motion state stops with the 1st order derivatives. With this representation a smooth motion is assumed.

Higher motion derivatives such as linear acceleration, angular acceleration or jitter can be modeled as noise. In MonoSLAM, the noise modeling stops with 2nd order derivatives. The vector $\vec{n}_{6 \times 1}$ represents the camera motion noise, and can be written as

$$\vec{n} = \begin{bmatrix} \vec{V}^W \\ \vec{\Omega}^C \end{bmatrix} = \begin{bmatrix} \vec{a}^W \Delta t \\ \vec{\alpha}^C \Delta t \end{bmatrix}, \quad (4.4)$$

where $a_{3 \times 1}^W$ is an unknown linear acceleration and $\alpha_{3 \times 1}^C$ is an unknown angular acceleration.

The covariance matrix of the noise vector \vec{n} is defined as:

$$\mathbf{P}_n = \begin{bmatrix} \mathbf{P}_{VV} & \mathbf{P}_{V\Omega} \\ \mathbf{P}_{\Omega V} & \mathbf{P}_{\Omega\Omega} \end{bmatrix}. \quad (4.5)$$

The values of \mathbf{P}_n define what kind of motion do we might expect. The higher are the values in the matrix \mathbf{P}_n , the more rapid motion is expected.

4.4.3.1 Constant Velocity Motion Model

The constant linear and angular velocity motion model is characterised by Davison in [27] as: “...we assume that the camera moves at a constant velocity over all time, but that our statistical model of its motion in a time step is that on average we expect undetermined accelerations occur with a Gaussian profile.”

In this subsection, for a better readability the exponents W and C are omitted in the following equations.

The state vector $\hat{\mathbf{x}}_v$ is described in Eq. 4.21, and the covariance of this state vector has this form:

$$\mathbf{P}_v = \begin{bmatrix} \mathbf{P}_{rr} & \mathbf{P}_{rq} & \mathbf{P}_{rv} & \mathbf{P}_{r\omega} \\ \mathbf{P}_{qr} & \mathbf{P}_{qq} & \mathbf{P}_{qv} & \mathbf{P}_{q\omega} \\ \mathbf{P}_{vr} & \mathbf{P}_{vq} & \mathbf{P}_{vv} & \mathbf{P}_{v\omega} \\ \mathbf{P}_{\omega r} & \mathbf{P}_{\omega q} & \mathbf{P}_{\omega v} & \mathbf{P}_{\omega\omega} \end{bmatrix}. \quad (4.6)$$

The prediction of the state $\hat{\mathbf{x}}_v$ is computed as:

$$\hat{\mathbf{x}}_v(k|k-1) = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \\ \vec{v} \\ \vec{\omega} \end{bmatrix}_{k|k-1} = \begin{bmatrix} \mathbf{r} + (\vec{v} + \vec{V}\Delta t) \\ \mathbf{q} \times \mathbf{q}((\vec{\omega} + \vec{\Omega})\Delta t) \\ \vec{v} + \vec{V} \\ \vec{\omega} + \vec{\Omega} \end{bmatrix}_{k-1|k-1}, \quad (4.7)$$

where \vec{V} and $\vec{\Omega}$ are unknown impulses with Gaussian profiles (see Eq. 4.4), and the quaternion product \times is defined in [101].

After the computation of Eq. 4.7 the predicted state $\hat{\mathbf{x}}_v(k|k-1)$ equals:

$$\hat{\mathbf{x}}_v(k|k-1) = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{I}_{3 \times 3} \Delta t & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}}{\partial \vec{\omega}} & \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}}{\partial \vec{\omega}} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix}. \quad (4.8)$$

After differentiating of the two elements in Eq. 4.8 the results are:

$$\begin{aligned} \frac{\partial \mathbf{q}}{\partial \vec{q}} &= M_\tau(\mathbf{q}(\vec{\omega} \Delta t)) \\ \frac{\partial \mathbf{q}}{\partial \vec{\omega}} &= M'_\tau(\mathbf{q}(k-1|k-1)) \frac{\partial \mathbf{q}(\vec{\omega} \Delta t)}{\partial \vec{\omega}}, \end{aligned} \quad (4.9)$$

where $M_\tau(\mathbf{q})$ equals:

$$M_\tau(\mathbf{q}) = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{bmatrix}, \quad (4.10)$$

$M'_\tau(\mathbf{q})$ has the form:

$$M'_7(\mathbf{q}) = \begin{bmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{bmatrix}, \text{ and} \quad (4.11)$$

$$\frac{\partial \mathbf{q}(\vec{\omega} \Delta t)}{\partial \vec{\omega}} = \begin{bmatrix} -\frac{\Delta t}{2} \frac{\omega_x}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\| \Delta t}{2} & \cdots & \cdots \\ \frac{\Delta t}{2} \frac{\omega_x^2}{\|\vec{\omega}\|^2} \cos \frac{\|\vec{\omega}\| \Delta t}{2} + \frac{1}{\|\vec{\omega}\|} \left(1 - \frac{\omega_x^2}{\|\vec{\omega}\|^2}\right) \sin \frac{\|\vec{\omega}\| \Delta t}{2} & \cdots & \cdots \\ \frac{\omega_x \omega_y}{\|\vec{\omega}\|^2} \left(\frac{\Delta t}{2} \cos \frac{\|\vec{\omega}\| \Delta t}{2} - \frac{1}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\| \Delta t}{2}\right) & \cdots & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \quad (4.12)$$

In the above-mentioned equation:

$$\|\vec{\omega}\| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}. \quad (4.13)$$

The missing elements in Eq. 4.12 can be computed by symmetry.

The last step of state vector prediction is the normalisation of the predicted quaternion $\mathbf{q}(k|k-1)$. The computation of quaternion normalisation can be found in [18, 58, 101].

The computation of the prediction covariance matrix, when applying the unknown impulses with Gaussian profiles (see Eq. 4.4), results in this form:

$$\mathbf{P}_v(k|k-1) = \begin{bmatrix} \mathbf{P}_{rr} & \mathbf{P}_{rq} & \mathbf{P}_{rv} & \mathbf{P}_{r\omega} \\ \mathbf{P}_{qr} & \mathbf{P}_{qq} & \mathbf{P}_{qv} & \mathbf{P}_{q\omega} \\ \mathbf{P}_{vr} & \mathbf{P}_{vq} & \mathbf{P}_{vv} + \mathbf{P}_{VV} & \mathbf{P}_{v\omega} + \mathbf{P}_{V\Omega} \\ \mathbf{P}_{\omega r} & \mathbf{P}_{\omega q} & \mathbf{P}_{\omega v} + \mathbf{P}_{\Omega V} & \mathbf{P}_{\omega\omega} + \mathbf{P}_{\Omega\Omega} \end{bmatrix}_{k|k-1}. \quad (4.14)$$

The values of \mathbf{P}_{VV} and $\mathbf{P}_{\Omega\Omega}$ define the expected linear and angular impulses. We assume no other impulses so the off-diagonal elements $\mathbf{P}_{V\Omega}$ and $\mathbf{P}_{\Omega V}$ equal zero.

The process noise covariance matrix $\mathbf{Q}_v(k-1)$ equals:

$$\mathbf{Q}_v(k-1) = \frac{\partial \hat{\mathbf{x}}(k|k-1)}{\partial \vec{n}} \mathbf{P}_{\vec{n}\vec{n}} \frac{\partial \hat{\mathbf{x}}(k|k-1)^T}{\partial \vec{n}}. \quad (4.15)$$

The noise vector \vec{n} and the noise covariance matrix are defined in Eq. 4.4, and Eq. 4.5 respectively.

After computing the partial differential equation in Eq. 4.15, we get:

$$\frac{\partial \hat{\mathbf{x}}(k|k-1)}{\partial \vec{n}} = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial \vec{V}} & \frac{\partial \mathbf{r}}{\partial \vec{\Omega}} \\ \frac{\partial \mathbf{q}}{\partial \vec{V}} & \frac{\partial \mathbf{q}}{\partial \vec{\Omega}} \\ \frac{\partial \vec{v}}{\partial \vec{V}} & \frac{\partial \vec{v}}{\partial \vec{\Omega}} \\ \frac{\partial \vec{\omega}}{\partial \vec{V}} & \frac{\partial \vec{\omega}}{\partial \vec{\Omega}} \end{bmatrix}_{k|k-1} = \begin{bmatrix} \mathbf{I}_{3 \times 3} \Delta t & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}(k|k-1)}{\partial \vec{\Omega}} \\ \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix}, \quad (4.16)$$

where $\frac{\partial \mathbf{q}(k|k-1)}{\partial \vec{\Omega}}$ equals $\frac{\partial \mathbf{q}}{\partial \vec{\omega}}(\vec{\omega}(k|k-1)\Delta t)$ in Eq. 4.12.

4.4.4 Mapping

In this subsection, a brief description of MonoSLAM map building is described in three parts. Firstly, active matching of known features with new measurements, and the computation of map updates is mentioned in Sec. 4.4.4.1. Secondly, feature initialisation using two different methods of representing a new feature’s depth is described in Sec. 4.4.4.2. Thirdly, recently presented MonoSLAM additional modules are explained in Sec. 4.4.4.3.

4.4.4.1 Computing Map Updates

The goal here is to explain how a known feature is found and updated. In the previous section we computed a predicted camera pose, which will be used here to maximise the feature’s search region.

Using the *predicted* camera position \mathbf{r}^W and orientation \mathbf{R}^{CW} , the new position of a feature in the camera frame \mathbf{h}_L^C is computed by:

$$\mathbf{h}_L^C = \mathbf{R}^{CW}(\mathbf{y}_i - \mathbf{r}^W), \quad (4.17)$$

where \mathbf{y}_i is feature’s last known position in the world frame. An explanation of transforming a point from world to camera coordinate system can be found in Sec. 2.2.1.

The computed position of the feature \mathbf{h}_L^C is projected in the image using a perspective-projective camera model as explained in Sec. 2.2.3. However, the camera lens introduces non-linear distortions. In MonoSLAM these distortions are approximated using the Swaminathan Model (see Sec. 2.2.4.3). The result of applying the *radial* distortion to the projected position \mathbf{h}_L^C is a two-dimensional point \mathbf{u}_{di} lying in the image plane.

The search region for this feature is defined by a 2×2 covariance matrix \mathbf{S}_i :

$$\mathbf{S}_i = \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v} \mathbf{P}_{xx} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v}^T + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v} \mathbf{P}_{xy_i} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i}^T + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i} \mathbf{P}_{y_i x} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{x}_v}^T + \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i} \mathbf{P}_{y_i y_i} \frac{\partial \mathbf{u}_{di}}{\partial \mathbf{y}_i}^T + \mathbf{R}, \quad (4.18)$$

where \mathbf{P}_{xx} , $\mathbf{P}_{y_i x}$, \mathbf{P}_{xy_i} , and $\mathbf{P}_{y_i y_i}$, are parts of the covariance matrix \mathbf{P} mentioned in Eq. 4.2. The covariance matrix \mathbf{R} defines the measurement noise, which is determined by the image resolution. This means that the value of \mathbf{R} increases from the centre to the image corners.

The active matching of a feature at the position \mathbf{u}_{di} and within the search region \mathbf{S}_i is the last step before computing the map update. When the feature’s stored template texture can be successfully matched (see Sec. 2.1.5) with the current image, this feature is selected for the map update computation. In MonoSLAM between 10 and 12 features are selected in each time step and the map update is computed as EKF update step (see Sec. 2.3.4).

4.4.4.2 Feature Initialisation

In the previous section, only the state of known features has been updated. When mapping an unknown scene new features have to be initialised. As described in the

introduction of this chapter the initial depth of new features is unknown and can not be acquired directly.

The initial feature position estimate is a difficult task, because the perspective-projective camera model is highly non-linear. In MonoSLAM the new features' position estimation is carefully treated to avoid updating the map with false landmarks. The challenging part of the scene structure estimation is the unknown depth. An unknown feature can possibly lie before the lens or at very large distances, which is difficult to recognise from a single image.

The depth of a new feature is estimated in [27] using a particle set. This means that a new feature lies on a semi-infinite line starting at the camera position and heading along ray \mathbf{h}_i^W . The representation for this feature is as follows:

$$\mathbf{y}_{pi} = \begin{bmatrix} \mathbf{r}_i^W \\ \mathbf{h}_i^W \end{bmatrix}, \quad (4.19)$$

where \mathbf{r}_i^W is one end of the semi-infinite line.

The main disadvantage of this representation is that only features in about 5m range around the camera position can be considered. Initialising more distant features would need a large particle set, which hinders the real-time constraint of MonoSLAM. Another disadvantage is that it takes several steps to convert a new feature to a fully-initialised point, as depicted in Fig. 4.4.

In [72] a straightforward feature initialisation is presented. A new feature is represented as:

$$\mathbf{y}_{pi} = \begin{bmatrix} \mathbf{r}_i^W \\ \theta_i \\ \phi_i \\ \rho_i \end{bmatrix}, \quad (4.20)$$

where \mathbf{r}_i^W is the position of the optical centre of the camera, θ_i is the azimuth, ϕ_i is the elevation, and ρ_i is the *inverse* depth of the feature.

Due to this improved initialisation very distant features can be considered. Another advantage is that no initial object is needed, because this representation allows new features to be added to the camera pose computation. An example is depicted in Fig. 4.5.

4.4.4.3 Additional Modules and Extensions

Due to the concentrated effort, MonoSLAM framework has been improved using several additional modules and extensions. This section briefly describes the algorithms used in this chapter.

One of the most important parts of the whole framework is the *map management* introduced by Davison et al. in [25]. The goal of this module is to keep only reliable features in the map of the scene. The count of these features has to be constrained by the real-time performance.

A disadvantage of MonoSLAM was that the used Shi-Tomasi features were not *rotation invariant*. A potential remedy would be to use the SIFT features (see Sec. 2.1.4),

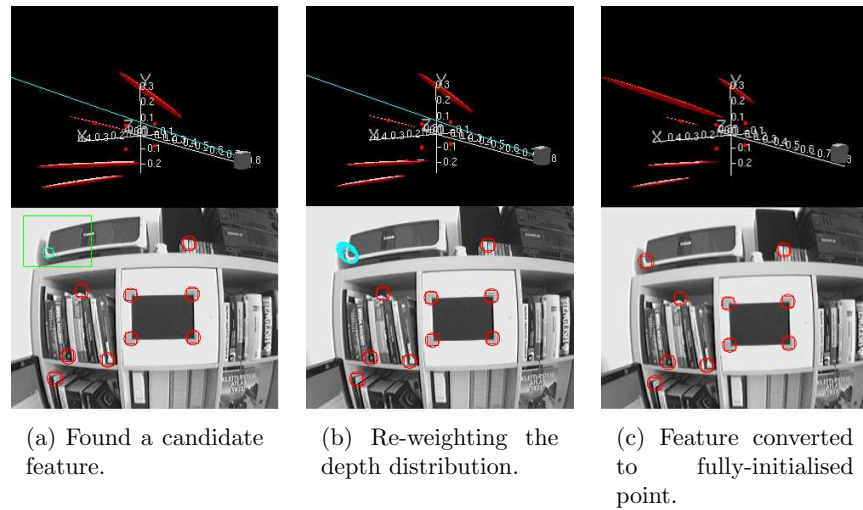


Figure 4.4: Feature initialisation using the depth representation based on the particle set. In the first frame, a candidate image patch is found. When a new feature is found, a ray is projected into subsequent images. A set of ellipses are searched along the ray to re-weight the depth distribution. Typically, after several time steps the depth uncertainty reduces the depth uncertainty, and the new feature is converted to a fully-initialised point.

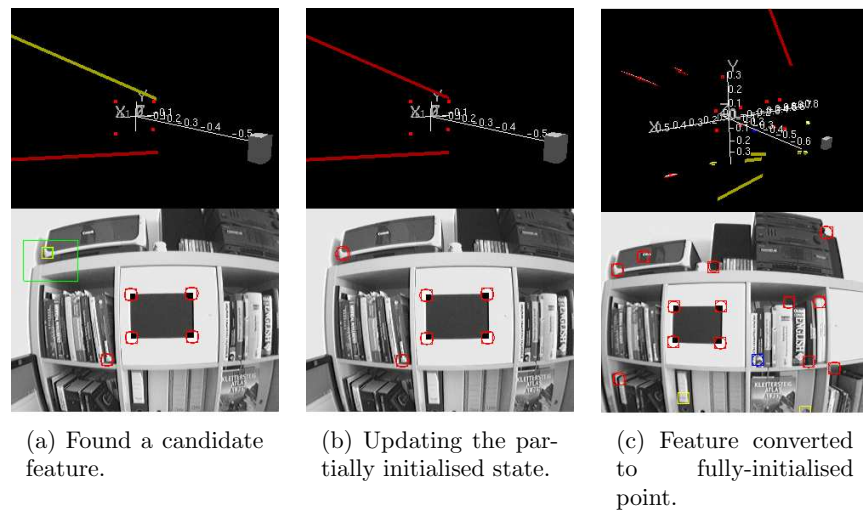


Figure 4.5: Feature initialisation based on the depth representation known as the inverse depth. Similarly, as by the particle set depth representation, in the first frame, a candidate feature is found. When a new feature is found, the explicit feature parametrisation is updated. In this example, this feature converts to the fully-initialised point after longer period of time.

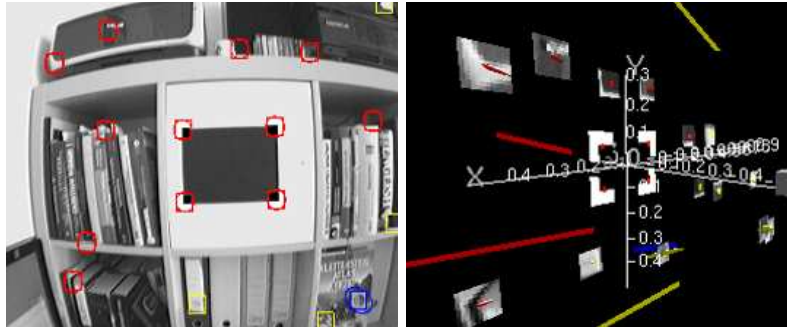


Figure 4.6: Feature orientation estimation of an indoor scene containing one dominant planar plane.

but they require higher computational burden. As depicted in Fig. 4.6, another way of handling the camera rotations is to estimate the orientation of the features [27].

In MonoSLAM the *data association* problem is crucial, because false feature matches typically cause the framework to collapse. The standard feature matching algorithm in SLAM systems is the Nearest Neighbour (NN) method. This algorithm is efficient, but selects only *individual* compatible hypothesis. In [75] Neira and Tardós introduced an improved version of NN, which is called Joint Compatibility Branch and Bound (JCBB). The most important difference between JCBB and the standard NN method is that JCBB is selecting only *joint* compatible hypothesis. One example of this difference is depicted in Fig. 4.7. JCBB improves the robustness of the data association in MonoSLAM as presented in [21].

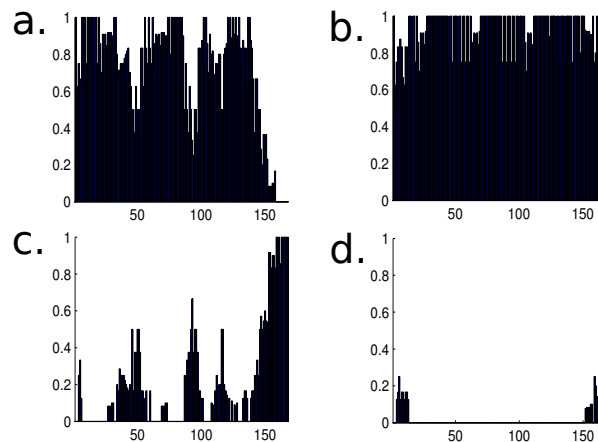


Figure 4.7: The result of a comparison between NN and JCBB algorithms using a two-dimensional robot simulation. The first row (a. and b.) displays the *true positives* values, and the second row (c. and d.) represents the *false positives* values. The first column depicts NN and the second column JCBB results.

Additionally, a possible way of improving the performance of a new feature detection in MonoSLAM is to use *Fast* features (see Sec. 2.1.3). An example is displayed in Fig. 4.5(a).

4.5 Improving Localisation Robustness

To explain the details related to the more robust localisation, this section comprises four parts. Firstly, the benefits of the localisation using the high-speed camera are presented in Sec. 4.5.1. Secondly, the proposed more robust MonoSLAM motion model is explained in Sec. 4.5.2. Thirdly, a discussion of motion parameters is included in Sec. 4.5.3. Fourthly, a problem of skipping vision information when using an asynchronous high-speed camera is reviewed in Sec. 4.5.4.

4.5.1 Dense Visual Information

The standard camera (difference of timestamps Δt is equal 33ms) provides rather sparse vision input, but when a smooth movement is expected (see Fig. 4.8), it is sufficient for MonoSLAM to robustly *localise*. If an erratic camera motion is performed, EKF based MonoSLAM is very likely to lose the pose. Observing the scene features with faster readout speed (e.g. Δt is equal 5ms) allows to update the camera state more frequently, and this can prevent MonoSLAM losing the pose as depicted schematically in Fig. 4.9.

4.5.2 Constant Acceleration Motion Model

In MonoSLAM it is assumed that the camera linear and angular velocities may change in every frame but they are expected to be *constant* on average. In other words, the camera movements are approximated using the so-called *constant linear and angular velocity* motion model. This model assumes that in each time step the unknown linear (\vec{a}^W) and the unknown angular ($\vec{\alpha}^R$) accelerations cause impulses of linear (\vec{V}^W) and angular ($\vec{\Omega}^W$) velocities. The noise vector \vec{n} is defined in Eq. 4.4 and the unknown accelerations are assumed to be zero mean Gaussian processes.

However, the high-speed camera can readout images several times faster (Δt is equal e.g. 5ms) than a standard camera (Δt is 33ms). Due to this high frame rate we assume that the velocities (\vec{v}^W and $\vec{\omega}^R$) can *change* arbitrarily, but the accelerations (\vec{a}^W and $\vec{\alpha}^R$) are expected to be *constant* on average.

The *constant linear and angular velocity* motion model can be extended to a second order model. The second order model includes linear (\vec{a}^W) and angular ($\vec{\alpha}^R$) accelerations in the camera state vector $\hat{\mathbf{x}}_{\mathbf{v}}$. We expect that this model can handle more erratic and jitter motion as depicted schematically in Fig. 4.10.

As introduced in Eq. 4.21. the camera vector state using the constant linear and angular velocity model has 13 states. The extended camera state vector $\hat{\mathbf{x}}_{\mathbf{v}}$ in the *constant linear and angular acceleration* model has 6 new states and includes

$$\hat{\mathbf{x}}_{\mathbf{v}} = [\mathbf{r}^W \quad \mathbf{q}^{WC} \quad \vec{v}^W \quad \vec{\omega}^C \quad \vec{a}^W \quad \vec{\alpha}^C]^T, \quad (4.21)$$

where the zero order (\mathbf{r}^W and \vec{q}^{WC}) and the first order (\vec{v}^W and $\vec{\omega}^C$) states are inherited from the first order motion model. The new state comprises of the linear (\vec{a}^W) and angular ($\vec{\alpha}^C$) accelerations.

We assume that in each time step when using the constant linear and angular acceleration model, an unknown linear jitter \vec{j}^W and unknown angular jitter $\vec{\eta}^C$ cause

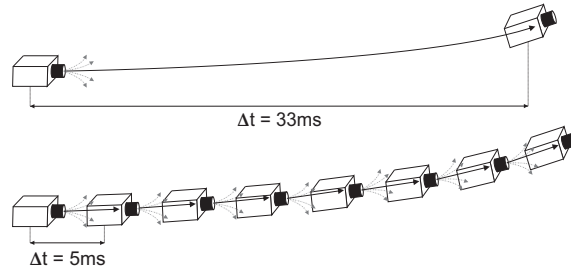


Figure 4.8: MonoSLAM can handle smooth camera movements using the standard camera (difference of timestamps - Δt is equal 33ms), and the high-speed camera (Δt is equal 5ms) is not needed.

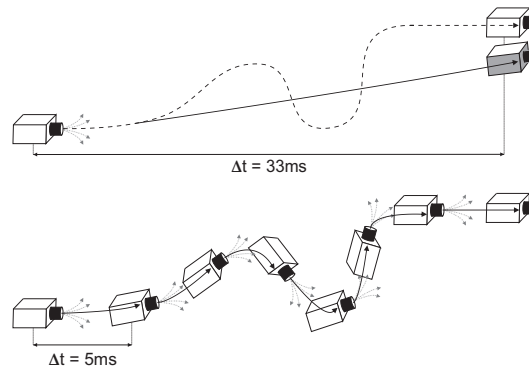


Figure 4.9: If erratic camera movements occur, MonoSLAM using the standard camera (difference of timestamps - Δt is equal 33ms) loses the pose. An example is displayed in the upper row, where the camera pose (solid line) drifted away from the true pose (dashed line). A high-speed camera provides more dense visual information, which helps to handle this erratic motion, as depicted in the bottom row.

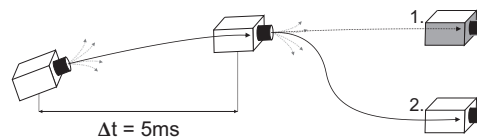


Figure 4.10: If erratic or jitter movements occur and dense vision information are available, the second order motion model (2) is assumed to be more robust than the first order model (1).

an impulse of linear (\vec{A}^W) and angular ($\vec{\Psi}^C$) accelerations. These accelerations are again zero mean Gaussian processes and the noise vector is

$$\vec{n} = \begin{bmatrix} \vec{A}^W \\ \vec{\Psi}^C \end{bmatrix} = \begin{bmatrix} \vec{j}^W \Delta t \\ \vec{\eta}^C \Delta t \end{bmatrix}. \quad (4.22)$$

For a better readability, the below-mentioned equations in this subsection are missing the exponents W and C .

The camera state update is computed as follows:

$$\hat{\mathbf{x}}_v(k|k-1) = \begin{bmatrix} \mathbf{r} \\ \mathbf{q} \\ \vec{v} \\ \vec{\omega} \\ \vec{a} \\ \vec{\alpha} \end{bmatrix}_{k|k-1} = \begin{bmatrix} \mathbf{r} + \vec{v}\Delta t + \frac{1}{2}(\vec{a} + \vec{A})\Delta t^2 \\ \mathbf{q} \times \mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}(\vec{\alpha} + \vec{\Psi})\Delta t^2) \\ \vec{v} + (\vec{a} + \vec{A})\Delta t \\ \vec{\omega} + (\vec{\alpha} + \vec{\Psi})\Delta t \\ \vec{a} + \vec{A} \\ \vec{\alpha} + \vec{\Psi} \end{bmatrix}_{k-1|k-1}, \quad (4.23)$$

where the quaternion product \times is defined in [101].

The notation:

$$\mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}(\vec{\alpha} + \vec{\Psi})\Delta t^2) \quad (4.24)$$

represents the quaternion trivially defined by the addition of two angle-axis rotation vectors:

$$\vec{\omega}\Delta t \text{ and } \frac{1}{2}(\vec{\alpha} + \vec{\Psi})\Delta t^2. \quad (4.25)$$

After the computation the predicted state $\hat{\mathbf{x}}_v(k|k-1)$ equals:

$$\hat{\mathbf{x}}_v(k|k-1) = \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{I}_{3 \times 3} \Delta t & \mathbf{0}_{3 \times 3} & \frac{1}{2}\mathbf{I}_{3 \times 3} \Delta t^2 & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}}{\partial \mathbf{q}} & \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}}{\partial \vec{\omega}} & \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}}{\partial \vec{a}} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \Delta t & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \Delta t \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 4} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix}. \quad (4.26)$$

The quaternion $\mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2)$ is treated as a result of an addition of two quaternions:

$$\mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2) = \mathbf{q}(\vec{\omega}\Delta t) + \mathbf{q}(\frac{1}{2}\vec{\alpha}\Delta t^2). \quad (4.27)$$

Using the result of quaternion addition:

$$\begin{aligned}
\mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2) &= \begin{bmatrix} \cos \frac{\|\vec{\omega}\|\Delta t}{2} \\ \frac{\omega_x}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\|\Delta t}{2} \\ \frac{\omega_y}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\|\Delta t}{2} \\ \frac{\omega_z}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\|\Delta t}{2} \end{bmatrix} + \begin{bmatrix} \cos \frac{\|\vec{\alpha}\|\Delta t^2}{4} \\ \frac{\alpha_x}{\|\vec{\alpha}\|} \sin \frac{\|\vec{\alpha}\|\Delta t^2}{4} \\ \frac{\alpha_y}{\|\vec{\alpha}\|} \sin \frac{\|\vec{\alpha}\|\Delta t^2}{4} \\ \frac{\alpha_z}{\|\vec{\alpha}\|} \sin \frac{\|\vec{\alpha}\|\Delta t^2}{4} \end{bmatrix}, \\
&= \begin{bmatrix} \cos \frac{\|\vec{\omega}\|\Delta t}{2} + \cos \frac{\|\vec{\alpha}\|\Delta t^2}{4} \\ \frac{\omega_x}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\|\Delta t}{2} + \frac{\alpha_x}{\|\vec{\alpha}\|} \sin \frac{\|\vec{\alpha}\|\Delta t^2}{4} \\ \frac{\omega_y}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\|\Delta t}{2} + \frac{\alpha_y}{\|\vec{\alpha}\|} \sin \frac{\|\vec{\alpha}\|\Delta t^2}{4} \\ \frac{\omega_z}{\|\vec{\omega}\|} \sin \frac{\|\vec{\omega}\|\Delta t}{2} + \frac{\alpha_z}{\|\vec{\alpha}\|} \sin \frac{\|\vec{\alpha}\|\Delta t^2}{4} \end{bmatrix}, \tag{4.28}
\end{aligned}$$

where $\|\vec{\omega}\|$ is defined in Eq. 4.13, and

$$\|\vec{\alpha}\| = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2}. \tag{4.29}$$

After differentiating of the three elements in Eq. 4.26 the results are:

$$\frac{\partial \mathbf{q}}{\partial q} = M_\tau(\mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2)), \tag{4.30}$$

$$\frac{\partial \mathbf{q}}{\partial \vec{\omega}} = M'_\tau(\mathbf{q}(k-1|k-1)) \frac{\partial \mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2)}{\partial \vec{\omega}}, \tag{4.31}$$

and

$$\frac{\partial \mathbf{q}}{\partial \vec{\alpha}} = M'_\tau(\mathbf{q}(k-1|k-1)) \frac{\partial \mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2)}{\partial \vec{\alpha}}. \tag{4.32}$$

The M_τ in Eq. 4.30 is defined in Eq. 4.10. M'_τ in Eq. 4.31 and Eq. 4.32 is defined in Eq. 4.11.

The first element of the result of the differentiating Eq. 4.31 is:

$$\frac{\partial \mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2)}{\partial \omega_x} = \begin{bmatrix} -\frac{1}{2} \sin(\frac{\|\vec{\omega}\|\Delta t}{2}) \frac{\omega_x \Delta t}{\|\vec{\omega}\|} \\ \frac{1}{\|\vec{\omega}\|} \sin(\frac{\|\vec{\omega}\|\Delta t}{2}) - \frac{\omega_x^2}{\|\vec{\omega}\|^3} \sin(\frac{\|\vec{\omega}\|\Delta t}{2}) + \frac{\omega_x^2}{\|\vec{\omega}\|^2} \cos(\frac{\|\vec{\omega}\|\Delta t}{2}) \frac{\Delta t}{2} \\ -\frac{\omega_x \omega_y}{\|\vec{\omega}\|^3} \sin(\frac{\|\vec{\omega}\|\Delta t}{2}) + \frac{\omega_x \omega_y}{\|\vec{\omega}\|^2} \cos(\frac{\|\vec{\omega}\|\Delta t}{2}) \frac{\Delta t}{2} \\ -\frac{\omega_x \omega_z}{\|\vec{\omega}\|^3} \sin(\frac{\|\vec{\omega}\|\Delta t}{2}) + \frac{\omega_x \omega_z}{\|\vec{\omega}\|^2} \cos(\frac{\|\vec{\omega}\|\Delta t}{2}) \frac{\Delta t}{2} \end{bmatrix}, \tag{4.33}$$

where the other elements can be computed by symmetry.

Similarly, the first element of the differentiation of Eq. 4.32 equals:

$$\frac{\partial \mathbf{q}(\vec{\omega}\Delta t + \frac{1}{2}\vec{\alpha}\Delta t^2)}{\partial \alpha_x} = \begin{bmatrix} -\frac{1}{4} S_a \frac{\alpha_x \Delta t^2}{\|\vec{\alpha}\|} \\ \frac{1}{\|\vec{\alpha}\|} S_a - \frac{\alpha_x^2}{\|\vec{\alpha}\|^3} S_a + \frac{\alpha_x^2}{\|\vec{\alpha}\|^2} C_a \frac{\Delta t^2}{4} \\ -\frac{\alpha_x \alpha_y}{\|\vec{\alpha}\|^3} S_a + \frac{\alpha_x \alpha_y}{\|\vec{\alpha}\|^2} C_a \frac{\Delta t^2}{4} \\ -\frac{\alpha_x \alpha_z}{\|\vec{\alpha}\|^3} S_a + \frac{\alpha_x \alpha_z}{\|\vec{\alpha}\|^2} C_a \frac{\Delta t^2}{4} \end{bmatrix}, \tag{4.34}$$

where

$$\begin{aligned} S_a &= \sin\left(\frac{\|\vec{\alpha}\| \Delta t^2}{4}\right) \\ C_a &= \cos\left(\frac{\|\vec{\alpha}\| \Delta t}{4}\right). \end{aligned}$$

The EKF process noise covariance $\mathbf{Q}_v(k-1)$ is defined in Eq. 4.15, but for this motion model $\frac{\partial \hat{\mathbf{x}}(k|k-1)}{\partial \vec{n}}$ equals:

$$\frac{\partial \hat{\mathbf{x}}(k|k-1)}{\partial \vec{n}} = \begin{bmatrix} \frac{\partial \mathbf{r}}{\partial A} & \frac{\partial \mathbf{r}}{\partial \Psi} \\ \frac{\partial \mathbf{q}}{\partial A} & \frac{\partial \mathbf{q}}{\partial \Psi} \\ \frac{\partial v}{\partial A} & \frac{\partial v}{\partial \Psi} \\ \frac{\partial \omega}{\partial A} & \frac{\partial \omega}{\partial \Psi} \\ \frac{\partial \dot{a}}{\partial A} & \frac{\partial \dot{a}}{\partial \Psi} \\ \frac{\partial \dot{\alpha}}{\partial A} & \frac{\partial \dot{\alpha}}{\partial \Psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \mathbf{I}_{3 \times 3} \Delta t^2 & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{4 \times 3} & \frac{\partial \mathbf{q}(k|k-1)}{\partial \Psi} \\ \mathbf{I}_{3 \times 3} \Delta t & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \Delta t \\ \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} \end{bmatrix}. \quad (4.35)$$

4.5.3 Motion Model Parameter

Choosing EKF process noise \mathbf{Q}_v is an important part of the filter deployment. The only part of \mathbf{Q}_v , which can be parameterised is the covariance matrix \mathbf{P}_n of the noise vector \vec{n} . This matrix is diagonal as required when the linear and angular components are uncorrelated.

In MonoSLAM using the constant linear and angular velocity motion model the covariance noise matrix \mathbf{P}_n is

$$\mathbf{P}_n = \begin{bmatrix} SD_a^2 \Delta t^2 & 0 \\ 0 & SD_\alpha^2 \Delta t^2 \end{bmatrix}, \quad (4.36)$$

where the standard deviation parameters (SD_a and SD_α) define the smoothness of motion we expect.

If the *constant linear and angular acceleration* motion model is applied, the covariance noise matrix \mathbf{P}_n is

$$\mathbf{P}_n = \begin{bmatrix} SD_j^2 \Delta t^2 & 0 \\ 0 & SD_\eta^2 \Delta t^2 \end{bmatrix}, \quad (4.37)$$

and standard deviation parameters (SD_j and SD_η) again define the kind of motion we assume.

If the camera with fast readout speed (e.g. Δt is 5ms) is used and the noise covariance matrix \mathbf{P}_n contains the same values as when using the standard camera (Δt is 33ms), the motion model expects very smooth motion. To increase the robustness of localisation using the high-speed camera, it set so the \mathbf{P}_n parameters to larger values in both motion models is sufficient. An example is displayed in Fig. 4.11.

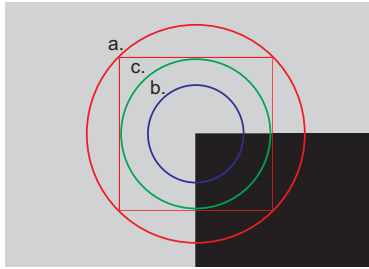


Figure 4.11: To update the camera pose in each time step, known scene feature descriptors (square) are matched with predicted features' positions (a). Localisation using the camera with fast readout speed (e.g. Δt is equal 5ms) causes these predicted feature positions to shrink obviously (b). To handle more erratic movements the motion noise uncertainty \mathbf{P}_n should contain larger values, which cause the feature search regions to enlarge (c).

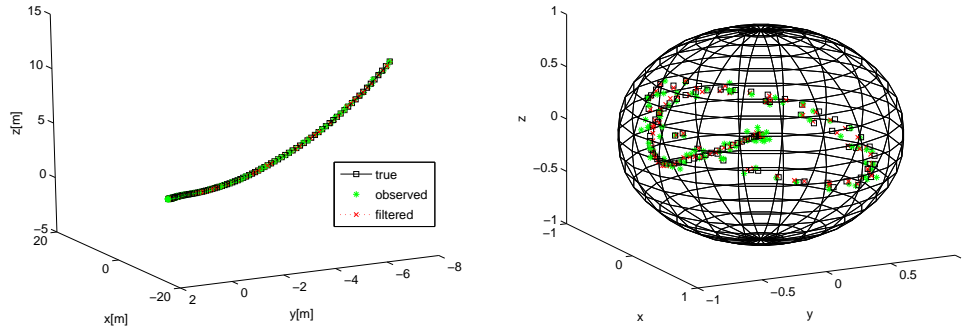


Figure 4.12: Simulated results of the constant linear and angular acceleration model. Left are the camera true, observed and estimated positions. Right are the camera orientations depicted as unit quaternions (see Sec. 2.2.2).

4.5.3.1 Example: Motion Model Simulation

A simulated example of the introduced constant linear and angular acceleration model is depicted in Fig. 4.12. The motion profile displays extreme translation and rotation.

The estimated state $\hat{\mathbf{x}}_v$ proves to be robust and accurate enough for this motion. We assume that the extended motion model is sufficiently robust for real-time experiments. Several examples of real-time experiments will be explained in the next section.

4.5.4 Skipping Vision Information

The standard camera (Δt is 33ms) is typically providing a fairly stable vision input, and missing frames are unlikely to occur. However, the asynchronous high-speed camera (e.g. Δt is 5ms) is not so reliable. A skipped or missing frame causes the Δt to increase

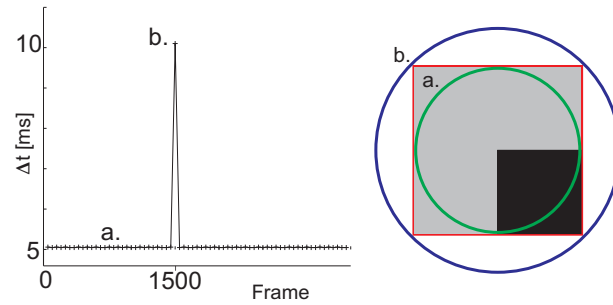


Figure 4.13: The asynchronous high-speed camera can not guarantee constant time between consecutive frames. We expect that the readout speed unpredictably decreases for some frames (left image). This Δt diminution causes the features' searching ellipses to grow (right image).

Constant Velocity Motion Model		
$\Delta t [ms]$	Translation [m/s^2]	Rotation [rad/s^2]
33	4	6
5	40	60
Constant Acceleration Motion Model		
$\Delta t [ms]$	Translation [m/s^3]	Rotation [rad/s^3]
33	40	60
5	4000	6000

Table 4.1: Standard deviation parameters for both motion models.

and this will have an effect on enlarged searching regions as depicted in Fig. 4.13.

To handle erratic motion using the high-speed camera the values of the noise covariance matrix \mathbf{P}_n (for both motion models) should be adjusted to large values. However, it is suitable to keep a balance between the motion model robustness and the real-time demand. Larger \mathbf{P}_n values require to search for feature matches in larger regions and this is a more time consuming operation. The \mathbf{P}_n values have been found empirically, and are showed in Tab. 4.1.

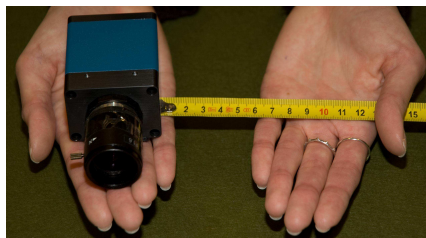


Figure 4.14: The gigabit ethernet camera has been thrown and caught for a distance approximately equal 0.1m.



Figure 4.15: The setup for the controlled experiments consists of the gigabit ethernet camera mounted onto a 7DOF robotic arm.

4.6 Experimental Results

To prove the improved robustness of MonoSLAM using the high-speed camera, these experiments are introduced:

- repeated, controlled and precise robotic arm movements have been executed,
- the camera has been thrown and caught and
- handheld jitter motions in different world frame directions have been performed.

Currently, there are several industrial high-speed cameras, and there is no need to develop a custom product. These cameras have the same measurement principle as the perspective-projective camera (see Sec. 2.2.3) used in the vision SLAM framework. The only difference is the communication interface. An example of the high-speed camera is displayed in Fig. 4.14.

4.6.1 Setup

The setup for the controlled experiments consists of: the commercial Amtec¹ 7DOF robotic arm (see Fig. 4.15), a robotics path planning software package² and the high-speed camera. The arm has been used to perform an accurate, pre-defined and smooth movement, which was repeated with different camera readout speeds.

For the two types of handheld experiments the high-speed camera has been needed. Firstly, the camera has been shortly thrown and caught (see Fig. 4.14). Secondly, a movement with higher dynamics has been performed to compare the motion models.

In this work, we have been using a very fast GE680C Prosilica³ gigabit ethernet camera (see Fig. 4.15 and Fig. 4.14), which - thanks to the 1/3" CCD sensor - offers a very good image quality.

¹www.amtec-robotics.com

²www.amrose.dk

³www.prosilica.com

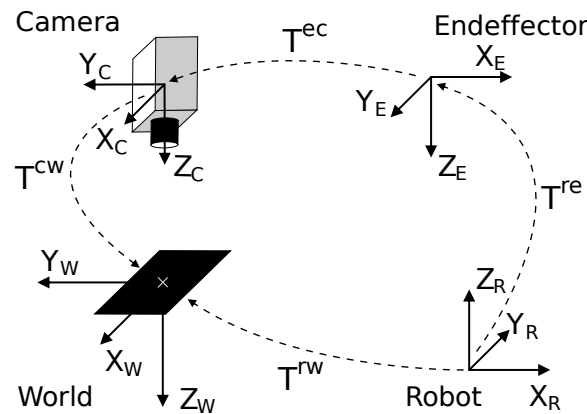


Figure 4.16: Scheme of coordination frames, which has been used in the robotic arm experiments.

This gigabit ethernet camera can provide VGA resolution vision output at 200Hz. In MonoSLAM, 320x240px images are more suitable, because VGA resolution would require more computationally intensive vision processing. This camera can resize the image on the chip, and this allows to address smaller images directly, faster, and at *requested* frame rates.

4.6.2 Ground-truth Computation

To precisely evaluate experiments with MonoSLAM motion models and the high-speed camera, we need accurate *localisation* and *mapping* ground truth. For the computation of accurate ground truth we used two offline, iterative and *computationally intensive* algorithms from the field of computer vision. The first is a camera pose algorithm presented in Sec. 2.2.6. This algorithm needs an object with known geometry (black rectangle in Fig. 4.16), which is visible in every frame. The second is a structure reconstruction algorithm based on nonlinear optimisation, summarised by Triggs et al. in [100]. An example of the scene reconstruction computed with the combination of these two algorithms is introduced in the next chapter.

Before we start with the explanation of the performed experiments, these were our assumptions:

- constant lighting conditions,
- calibrated camera,
- calibrated robotic arm,
- one object with known geometry,
- rigid scene (no moving objects) and
- scene contains textured objects.

As explained in Sec. 2.2.1, a camera pose is defined by the rotation (e.g. rotation matrix \mathbf{R}) and the translation \mathbf{t} . These two parts of the pose form a transformation matrix \mathbf{T} , commonly known as *Euclidean* transformation [67] :

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}. \quad (4.38)$$

The coordination frames of the experimental setup are depicted in Fig. 4.16, where only two Euclidean transformations are unknown. The first is the camera pose \mathbf{T}^{ec} wrt. to the robot endeffector. The second unknown Euclidean transformation is the object pose \mathbf{T}^{rw} wrt. to the robot frame. Both of these transformations remain constant during the experiments.

To find the optimal \mathbf{T}^{ec} and \mathbf{T}^{rw} values, this calibration procedure has been suggested:

1. obtain several images of the known object and simultaneously record the poses of the robot endeffector \mathbf{T}^{re} ,
2. calculate the poses (see Sec. 2.2.6) of the camera wrt. to the known object \mathbf{T}^{cw} ,
3. estimate initial unknown poses \mathbf{T}^{ec} and \mathbf{T}^{rw} ,
4. optimise the unknown poses using computed camera and recorded endeffector poses.

As depicted in Fig. 4.16, \mathbf{T}^{rw} equals:

$$\mathbf{T}^{rw} = \mathbf{T}^{re} \mathbf{T}^{ec} \mathbf{T}^{cw}, \quad (4.39)$$

and the optimisation error is defined as:

$$e = \sum_{i=1}^n \|\mathbf{T}^{rw} - \mathbf{T}^{re}_i \mathbf{T}^{ec} \mathbf{T}^{cw}_i\|. \quad (4.40)$$

The Levenberg-Marquardt optimisation procedure [68] has been used to find the unknown \mathbf{T}^{rw} and \mathbf{T}^{ec} . The minimisation function is presented in Eq. 4.40. The unknown Euclidean transformations has not been represented in the minimization procedure as matrices, but as vectors e.g.:

$$x_0 = [\mathbf{t}^{ec}, \mathbf{q}^{ec}, \mathbf{t}^{rw}, \mathbf{q}^{rw}], \quad (4.41)$$

where a non-linear function computing a rotation matrix from a quaternion can be found here [18].

4.6.2.1 Ground Truth Accuracy

To present the ground truth accuracy, an example of a comparison of four reconstructed feature positions and their known values is given in Tab. 4.2. This shows that the estimation using the procedure described in Section 4.6.2 is accurate to within 0.1 percent of the true values, which refers to an estimate of better than 0.1mm.

	True			Reconstructed		
	X[m]	Y[m]	Z[m]	X[m]	Y[m]	Z[m]
1	0.105	0.07425	0.0	0.10497	0.07428	-0.00010
2	-0.105	0.07425	0.0	-0.10499	0.07429	-0.00013
3	0.105	-0.07425	0.0	0.10502	-0.07428	0.00008
4	-0.105	-0.07425	0.0	-0.10502	-0.07426	-0.00005

Table 4.2: Three-dimensional features’ true vs. reconstructed positions. The root mean square error (RMSE) of the reconstructed positions is smaller than 0.1mm.

4.6.3 Evaluations

This section describes the localisation and mapping evaluations, when different real-time motion patterns have been tested. These tests should prove the results of the simulation (see Sec. 4.5.3.1), and the suitability of the extended motion model for high-speed applications.

4.6.3.1 Evaluation of Localisation Accuracy

Prior to the evaluation of MonoSLAM *localisation* results against the ground truth we explain the used criteria and present the motion model parameters.

The criteria to compare the true and estimated camera poses are:

- the root mean square error (RMSE) to calculate the deviations of *positions* and
- the RMSE to compute the discrepancies of *rotations*.

For comparison purposes the camera rotation representation is transformed from quaternions to Roll-Pitch-Yaw (RPY) radians.

An overview of both MonoSLAM motion model parameters is given in Tab. 4.1. These parameters have been adjusted according to the Sec. 4.5.3 and 4.5.4.

Robotic Arm Movements When performing smooth robotic arm movements using 5ms camera readout speed (Δt), both motion models were providing robust results as depicted in Fig. 4.17. The position RMSE was 0.227m and the rotation RMSE was 0.2922rad, when using the constant linear and angular velocity motion model. However, the constant linear and angular acceleration motion model provided less accurate results: the position RMSE was 1.461m and the rotation RMSE was 1.9595rad. An overview of the localisation accuracies is given in Tab. 4.3. The results in this Table indicate that the extended motion model is less accurate.

An important *localisation* advantage when using the fast vision readout speed is that the uncertainties of the camera poses are obviously smaller as depicted in Fig. 4.18.

The MonoSLAM output quaternion is \mathbf{q}_i^m and \mathbf{q}_i^{gt} represents the ground truth quaternion. The error in quaternions has been calculated as presented in Eq. 2.12.

Constant Velocity Motion Model		
Δt [ms]	Translation[m]	Rotation
33	0.054	0.908
5	0.227	19.819
Constant Acceleration Motion Model		
Δt [ms]	Translation[m]	Rotation
33	0.523	11.998
5	1.461	104.071

Table 4.3: The evaluated localisation accuracy.

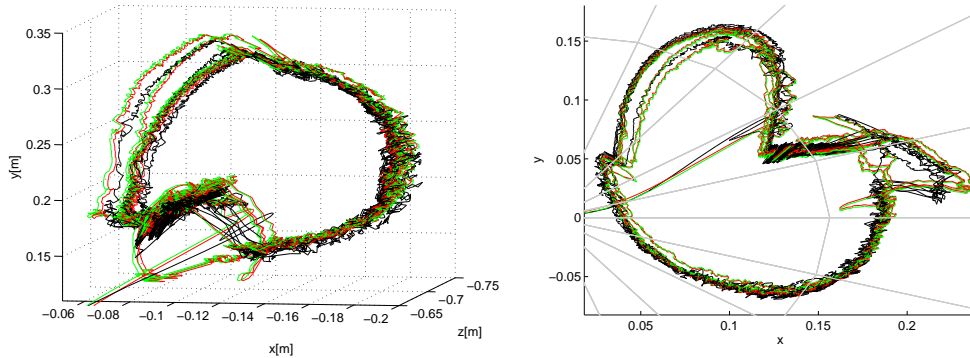


Figure 4.17: Three loops of the controlled movement, using the high-speed camera (Δt equals 5ms) carried by the robotic arm. Left are the three-dimensional camera world frame positions and right are the camera rotations. The rotations in MonoSLAM are represented as normed quaternions, which can be visualised in a unit sphere (see Sec. 2.2.2). The red lines are the estimated camera poses using the constant linear and angular velocity motion model, and the green lines are the computed ground truth. The black lines are the estimated camera poses using the constant linear and angular acceleration motion model.

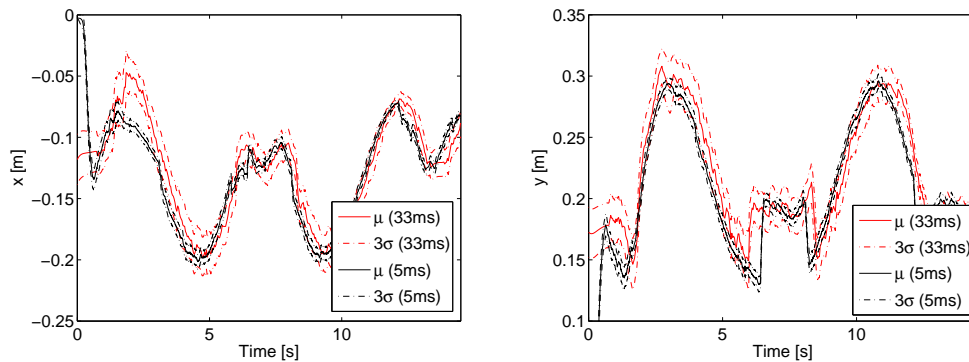


Figure 4.18: The *localisation* uncertainties using two different vision readout speeds (Δt): 33ms and 5ms. The camera performed the same movements repeatedly as depicted in Fig. 4.17, but only the camera world frame positions of the first 15s are displayed here.

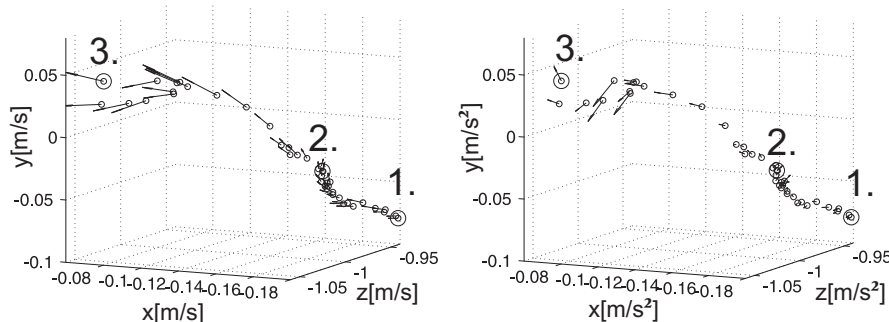


Figure 4.19: The Euclidean distance between the initial (1.) and final (3.) camera positions is equal 0.175m. The high-speed camera has been thrown (2.) and caught (3.) over a distance of approximately 0.1m. In MonoSLAM, the *constant linear and angular acceleration* motion model has been used. The velocity (left) and acceleration vectors (right) are scaled.

Thrown Camera The setup for camera throwing and catching is displayed in Fig. 4.14, where the distance between two hands was approximately 0.1m. This experiment has been performed repeatedly. The result is that MonoSLAM can operate in real-time capturing both the free as well as high accelerations in the throwing and catching motions.

As an example, Fig. 4.19 displays the individual track points and for each point the estimated motion using the constant linear and angular acceleration model. The throwing of the camera does not contain significant jitter, so the accuracy differences between the two motion models are similar to the robotic arm movements. Again the initialisation used is a black rectangular object (see 4.2-b) in the first field of view.

High Acceleration Handheld Motion Finally, we want to demonstrate the behavior for fast and shaky motion. Fig. 4.20 depicts the motion following rapid up and down movements of the camera in hand. Maximum frequency of the up/down motion is about 5Hz. The graphs show the MonoSLAM reconstruction using the two motion models. As the Figure indicates, the constant linear and angular acceleration model (Δt equals 5ms) successfully tracks such a motion while the constant linear and angular velocity model first shows considerably high inaccuracy and then loses track (see Tab. 4.4).

4.6.3.2 Mapping Evaluations

MonoSLAM using high-speed visual information can faster converge feature location uncertainties as depicted in Fig. 4.21.

MonoSLAM's known problem is that the processing time associated with the EKF update is $O(n^2)$, where n is the number of features in the map. Due to the real-time demand all experiments in this paper have been performed in a small scene with

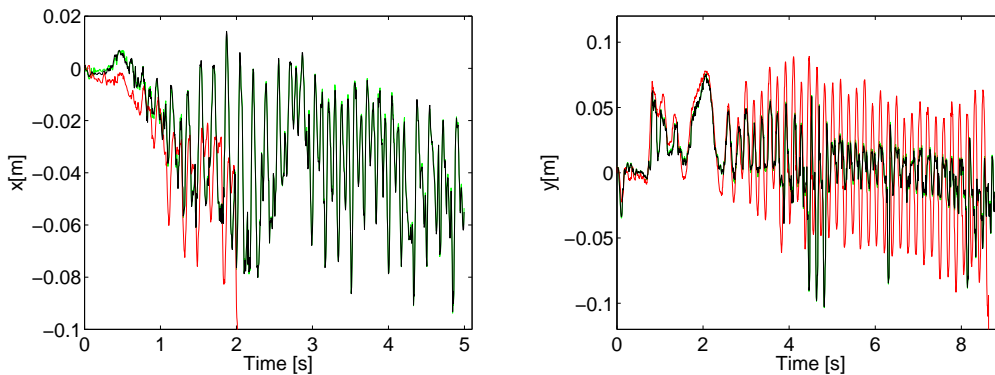


Figure 4.20: In the upper row a handheld jitter motion has been performed in world x direction, and the camera world frame x positions are depicted. In the bottom row a jitter movement in world y direction is displayed as camera y positions. The *constant linear and angular velocity* model (red lines) lost the camera pose during both motions (time equals 2s in upper row and short after 8s in bottom row). The *constant linear and angular acceleration* model (black lines) proved to be more robust to these movements. The green lines are the computed ground truth.

Type	Velocity model errors		Acceleration model errors	
	translation [m]	rotation	translation [m]	rotation
Catch/Throw	0.237	10.927	1.767	73.581
Catch/Throw	0.090	3.636	0.301	12.071
Jitter in x^W	16.656	1054.800	0.088	3.135
Jitter in y^W	6.567	143.946	0.085	3.653

Table 4.4: Handheld motion. Δt is equal 0.005s.

approximately 20 features. MonoSLAM has been able to operate in such an environment in real-time using the high frame rate vision input (Δt equals 5ms).

An overview of the mapping accuracy is given in Tab. 4.5, where both readout speeds deliver similar results.

4.7 Conclusion

This chapter presents these contributions: the introduction of a high-speed camera to monocular SLAM and a combination of this fast readout speed with a second order motion model to allow MonoSLAM operate under motions with significant jitter. Such a MonoSLAM algorithm running at a frame rate of 200Hz was implemented and investigated. We performed repeated and controlled experiments with a robot arm and various handheld motion. The comparison of the camera poses is made against accurate ground truth from off-line vision. As expected the constant velocity model is better

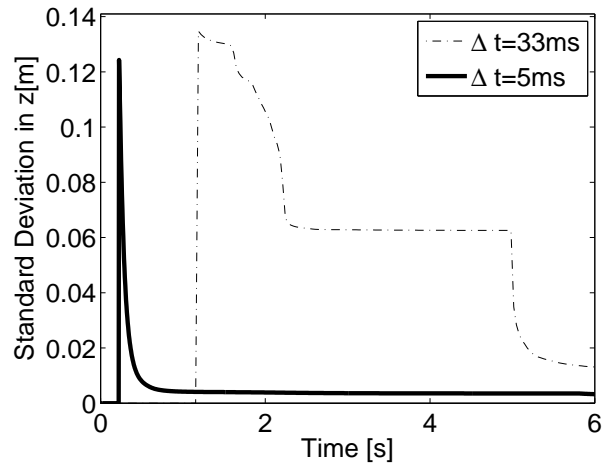


Figure 4.21: In MonoSLAM, the uncertainty of a feature location is decreasing as presented in [30]. If the fast readout speed is used, then the feature three-dimensional position uncertainty is converging several times faster.

Feature	Mapping Accuracy	
	$\Delta t=33\text{ms}$ [m]	$\Delta t=5\text{ms}$ [m]
1	0.0044	0.0058
2	0.0096	0.0075
3	0.0015	0.0073
4	0.0082	0.0004
5	0.0025	0.0039
6	0.0104	0.0123
7	0.0002	0.0032
8	0.0003	0.0085

Table 4.5: Mapping accuracy.

suited for rather smooth motions. In handheld motions with an up and down motion at a frequency of about 5Hz, the acceleration model is superior and successfully tracks throughout the sequence. Additionally, the high frame rate operation of MonoSLAM makes it possible to throw the camera from one hand to the other.

By showing that high frame rate is able to considerably improve the localization robustness of monocular SLAM, new applications become possible. For example, in a future project it will be evaluated in consumer robotics and wearable computing applications.

Future ideas of enhancing the real-time performance of MonoSLAM include:

- Davison presented in [26] a more efficient matching approach. This has been achieved using methods from information theory. This approach would be very attractive for high-speed MonoSLAM, because we can select features based on

their mutual information.

- An interesting work of Clemente et al. in [21] would help significantly to improve the mapping real-time performance of MonoSLAM. As introduced by the authors, this improvement can be achieved using local coordinate frames. Due to this mapping strategy the computational complexity can be reduced from current $O(n^2)$ to $O(n)$.
- Recently, Civera et al. [20] presented a Bayesian Interacting Multiple Models (IMM) for MonoSLAM. This IMM framework was successfully tested on sequences containing: no motion, pure rotation and general motion with various dynamics. We think that the extended motion model introduced in this chapter could be integrated into this framework and could increase the localisation robustness of MonoSLAM. However, the disadvantage of IMM is the higher computational burden because all filtering operations have to be duplicated for each model.

Good Corners for Structure and Motion Recovery

When mapping an unknown scene using a perspective-projective camera, the camera has to move to observe landmarks from different viewpoints. Due to this motion between subsequent frames, the *correspondence* has to be maintained. However, one can not predict if a new landmark lies on a false T-junction. If we detect features lying on an occlusion boundary, or other so-called *false* features, these can be rejected in MonoSLAM only after several time steps [27]. Inserting more false features into the map of the scene usually causes the SLAM framework to collapse.

To prove the influence of false landmarks on the accuracy of the final map, we present a simulated example. In this example the estimated 3D positions of several landmarks are compared with the truth values. The estimated 3D positions were computed using a non-linear optimisation technique known as *Bundle Adjustment* [100]. Our results show how the accuracy of the final map diverges with the increasing count of false features.

To avoid inserting false features into the map of the environment, we tested a recently introduced ToF camera. This ToF camera belongs to the category of the so-called Range Imaging sensors (RIM). These sensors can acquire a digital image of the scene and for each pixel deliver the distance measurement. These parallelly acquired measurements are synchronised and spatially aligned in the camera coordination frame. The distance measurements of ToF camera can be based on the *direct* or the *indirect* measurement principle [53]. Our ToF camera uses the indirect measurement principle, which has roots in the theory of the amplitude-modulated light.

The ToF camera is a very appealing sensor for mobile robotics. As argued in Sec. 1.3, a vision sensor provides rich scene information, which is helpful for the localisation and mapping tasks. Additional distance measurements would ease the task for the mobile robot in many real-time applications. ToF camera capable of measuring digital image data with correlated distances in higher-resolution is still not available, but is a topic of intensive applied research. We think that other fields in robotics or computer vision would certainly benefit from this product.

This chapter is divided into six sections. The related work is presented in Sec. 5.1 and the contribution is outlined in Sec. 5.2. The Bundle Adjustment is explained in

Sec. 5.3 and Sec. 5.4 introduces the corner selection scheme. The experimental results are introduced in Sec. 5.5 and the conclusion including the future work is presented in Sec. 5.6.

5.1 Related Work

Due to the fact that the ToF measuring principle has been introduced in robotics only recently, there are few papers exploiting the ToF camera.

May et al. in [69] presented results using a ToF camera when estimating parallelly an unknown scene and the camera poses. Their framework is based on two major steps, and is not designed to work on-line. These two steps are: the pose estimation and the 3D mapping. The pose estimation uses the Iterative Closest Point (ICP) algorithm, which takes advantage of parallelly acquired 2D and 3D data. The 3D mapping is a three-stage process, which refines and smoothes the surfaces. The pose estimation accuracy result of the accumulated translational error is on average an equal 140mm, when comparing different 2D feature extraction methods (SIFT, KLT, etc.).

Recently, May et al. introduced in [70] an improved version of their 3D mapping approach. The authors also presented a more accurate distance calibration method, where the ToF camera light scattering effect is tackled.

Bearing-Only SLAM [27, 82, 31] and SfM [77, 81, 78, 56] approaches can not infer the depth information of the scene from a single view due to the perspective-projective nature of the camera. To initialize new landmarks' positions, different views of the scene are usually needed, and in Bearing-Only SLAM [27] this is known as delayed initialisation. However, in [92] an undelayed initialisation method is proposed, however only simulated results are introduced. Despite these efforts, landmarks' initialisation in Bearing-Only SLAM and SfM is still a challenging task, because if these new landmarks do not represent distinct 3D corners and lie at e.g. occlusion boundaries, it can lead to SLAM [27] failure or SfM [77] inaccuracies.

An alternative approach of how to use the depth ToF measurements to initialize new landmarks could be Range-Bearing SLAM, e.g. [80]. This SLAM algorithm only takes advantage of the depth geometric information. However, the contribution of our work focuses on how to combine the amplitude and depth ToF measurements to select good scene corners.

A comprehensive overview of the ToF principle and camera which has been used in this work can be found in [53].

5.2 Contribution

Introducing false scene landmarks, lying at, e.g., occlusion boundaries or curved surfaces, to SLAM or SfM framework causes, e.g., inaccuracies or loss of position. To prove this statement a virtual SfM example is presented, where the dependence of the count of false features versus inaccuracy of projection is computed.

To improve the accuracy and robustness of the final map, a ToF camera capable of measuring the scene amplitude and depth directly has been used. Due to the available

depth data the geometry of the visible scene can be inferred. We presented a three-stage selection scheme capable of pruning false landmarks. The first stage of the presented selection scheme uses a well-known corner detector applied on the amplitude data. The second stage analyses the geometry of the depth measurements, and decides if the landmark is a valid three-dimensional corner. If necessary, the third stage verifies using RANSAC [36] if the corner lies in a plane.

To our knowledge, this is the first use of such an approach to locate 3D corners in the scene. We published this approach result recently in [41].

5.3 Bundle Adjustment

To explain how the false landmarks influence the accuracy of the recovered map, we divided this section into three parts. Firstly, a description of the pinhole camera model is provided. Secondly, the non-linear optimisation method used in the bundle adjustment algorithm is introduced. Thirdly, a simulated example of how false scene corners influence the accuracy of the final map is presented.

5.3.1 The Perspective-Projective Camera Model

The perspective-projective, also known as the pinhole camera, maps 3D landmarks into a 2D image plane. This mapping, in our case *central projection* \mathbf{P}_i [45, 43], has the following form:

$$\lambda_{ij}\mathbf{x}_{ij} = \mathbf{P}_i\mathbf{X}_j, \quad i = 1 \dots m, \quad j = 1 \dots n, \quad (5.1)$$

where \mathbf{X}_j are the homogeneous 3D landmarks, \mathbf{x}_{ij} are the homogeneous 2D points and λ_{ij} are the projective depths.

The central projection \mathbf{P}_i is a part of this equation:

$$\lambda_{ij} \underbrace{\begin{bmatrix} x_x \\ x_y \\ 1 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} f & sf & x_0 \\ 0 & af & y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{[\mathbf{R} | -\mathbf{R}\mathbf{t}]}_{\mathbf{P}} \underbrace{\begin{bmatrix} X_x \\ X_y \\ X_z \\ 1 \end{bmatrix}}_{\mathbf{X}}, \quad (5.2)$$

where \mathbf{K} is the calibration matrix. The camera pose is defined by \mathbf{R} and \mathbf{t} (see Sec. 2.2.1). For further details considering the perspective-projective camera model please refer to Sec. 2.2.3.

5.3.2 Iterative Non-linear Optimisation

It is generally considered that there exist two groups of SfM algorithms. The first group are known as *sequential* algorithms (e.g. [77]) that try to estimate unknown scene and camera parameters as soon as new data is available. The other group called *batch* or

bundle adjustment algorithms [43] optimises the unknown parameters, when the data acquisition is complete.

As mentioned in Sec. 4.4.1, the problem of sequential SfM algorithms is that the reconstruction error accumulates along the process. Beyond a certain point, i.e., for longer sequences, the accumulated error might be unrecoverable [43]. However, the batch algorithms can overcome local ambiguities, since the problem is solved globally. In this section, to generate truth values of the camera pose and the scene 3D features, the batch algorithm has been used.

The difference between a Bearing-Only SLAM technique like MonoSLAM (ref Sec. 4.4) and bundle adjustment is that MonoSLAM recovers a rather sparse map of the scene in real-time. An example is displayed in Fig. 5.1. When the camera moves, new distinctive features can be detected (e.g. corners with numbers 5-10 in Fig. 5.1-b and Fig. 5.1-c). The output of MonoSLAM are the camera poses and a sparse map of recovered features, as depicted in Fig. 5.1-c. Due to the real-time demand, *mapping* in MonoSLAM does not play a crucial role and should rather support *localisation*.

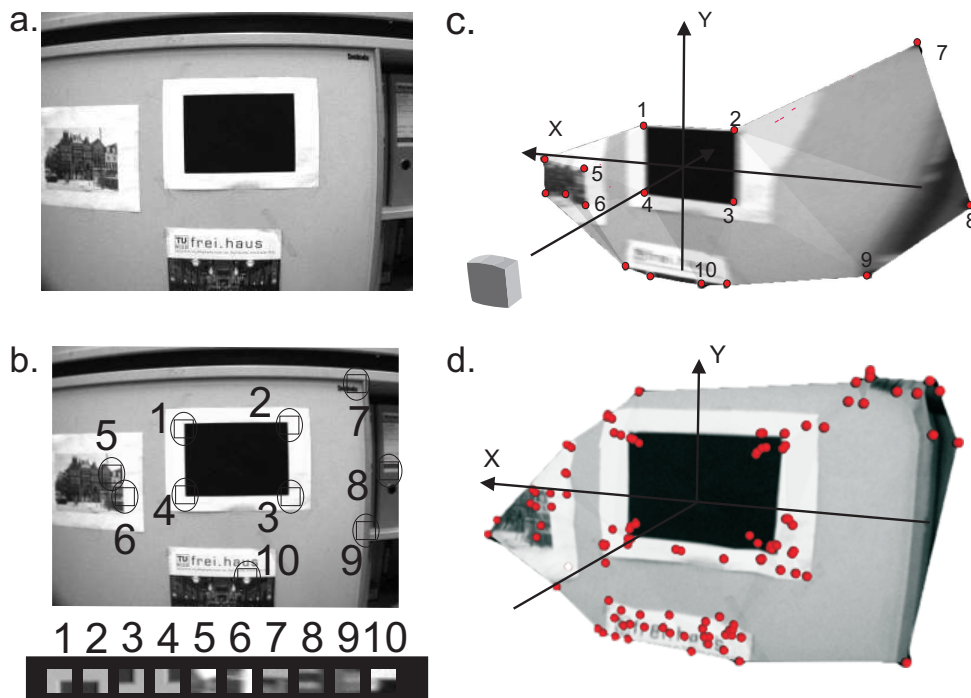


Figure 5.1: The only sensor input in MonoSLAM are images from a single camera (a). As the camera moves, new distinctive features are detected (b). The output of MonoSLAM is the camera pose and a sparse, three-dimensional map of these distinctive features (c). An alternative to MonoSLAM sparse mapping is a more dense structure reconstruction using bundle adjustment (d).

The recovery of an unknown structure and motion can be solved when this problem is formulated as Maximum Likelihood Estimation [43]:

$$L = \max \prod_{i,j \in I} e^{|\mathbf{x}_{ij} - g(\mathbf{P}_i, \mathbf{X}_j)|^2 / 2\sigma^2} , \quad (5.3)$$

where $g(\mathbf{P}_i, \mathbf{X}_j)$ represents the re-projected landmarks.

A standard reformulation of this maximisation problem is to minimise the negative logarithm of the likelihood function in Eq. 5.3:

$$f = \sum_{i,j \in I} (\mathbf{x}_{ij} - g(\mathbf{P}_i, \mathbf{X}_j))^2 . \quad (5.4)$$

The advantage of this reformulation is that this problem can be minimised as a Euclidean distance between measured (\mathbf{x}_{ij}) and modelled (\mathbf{X}_j) scene points. The minimisation of f is usually computed using iterative non-linear methods.

The 2D re-projections \mathbf{x}_{ij} of 3D corners \mathbf{X}_j are calculated as:

$$\mathbf{x}_{ij} = g(\mathbf{K}_i, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j) . \quad (5.5)$$

Central projection \mathbf{P}_j in (5.1) encompasses the same camera parameters (\mathbf{K}_i , \mathbf{R}_i and \mathbf{t}_i), where \mathbf{K}_i are the intrinsic parameters, \mathbf{R}_i is the orientation and \mathbf{t}_i is the position.

The task of bundle adjustment is to minimise the deviation between 2D measured and re-projected points in the L^2 norm. The formulation is to calculate the:

$$\min_{\mathbf{K}_i, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j} \sum_{ij} (\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij})^2 . \quad (5.6)$$

This solution is optimal in a statistical sense, i.e. when the coordinates of the measured image points are corrupted by zero mean and equal standard deviation Gaussian noise. A Gauss-Newton method is used here to find the minimum solution.

Introduce a residual vector \mathbf{Y} , formed by putting all deviations between measured and re-projected image coordinates in a column vector. The sum of $f = \mathbf{Y}^T \mathbf{Y}$ is minimised w.r.t. to the unknown parameters $\Delta \mathbf{x}$. A linearisation of $\mathbf{Y}(\Delta \mathbf{x})$ gives:

$$\mathbf{Y}(\Delta \mathbf{x}) \approx \mathbf{Y}(0) + \frac{\partial \mathbf{Y}}{\partial \Delta \mathbf{x}} \Delta \mathbf{x} \quad (5.7)$$

Then $\Delta \mathbf{x}$ equals:

$$\begin{aligned} \Delta \mathbf{x} &= -(\mathbf{A}^T \mathbf{A} + \epsilon I)^{-1} \mathbf{A}^T \mathbf{b}, \\ \mathbf{A} &= \frac{\partial \mathbf{Y}}{\partial \Delta \mathbf{x}}(0), \\ \mathbf{b} &= \mathbf{Y}(0) \end{aligned} \quad (5.8)$$

In practice, to find a minimum to this formulation, the Levenberg-Marquardt iterative non-linear optimisation method is used.

For further camera geometry details and bundle adjustment explanations please refer to, e.g., [45, 43].

5.3.3 Example: Simulated Bundle Adjustment

As discussed in [32], the recovered scene using the bundle adjustment algorithm can be considered as the ground-truth.

To show how false corners influence the bundle adjustment results, an unknown structure of a simulated scene and virtual camera motion has been recovered. The simulated scene is depicted in Fig. 5.2.

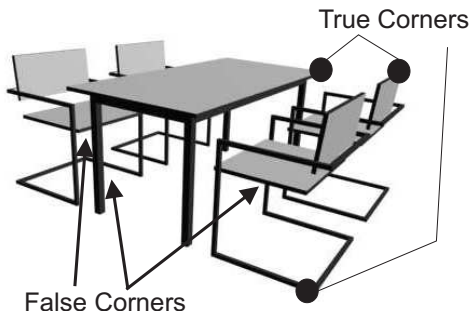


Figure 5.2: Our simulated scene contains one table and four chairs. In this bundle adjustment example, twenty-two good corners were used, but in this figure, only three of them are depicted as circles. We used up to three bad corners, which are marked with arrows.

The evaluation criterion is the re-projection error f as defined in Eq. 5.4. The empirical results are displayed in Fig. 5.3, where it is obvious that every additional false corner introduces inaccuracies to the re-projection errors.

5.4 Corner Selection Scheme

In real scenes it is obvious that there are usually features which do not represent true 3D points, e.g. lying at occlusion boundaries, caused by specular highlights or curved surfaces (see also example in Fig. 5.2). However, one of the underlying Bearing-Only SLAM or SfM assumptions is that the scene is rigid, which means that features are not expected to change their position within the scene. The map management implemented (e.g. in MonoSLAM) can prune some of these bad corners, but using the depth information a further feature validation can be performed.

Our proposed corner selection scheme contains three stages:

- detect the best 2D corners using the amplitude data,
- decide whether it is a true 3D point using the computed 3D cornerness measure, and

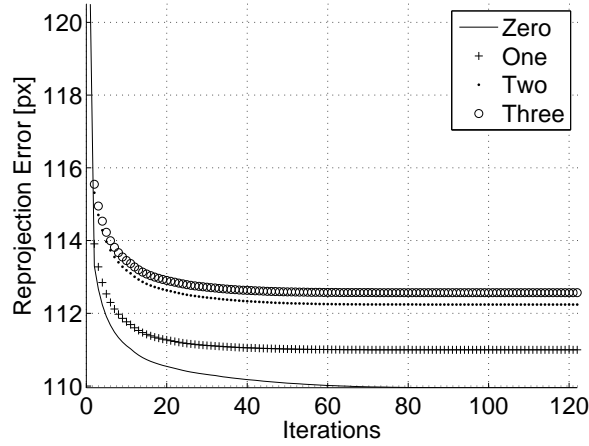


Figure 5.3: Simulated bundle adjustment reprojection errors using zero, one, two or three false corners. It can be seen that the addition of a false corner increases the reprojection errors.

- if not prove that this 2D point belongs to a planar surface.

In the first stage, the **Fast** feature detector [86] is applied to the amplitude data. The best **Fast** features are then found using the Shi and Tomasi [89] cornerness measure (ref to Sec. 2.1).

With the second stage we decide whether it is a good 3D corner. To evaluate this, the standard matrix of image partial gradients:

$$\mathbf{S}_i = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \quad (5.9)$$

is extended to three dimensions as proposed in [85]

$$\mathbf{S}_d = \begin{pmatrix} \sum D_x^2 & \sum D_x D_y & \sum D_x D_z \\ \sum D_x D_y & \sum D_y^2 & \sum D_y D_z \\ \sum D_x D_z & \sum D_y D_z & \sum D_z^2 \end{pmatrix}, \quad (5.10)$$

using the depth instead of the visual information. To compute the 3D cornerness measures, two operators have been compared. Rohr presented in [85] a survey on 3D operators, and we selected the following one:

$$Op(x, y, z) = \frac{\det \mathbf{S}_d}{\text{trace } \mathbf{S}_d} \rightarrow \max, \quad (5.11)$$

because it is related to the Shi and Tomasi [89] 2D cornerness measure. Another 3D cornerness measure mentioned by Arseneau in [4] is based on the eigen-decomposition of the \mathbf{S}_d matrix, where this matrix is called the *structure tensor*. If a corner does not pass this stage, it can still be a good corner, possibly lying in the plane. This is explained in detail in Sec. 5.4.1.

In the third stage, a plane is fitted to the depth data at the 2D corner position found using RANSAC (ref to Sec. 2.1.6) to test whether this is a planar feature. We propose fitting the plane using two criteria. The first is the distance of points to the fitted plane, and the second one is the angle between the direction of a new feature and the plane normal vector.

For the robust plane fit, the distance function between the plane \mathbf{P} and an array of measured depth points is computed as follows

$$\begin{aligned}\vec{n} &= (P_2 - P_1) \times (P_3 - P_1) \\ d_i &= |(\mathbf{X}_i - P_1) \cdot \vec{n}|, \quad i = 1 \dots m\end{aligned}$$

where the plane \mathbf{P} is a matrix defined column-wise by three points, \vec{n} is the plane normal vector, \mathbf{X}_i are the measured points, and d_i are the distances.

5.4.1 Structure Tensor

As explained in the previous section, the structure tensor matrix (ref Eq. 5.10) can be used to evaluate the measured 2D or 3D data. In this subsection, we introduce the computation and the properties of the structure tensor in more detail.

The computation of the structure tensor has three stages. In the first stage, we obtain the partial derivative information. However, there are several ways how to calculate the partial derivative information: simple differencing between neighbouring pixels, 'Difference of Gaussians', or using the average of absolute pixel difference. Arseneau in [4] uses the average of absolute pixel difference. The key idea here is that using any previously mentioned method the coherence structure tensor property can distinguish between the uniform and isotropic case [4]. During the second stage we selected the tensor form (ref Eq. 5.10) as presented in [4]. This means that we consider only the diagonal elements (D_x^2, D_y^2, D_z^2). In the last stage, the three eigenvalues and the three eigenvectors are calculated.

As depicted in Fig. 5.4(a), we generated an ideal 3D corner. The structure tensor computed from this simulated data has three identical gradient elements. The result of the eigen-decomposition of this tensor are three identical eigenvalues and three orthogonal eigenvectors as displayed in Fig. 5.4(b). According to Arseneau [4], every structure tensor with $\lambda_3 \gg 0$ (λ_3 is the third eigenvalue of \mathbf{S}_d in Eq. 5.10) has isotropic behaviour.

There are three possible results of the structure tensor eigen-decomposition:

- $\lambda_1 \gg 0$ and $\lambda_2 \gg 0$ and $\lambda_3 \gg 0$, which forms an ideal corner as mentioned in the text above.
- $(\lambda_1 \gg 0 \text{ and } \lambda_2 \gg 0)$ or $(\lambda_1 \gg 0 \text{ and } \lambda_3 \gg 0)$ or $(\lambda_2 \gg 0 \text{ and } \lambda_3 \gg 0)$, which would form a 3D circle after visualisation.
- $\lambda_1 \gg 0$ or $\lambda_2 \gg 0$ or $\lambda_3 \gg 0$, which depicts a 3D line.

These eigenvalues determine the underlying structure of the 3D data [4]. For example, $(\lambda_1 - \lambda_2) \gg 0$ indicates a surface element, where the eigenvector corresponding to λ_1 is the normal vector to the surface.

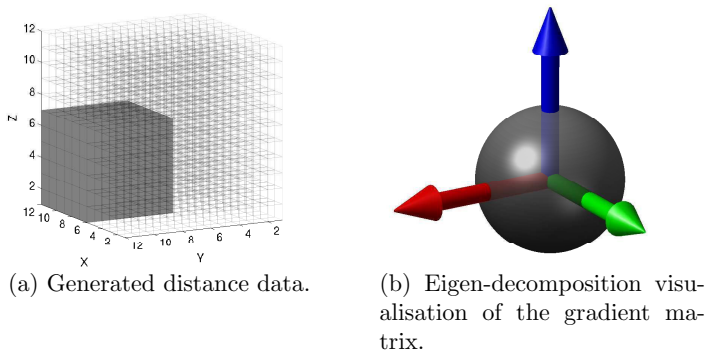


Figure 5.4: Generated ideal 3D corner and the visual interpretation of the structure tensor eigen-decomposition.

5.5 Experimental Results

In this section, we present our experimental setup as well as simulated and real data results in three parts. Firstly, the measurement principle as well as the properties of the ToF camera are briefly explained. Secondly, the relation between the amplitude and standard deviation in distance information is estimated. Thirdly, our proposed feature extraction scheme is evaluated.

5.5.1 ToF Camera

The ToF distance measurement process is divided into two categories [53]. The first category is known as the *direct* measurement principle. Direct means that the distance is determined by the time difference, which needs an emitted light pulse to return to the receiver. The second category is by analogy known as the *indirect* measurement principle. This principle uses more complex signals. The light can be modulated in amplitude, frequency, polarisation or phase.

In this work, we have been using the state of the art Swiss Ranger SR3000¹ ToF camera (ref Fig. 5.5(a)). The measurement principle of this camera is based on the indirect measurement principle. Swiss Ranger SR3000 uses amplitude-modulated near infrared (NIR) light, and this leads to the phase measurement principle as depicted in Fig. 5.5(b) and Fig. 5.6.

Using this principle the distance D is computed as:

$$D = L \left(\frac{\varphi}{2\pi} + N \right), \quad (5.12)$$

where $N = 0$ when measuring the depth up to 7.5m, and L equals:

$$L = \frac{c\lambda_{mod}}{2}. \quad (5.13)$$

The speed of light is c and λ_{mod} is the modulation wavelength.

¹Produced by Mesa Imaging - <http://www.mesa-imaging.ch>.

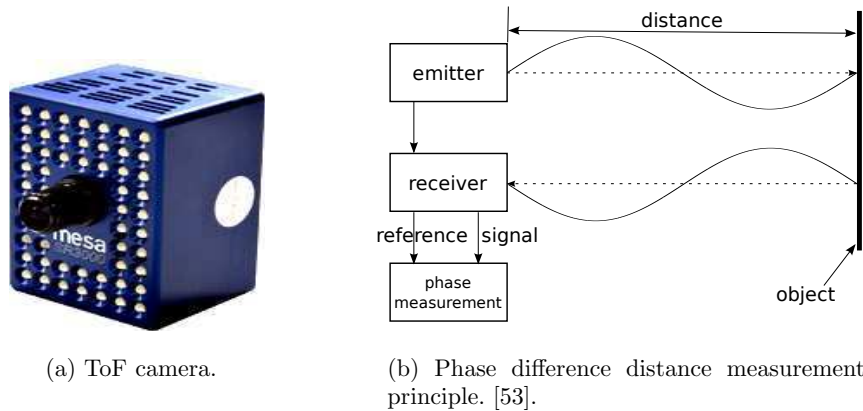


Figure 5.5: The Swiss Ranger SR3000 ToF camera (courtesy of Mesa Imaging - <http://www.mesa-imaging.ch>).

The φ is computed as:

$$\varphi = \tan^{-1} \frac{c(\tau_1) - c(\tau_3)}{c(\tau_0) - c(\tau_2)}, \quad (5.14)$$

where the $c(\tau_i)$ are the four measured intensities as depicted in Fig. 5.6.

The amplitude A is determined as:

$$A = \frac{\delta}{\Delta t \sin(\delta)} \frac{\sqrt{((c(\tau_3) - c(\tau_1))^2 + (c(\tau_0) - c(\tau_2))^2)}}{2}, \quad (5.15)$$

and offset B is calculated as:

$$B = \frac{c(\tau_0) + c(\tau_1) + c(\tau_2) + c(\tau_3)}{4}. \quad (5.16)$$

The δ equals:

$$\delta = \pi \frac{\Delta t}{T}. \quad (5.17)$$

This principle is complex and includes systematic (e.g. distance-related, amplitude-related, fixed pattern noise) and non-systematic (e.g. signal-to-noise ratios, rays reflection, light scattering) errors, which are further described in [53, 69]. The camera has a limited field of view (horizontal 47.5° , vertical 39.6°), a small resolution (176x144pix), and a depth range up to 7.5m.

The SR3000 ToF camera uses a perspective-projection camera model [113] for the amplitude data. As explained in [53], the distance measurements have been calibrated for every single pixel separately. The amplitude and distance calibration has been performed by the camera manufacturer.

Before capturing real data sequences, the camera integration time, which influences the measurement precision and read-out speed, has to be adjusted. In this paper, the camera integration time is adjusted to $7200\mu\text{s}$, and the read-out speed then equals

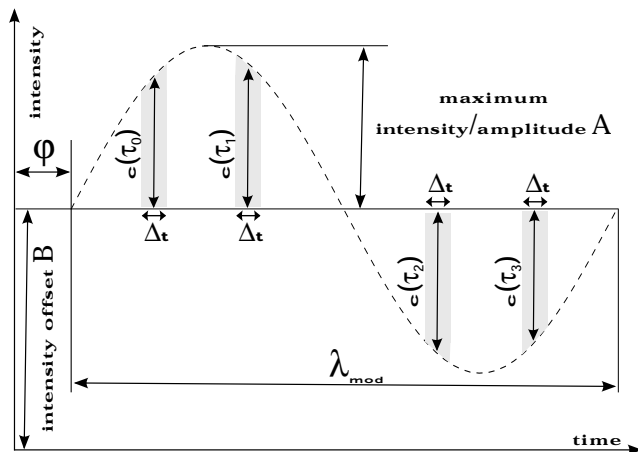


Figure 5.6: Phase measurement principle [53]. The not displayed emitted wave has phase $\varphi = 0$. The incoming wave is shifted in phase, and the demodulation is based on four intensity values.

approximately 21Hz. A faster read-out speed would be suitable to reduce the motion blur. However, a faster read-out leads to a higher signal-to-noise ratio (SNR), and this introduces inaccuracies in feature extraction [50].

The ToF camera is calibrated by the producer, so the intrinsic parameters (optical centre and focal length) are known, and only the distortion coefficient needs to be adjusted. The distance-related error calibration is also provided by the producer, and it is done by reducing the measurement offset using an acquired Fixed Pattern Noise matrix [53].

5.5.2 Static Depth Noise Modelling

To estimate the relation between the amplitude and the standard deviation in depth measurements as proposed by Kahlmann in [53], we perceived multiple measurements of a white wall using several integration times (from $200\mu\text{s}$ to $9800\mu\text{s}$). The obtained relation between amplitude and standard deviation in distance measurement for our camera is depicted in Fig. 5.7.

5.5.3 Feature Extraction

To prevent inserting bad corners into the model of the scene, we have implemented three validation stages, as explained in Sec. 5.4. This sub-section presents the experimental results using these three stages.

5.5.3.1 2D Corner Extraction

The first validation stage is used to detect the 2D corner in the amplitude data. As displayed in the first row of Fig. 5.8(a), this validation stage works well for most of the real corners.

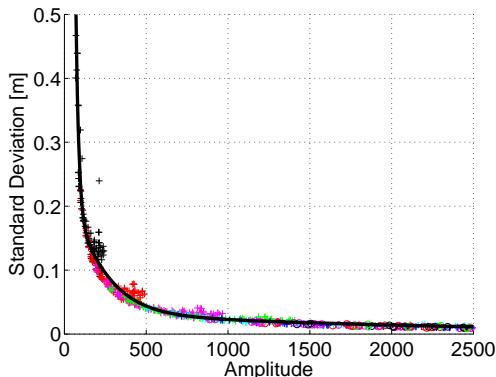


Figure 5.7: The standard deviation of depth data as a function of amplitude data using different integration times. The longer is the integration time, the smaller is the standard deviation of the depth data. This function has been fitted using the Levenberg-Marquardt algorithm.

	Cornerness Measure
planar corner	0.022
good corner	0.030
false corner	0.018

Table 5.1: Result of cornerness measure proposed by Rohr in [85].

However, we typically found occluded features such as, e.g., the last row in Fig. 5.8(a). Using the ToF camera amplitude data, other kinds of bad features can also occur, e.g. caused by rays reflection as depicted in Fig. 5.9. To prune these false features the 3D corner extraction has been proposed.

5.5.3.2 3D Corner Extraction

To improve the feature extraction, we have implemented two similar 3D feature cornerness measures, as described in Sec. 5.4. Both of these measures use the depth gradient information. As compared in Tab. 5.1, we think that the cornerness measure proposed by Rohr in [85] proved to be less depth distinctive.

In this validation stage, only the structure tensor cornerness measures have been used (see Eq. 5.10). As presented in 5.4.1, the visualiation of a good 3D corner is expected to be similar to a sphere like structure. A real good corner is depicted in the second row of Fig. 5.8(c).

The corners in Tab. 5.1 are the same as depicted in Fig. 5.8.

Our criterion for a good 3D corner (see the second row of Fig. 5.8 or Fig. 5.10) is that the differences of the three computed structure tensor eigenvalues have to be lower than a predefined threshold:

$$|\lambda_1 - \lambda_2| < t \quad \text{and} \quad |\lambda_1 - \lambda_3| < t \quad \text{and} \quad |\lambda_2 - \lambda_3| < t, \quad (5.18)$$

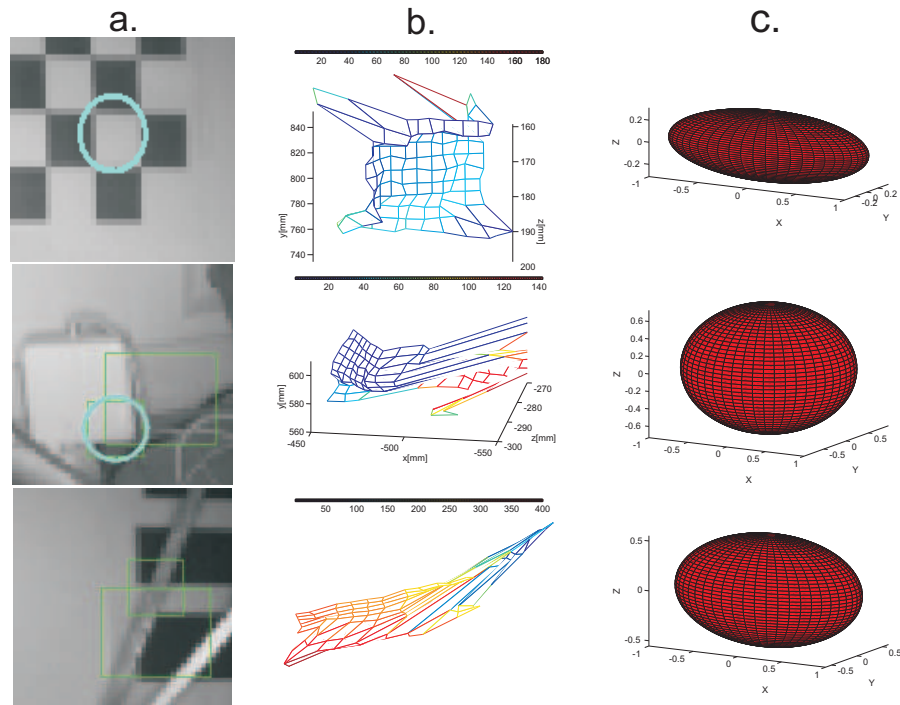


Figure 5.8: Detection of new features using the real image and depth information. The upper row is a planar feature, the middle row depicts a good 3D feature, and the bottom row displays a bad 3D corner. The first column displays the amplitude data provided by the camera. The second column depicts measured depth information w.r.t. the world frame. The third column represents the visualisation of the eigen-decomposition of the 3D feature gradient matrix. The structure tensor visualisation of a good 3D corner is expected to have a sphere like structure, as displayed in the last column of the middle row. The other two features did not pass the 3D cornerness measure.

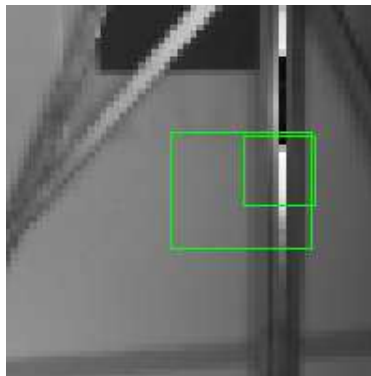
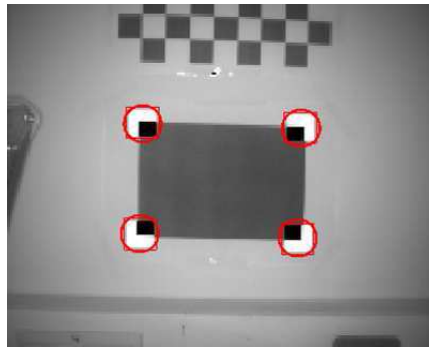
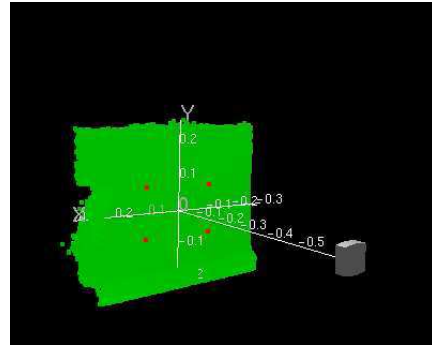


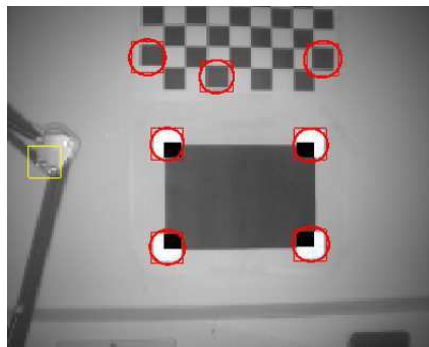
Figure 5.9: An example of a ray reflection on a metallic surface, which is one type of the possible non-systematic ToF camera errors.



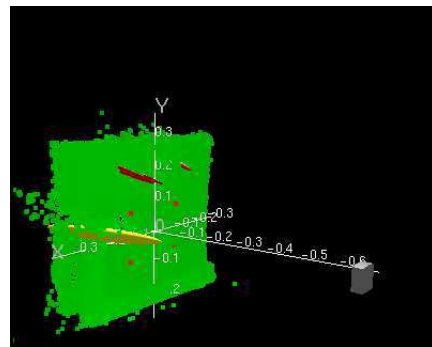
(a) Initialisation.



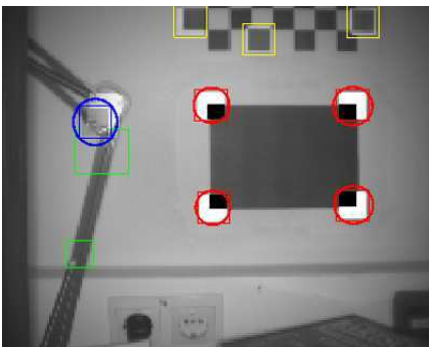
(b) Initialisation+Depth.



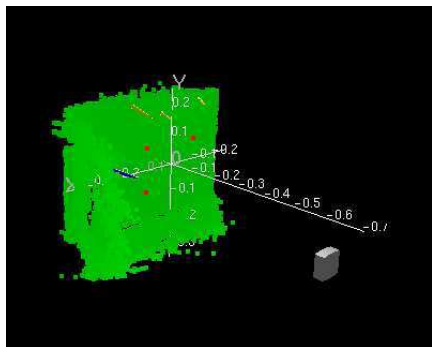
(c) Mapping Features.



(d) Mapping+Depth.



(e) Rejecting Corner.



(f) Rejecting+Depth.

Figure 5.10: An example of rejecting a false corner. The newly detected corner in the last row (the smallest green box on the left side) is rejected.

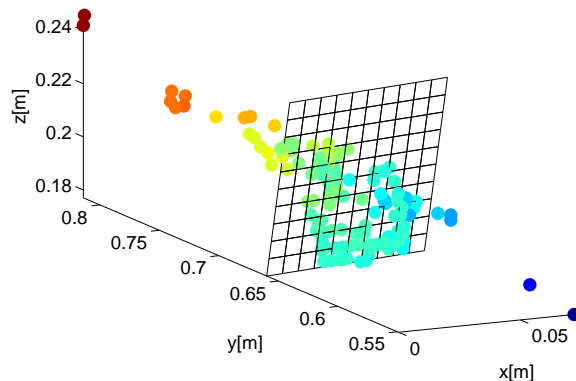


Figure 5.11: Plane fitted to the depth data using RANSAC.

where λ_1 , λ_2 and λ_3 are the eigenvalues of \mathbf{S}_d (ref Eq. 5.10).

Using this criterion, planar features and bad 3D corners are pruned (see the first and third row of Fig. 5.8). These false 3D corners are pruned, because at least one of the eigenvalues is different.

5.5.3.3 Planar Corner Extraction

The third stage is needed when a new real feature is pruned by the 3D corneriness measure but it can lie in a plane. To evaluate this stage, a robust plane fit using RANSAC has been implemented, and an example of the fitted plane is depicted in Fig. 5.11.

Using this RANSAC plane fit, planar features are detected and initialized as depicted in the first row of Fig. 5.8(a). The criterion here is the angle between the normal vector of the plane and the ray of the back-projected 2D feature (ref Sec. 2.2.4.3).

RANSAC's plane fit criterion is the distance of the other points to the potential candidate plane. However, it occurs that RANSAC fails to find the correct plane, and a possibly good planar feature is pruned. This usually happens when the 3D measurements form several optional planes. For example, Fig. 5.12 depicts such a case. We think that a possible solution is to introduce additional geometrical constraints, which would select e.g. fronto-parallel planar corners.

5.6 Conclusion

Inserting false corners into a map of the scene can cause e.g. a mobile robot to get lost [27]. We presented a simulated example how these corners influence the accuracy of the map. From the result it is clear that every false feature introduces additional inaccuracy.

The contribution of this is a three-stage corner selection scheme, which enables the rejection of false features lying at occlusion boundaries or curved surfaces, for example. Our corner selection scheme takes advantage of a recently introduced ToF camera, which

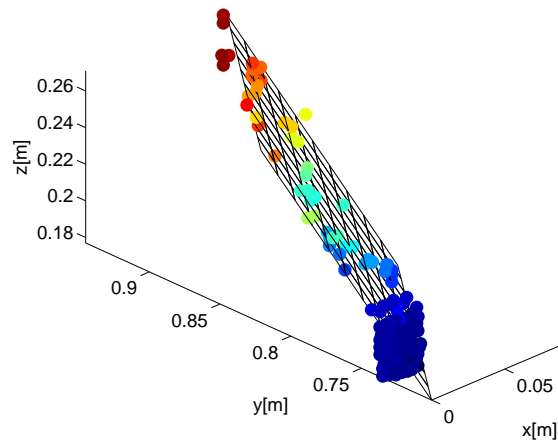


Figure 5.12: RANSAC failed to find the correct plane.

can measure scene amplitude and depth directly. We think that our scheme helps e.g. SLAM approaches to operate in indoor environments more accurately.

Current ToF cameras have several drawbacks (e.g. field of view, noisy data) due to the complex measurement principle, but enhanced sensors have already been announced. It is obvious that this camera can ease the parallel localisation and mapping task significantly, and we think that it is an interesting sensor for indoor robotics navigation and many future applications. The two main disadvantages of the current ToF cameras are: higher cost and lower resolution. However, there is an announced ToF, which addresses these two issues, and should be available soon.

5.6.1 Future Work

The fusion of ToF amplitude and depth information for parallel localisation and mapping has been addressed in the work of Weingarten in [105]. The conclusion is that the ToF camera is less suitable for localisation and mapping than the rotating laser scanner due to its noisy data and limited field of view. However, with the advantage of excluding false corner points this conclusion may now be revised.

In an offline 3D pose estimation and mapping approach introduced by May et al. in [69], the *Iterative Closest Point* algorithm has been used to register point clouds from different perspectives, and KLT or SIFT features have been applied to amplitude data. We think that our proposed corner selection scheme could increase the performance of May et al. approach.

Chapter 6

Conclusion

There are many areas, where robots already help or assist humans. For example, they explore distant spaces or repeat certain tasks accurately. We believe that robotics has the potential to further improve the quality of our lives in the near future. We think that mobile robots especially will play an important role, because they could be applied in a broad range of areas like: office assistance, autonomous transportation vehicles, helping elderly persons, etc.

To present those future applications, the mobile robots need to navigate in unknown indoor or outdoor environments. An important part of the navigation task is the localisation and mapping of an unknown scene. These two subtasks have to be tackled simultaneously, because the success of driving to the right place depends on both of them. The simultaneous nature of these subtasks makes the problem solving very challenging, and it is well-known as SLAM in the robotics community. Nowadays, there exist several well-matured SLAM algorithms suitable for various conditions and sensor setups of which most use laser sensors.

In this thesis, we decided to use visual information as the main sensor input. We think that vision sensors provide the richest information content about an unknown scene. However, the bottleneck of this sensor source is that we had to carefully select the usable content, because the processing time is usually constrained. The reason behind this is that the mobile robot has to make on-line decisions when moving through an unknown environment.

The goal of this thesis was to improve the robustness and the accuracy of present localisation and mapping algorithms. We divided our contribution into three parts.

Our first contribution addresses the problem of localising and mapping an unknown scene using the fusion of vision and inertial sensors. The presented implementation of the motion and structure estimation has two loosely coupled parts. Firstly, we introduced a real-time implementation of the asynchronous non-linear motion estimation filter. This filter uses the fusion of inertial and visual data. Using the combination of these sensor measurements proved to be very efficient for motion estimation. Secondly, we proposed a structure estimation algorithm. This algorithm estimates the unknown scene landmarks using a set of independent non-linear filters. The experiments proved that this concept works for small workspaces. However, our idea with the structure

estimation was missing the inter-feature correlations.

In the second contribution, we substituted the previous motion and estimation framework with SLAM. The goal of this contribution was to enhance the robustness of a monocular SLAM approach using a high-speed camera. We introduced here a high-speed camera to a well-known vision SLAM framework [27], and suggested to use an extended motion model. This motion model presents two additional continuous parameters, which allowed us to follow movements containing higher dynamics. Our experiments with handheld and robotic arm setups validated these expectations. We think that the sensors with a higher readout speed will play an important role for future mobile robot applications. For example, the gaming industry can benefit from potential applications.

In the last contribution, we suggested to select good scene landmarks using a ToF camera. Before presenting our contribution we empirically proved that every false scene landmark decreases the accuracy of the map. These false landmarks are lying at e.g. occlusion boundaries or curved surfaces. To prevent inserting false features into the map of the scene, we proposed a three-stage selection scheme, which takes advantage of using a ToF camera. The ToF camera provides a digital image of the scene together with the depth data. This image and depth information is synchronised and spatially aligned in the camera coordination frame. Our experimental results showed that the selection scheme is capable of rejecting false features. We think that our contribution can help mobile robots to navigate in unknown environments more accurately.

We think that in the future more applications in the area of mobile robotics will emerge. As previously mentioned, using vision sensors has an important advantage, but the algorithms need to process a significant amount of information. Considering, e.g., the high-speed camera and the SLAM framework, the algorithm receives huge amount of incoming data. In this case the algorithm is carefully selecting the useful information, but due to the increasing count of features in the map, the algorithm fails to fulfil the real-time constraint. This is an example, where ongoing research tries to improve the state-of-the-art methods. We believe that this topic has an interesting commercial potential, because many applications require higher dynamics.

The main hardware obstacles in mobile robotics applications are: the computational performance of mobile workstations, the cost of some sensors and the battery capacity. There are many cases when, e.g., a price of a sensor influences the success of a commercial application. In this work, we used an expensive ToF camera, but recently a cheaper alternative has been announced. We think that such a cheap sensor, which can provide amplitude and depth information, would raise immense interest in the robotics community.

Bibliography

- [1] J. Alves, J. Lobo, and J. Dias. Camera-inertial sensor modelling and alignment for visual navigation. In *IEEE International Conference on Advanced Robotics*, pages 1693–1698, Coimbra, Portugal, 2003.
- [2] L. Armesto, S. G. Chroust, M. Vincze, and J. Tornero. Multi-rate fusion with vision and inertial sensors. In *IEEE International Conference on Robotics and Automation*, pages 193–199, 2004.
- [3] L. Armesto, J. Tornero, and M. Vincze. Fast ego-motion estimation with multi-rate fusion of inertial and vision. *International Journal of Robotics Research, Special Issue on Integration of Vision and Inertial Sensors*, 26(6):577–589, June 2007.
- [4] S. Arseneau. *Junction Analysis*. VDM Verlag Dr. Mueller e.K., April 2008.
- [5] A. W. Babister. High-speed flight. *Nature*, 182(4637):695–696, 1958.
- [6] T. Bailey and H. F. Durrant-White. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117, September 2006.
- [7] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley-Interscience, 2001.
- [8] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, June 1995.
- [9] M. Bosse, P. M. 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(12):1113–1139, December 2004.
- [10] M. Bosse, R. Rikoski, J. Leonard, and S. Teller. Vanishing points and 3D lines from omnidirectional video. In *IEEE International Conference on Image Processing*, pages 513–516, September 2002.
- [11] M. Bosse and R. Zlot. Keypoint design and evaluation for place recognition in 2D lidar maps. In *RSS2008: Inside Data Association Workshop*, 2008.

- [12] J. Y. Bouguet. www.vision.caltech.edu/bouguetj/calib_doc/, 2007.
- [13] L. Chai, W. A. Hoff, and T. Vincent. 3-D motion and structure estimation using inertial sensors and computer vision for augmented reality. *Presence*, 11(5):474–492, 2002.
- [14] M. K. Chandraker, Ch. Stock, and A. Pinz. Real-time camera pose in a room. In *3rd International Conference on Computer Vision Systems*, pages 98–110, Graz, Austria, 2003.
- [15] A.B. Chatfield. *Fundamentals of High Accuracy Inertial Navigation*. American Institute of Aeronautics and Astronautics, Inc., September 1997.
- [16] D. Chekhlov, M. Pupilli, W. Mayol, and A. Calway. Robust real-time visual SLAM using scale prediction and exemplar based feature description. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1–7, June 2007.
- [17] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):523–535, April 2002.
- [18] J. C. K. Chou. Quaternion kinematic and dynamic differential equations. *IEEE Transactions on Robotics and Automation*, 8(1):53–64, 1992.
- [19] S. G. Chroust and M. Vincze. Fusion of vision and inertia data for motion and structure estimation. *Journal of Robotic Systems*, 21(2):73–83, 2004.
- [20] J. Civera, J. M. M. Montiel, and A. J. Davison. Interacting multiple model monocular SLAM. In *IEEE International Conference on Robotics and Automation*, April 2008.
- [21] L. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Robotics Science and Systems*, 2007.
- [22] P. Corke. An inertial and visual sensing system for a small autonomous helicopter. *Journal of Robotic Systems*, 2(2):43–51, 2004.
- [23] P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing. *International Journal of Robotics Research, Special Issue on Integration of Vision and Inertial Sensors*, 26(6):519–535, June 2007.
- [24] H. Crabtree. An elementary treatment of the theory of spinning tops and gyroscopic motion. *Nature*, page 182, 1909.
- [25] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE International Conference on Computer Vision*, pages 1403–1410, Nice, France, 2003.

- [26] A. J. Davison. Active search for real-time vision. In *IEEE International Conference on Computer Vision*, volume 1, pages 66–73, October 2005.
- [27] A. J. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.
- [28] D. D. Diel, P. DeBitetto, and S. Teller. Epipolar constraints for vision-aided inertial navigation. In *IEEE Workshop on Motion and Video Computing*, volume 2, pages 221–228, January 2005.
- [29] G. Dissanayake, S. Sukkariéh, E. M. Nebot, and H. F. Durrant-Whyte. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Transactions on Robotics and Automation*, 17(5):731–747, October 2001.
- [30] H. F. Durrant-White and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, June 2006.
- [31] E. Eade and T. Drummond. Scalable monocular SLAM. In *IEEE Computer Vision and Pattern Recognition*, 2006.
- [32] E. Eade and T. Drummond. Monocular SLAM as a graph of coalesced observations. In *IEEE International Conference on Computer Vision*, 2007.
- [33] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *British Machine Vision Conference*, September 2008.
- [34] P. Elinas, R. Sim, and J. J. Little. sigmaSLAM: Stereo vision SLAM using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *IEEE International Conference on Robotics and Automation*, pages 1564–1570, 2006.
- [35] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Robotics Science and Systems*, June 2005.
- [36] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
- [37] E. Foxlin and L. Naimark. Vis-tracker: A wearable vision-inertial self-tracker. In *IEEE Virtual Reality*, pages 199–206, 2003.
- [38] P. Gemeiner, L. Armesto, N. Montés, J. Tornero, M. Vincze, and A. Pinz. Visual tracking can recapture after fast motions with the use of inertial sensor. In *30th Workshop of the Austrian Association for Pattern Recognition*, pages 141–150, Austria, 2006.

- [39] P. Gemeiner, A. J. Davison, and M. Vincze. Improving localization robustness in monocular SLAM using a high-speed camera. In *Robotics Science and Systems*, June 2008.
- [40] P. Gemeiner, P. Einramhof, and M. Vincze. Simultaneous motion and structure estimation by fusion of inertial and vision data. *International Journal of Robotics Research, Special Issue on Integration of Vision and Inertial Sensors*, 26(6):591–605, June 2007.
- [41] P. Gemeiner, P. Jojic, and M. Vincze. Selecting good corners for structure and motion recovery using a Time-of-Flight camera. In *IEEE International Conference on Intelligent Robots and Systems*, October 2009.
- [42] P. Gemeiner and M. Vincze. Motion and structure estimation from vision and inertial sensor data with high speed CMOS camera. In *IEEE International Conference on Robotics and Automation*, pages 1853–1858, Barcelona, Spain, 2005.
- [43] N. Guilbert. *Structure from Motion*. PhD thesis, LTH, Lund, Sweden, 2004.
- [44] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.
- [45] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2003.
- [46] A. Huster, E. W. Frew, and S. M. Rock. Relative position estimation for AUVs by fusing bearing and inertial rate sensor measurements. In *Proceedings of the Oceans 2002 Conference*, 2002.
- [47] A. Huster and S. M. Rock. Relative position estimation for manipulation tasks by fusing vision and inertial measurements. In *Proceedings of the Oceans 2001 Conference*, 2001.
- [48] A. Huster and S. M. Rock. Relative position sensing by fusing monocular vision and inertial rate sensors. In *IEEE International Conference on Advanced Robotics*, Coimbra, Portugal, 2003.
- [49] I. Ishii, Y. Nakabo, and M. Ishikawa. Target tracking algorithm for 1 ms visual feedback system using massively parallel processing. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2309–2314, April 1996.
- [50] P. Jojic. Implementing a time of flight camera interface for visual simultaneous localization and mapping. Master’s thesis, Vienna University of Technology, 2008.
- [51] S. Julier, J. Uhlmann, and H. F. Durrant-White. A new method for the non-linear transformations of means and covariances in filters and estimators. *IEEE Transactions on Automation Control*, 45(3):477–482, 2000.
- [52] S. J. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004.

- [53] T. Kahlmann. *Range Imaging Metrology: Investigation, Calibration and Development*. PhD thesis, ETH Zuerich, 2007.
- [54] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(D):35–45, 1960.
- [55] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776, September 2004.
- [56] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR 2007 Video Results*, 2007.
- [57] T. Komuro and M. Ishikawa. A moment-based 3D object tracking algorithm for high-speed vision. In *IEEE International Conference on Robotics and Automation*, pages 58–63, April 2007.
- [58] J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, 1999.
- [59] D. Kundur and D. Hatzinakos. Blind image deconvolution revisited. *IEEE Signal Processing Magazine*, 13(6):61–63, November 1996.
- [60] J. S. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, September 1998.
- [61] J. Lobo. *Integration of Vision and Inertial Sensing*. PhD thesis, Coimbra, Portugal, 2006.
- [62] J. Lobo and J. Dias. Integration of inertial information with vision towards robot autonomy. In *IEEE International Symposium on Industrial Electronics*, volume 3, pages 825–830, July 1997.
- [63] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1597–1608, 2003.
- [64] J. Lobo and J. Dias. Inertial sensed ego-motion for 3D vision. *Journal of Robotic Systems*, 21(1):3–12, January 2004.
- [65] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [66] Ch. P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.
- [67] Y. Ma, S. Soatto, J. Košecák, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

- [68] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of Applied Mathematics*, 11:431–441, 1963.
- [69] S. May, D. Droschel, D. Holz, C. Wiesen, and S. Fuchs. 3D pose estimation and mapping with time-of-flight cameras. In *IROS 2008: Workshop on 3D-Mapping*, 2008.
- [70] S. May, S. Fuchs, D. Droschel, D. Holz, and A. Nuechter. Robust 3D-mapping with Time-of-Flight cameras. In *IEEE International Conference on Intelligent Robots and Systems*, October 2009.
- [71] P. Mayer, G. Edelmayer, G. J. Gelderblom, M. Vincze, P. Einramhof, M. Nuttin, T. Fuxreiter, and G. Kronreif. MOVEMENT - modular versatile mobility enhancement system. In *IEEE International Conference on Robotics and Automation*, pages 2892–2897, Rome, Italy, 2007.
- [72] J. M. M. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.
- [73] U. Mühlmann, M. Ribo, P. Lang, and A. Pinz. A new high speed CMOS camera for real-time tracking applications. In *IEEE International Conference on Robotics and Automation*, volume 5, pages 5195–5200, New Orleans, USA, 2004.
- [74] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 27, Washington, DC, USA, 2002. IEEE Computer Society.
- [75] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, December 2001.
- [76] N.J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998.
- [77] D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *IEEE International Conference on Computer Vision*, volume 1, pages 199–206, 2003.
- [78] D. Nistér. A minimal solution to the generalised 3-point pose problem. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:560–567, 2004.
- [79] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 652–659, 2004.
- [80] E. Olson, J. Leonard, and S. Teller. Robust range-only beacon localization. In *Proceedings of Autonomous Underwater Vehicles, 2004*, 2004.

- [81] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.
- [82] M. Pupilli and A. Calway. Real-time visual SLAM with resilience to erratic motion. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1244–1249, June 2006.
- [83] H. Rehbinder and B. K. Ghosh. Multi-rate fusion of visual and inertial data. In *International Conference on Multi-Sensor Fusion and Integration for Intelligent Systems*, 2001.
- [84] M. Ribo, H. Ganster, M. Brandner, P. Lang, Ch. Stock, and A. Pinz. Hybrid tracking for outdoor AR applications. *IEEE Computer Graphics and Applications Magazine*, 22(6):54–63, 2002.
- [85] K. Rohr. On 3D differential operators for detecting point landmarks. *Image and Vision Computing*, 15:219–233, July 1997.
- [86] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.
- [87] M. Schuler. Die Störung von Pendel-und Krieselapparaten durch die Beschleunigung des Fahrzeuges. *Physikalische Zeitschrift*, 24(16), 1923.
- [88] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Maching Intelligence*, 28(12):2024–2030, 2006.
- [89] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition*, pages 593–600, June 2004.
- [90] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Massachusetts Insitute of Technology, 2004.
- [91] R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.
- [92] J. Sola, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2499–2504, August 2005.
- [93] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing: Analysis and Machine Vision*. CL-Engineering, September 1998.
- [94] M. St-Pierre and D. Gingras. Comparison between the unscented Kalman filter and the extended Kalman filter for the position estimation module of an integrated navigation information system. In *IEEE Intelligent Vehicles Symposium*, pages 831–835, Parma, Italy, 2004.

- [95] O. Stasse, A. J. Davison, R. Sellaouti, and K. Yokoi. Real-time 3D SLAM for humanoid robot considering pattern generator information. In *IEEE International Conference on Intelligent Robots and Systems*, pages 348–355, October 2006.
- [96] R. Swaminathan and S. K. Nayar. Nonmetric calibration of wide-angle lenses and polycameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10), October 2000.
- [97] S. Thrun, W. Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.
- [98] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and Mahoney P. Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics*, 23(1):661–692, June 2006.
- [99] D. H. Titterton and J. L. Weston. *Strapdown Inertial Navigation Technology*. American Institute of Aeronautics and Astronautics, 2 edition, 2004.
- [100] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis, 1999.
- [101] A. Ude. Filtering in a unit quaternion space for model-based object tracking. *Robotics and Autonomous Systems*, 28(2-3):163–172, 1999.
- [102] T. Viéville and O. D. Faugeras. Computation of inertial information on a robot. In *The fifth international symposium on Robotics research*, pages 57–65, Cambridge, MA, USA, 1989. MIT Press.
- [103] M. Vincze. Dynamics and system performance of visual servoing. *IEEE International Conference on Robotics and Automation*, pages 644–649, 2000.
- [104] Y. Watanabe, T. Komuro, and M. Ishikawa. 955-fps real-time shape measurement of a moving/deforming object using high-speed vision for numerous-point analysis. In *IEEE International Conference on Robotics and Automation*, pages 3192–3197, April 2007.
- [105] J. Weingarten. *Feature-based 3D SLAM*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2006.
- [106] J. Weingarten and R. Siegwart. EKF-based 3D SLAM for structured environment reconstruction. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3834–3839, 2005.
- [107] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *IEEE International Conference on Computer Vision*, 2007.

- [108] B. Williams, P. Smith, and I. Reid. Automatic relocalisation for a single-camera simultaneous localisation and mapping system. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 2784–2790, April 2007.
- [109] XSens. Motion tracker B technical documentation, 2003.
- [110] XSens. www.xsens.com, 2007.
- [111] S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *Proceedings of the Virtual Reality 2001 Conference (VR'01)*, page 71, Washington, DC, USA, 2001. IEEE Computer Society.
- [112] S. You, U. Neumann, and Ronald Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Proceedings of the IEEE Virtual Reality*, page 260, Washington, DC, USA, 1999. IEEE Computer Society.
- [113] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *IEEE International Conference on Computer Vision*, pages 666–673, September 1999.
- [114] T. Zinsser, Ch. Gräßl, and H. Niemann. High-speed feature point tracking. In *Vision Modeling and Visualization (Workshop Vision Modeling and Visualization)*, pages 49–56, 2005.
- [115] G. Zunino and H. Christensen. Simultaneous localization and mapping in domestic environments. In *Multisensor Fusion and Integration for Intelligent Systems*, pages 67–72, 2001.