TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Echtzeitsysteme und Robotik

Efficient and Robust Pose Estimation Based on Inertial and Visual Sensing

Elmar Mair

Vollständiger Abdruck der von der Fakultät der Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Nassir Navab
Prüfer der Dissertation: 1. Univ.-Prof. Dr.-Ing. Darius Burschka
2. Prof. Gregory Donald Hager, Ph.D.
Johns Hopkins University, Baltimore/USA

Die Dissertation wurde am 01.12.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 30.03.2012 angenommen.

Abstract

Reliable motion estimation on resource-limited platforms is important for many applications. While insects solve this problem in an exemplary manner, mobile robots still require a bulky computation and sensor equipment to provide the required robustness. In this thesis, we aim for an efficient and reliable navigation system which is independent of external devices. For that, we assess highly effectual, but still application-independent, biological concepts. Based on these insights, we propose an inertial-visual system as a minimal sensor combination which still allows for efficient and robust navigation.

Thereby, we focus especially on algorithms for image-based motion estimation. Different methods are developed to allow for efficient image processing and pose estimation at high frame rates. Tracking of several hundreds of features and a motion estimation from these correspondences in a few milliseconds on low-power processing units have been achieved. The precision of the motion computation is evaluated in dependence of the aperture angle, the tracking accuracy, and the number of features. In addition, we derive error propagations for image-based pose estimation algorithms. These can be used as accuracy estimate when fusing camera measurements with other sensors. We propose two different ways of combining inertial measurement units and cameras. Either the inertial data is used to support the feature tracking or it is fused with the visual motion estimates in a Kalman filter. For the spatial and temporal registration of the sensors we present also different solutions.

Finally, the presented approaches are evaluated on synthetic and on real data. Furthermore, the algorithms are integrated into several applications, like hand-held 3D scanning, visual environment modeling, and driving as well as flying robots.

Zusammenfassung

Eine zuverlässige Bewegungsschätzung auf Plattformen mit eingeschränkten Ressourcen ist essentiell für viele Anwendungen. Während Insekten diese Aufgabe auf vorbildliche Weise lösen, benötigen mobile Roboter immer noch umfangreiche Sensorik und Rechen-Kapazität, um die erforderliche Robustheit gewährleisten zu können. Auf der Suche nach einer effizienten und zuverlässigen Navigationslösung, beginnt diese Arbeit mit einer Beurteilung von effektiven, aber noch anwendungsunabhängigen, biologischen Konzepten. Basierend auf diesen Einsichten wird die Kombination eines Inertialsensors und einer Kamera als minimales Navigationssystem, das noch angemessene Zuverlässigkeit gewährt, vorgeschlagen.

Dabei liegt der Fokus vor allem auf der bildbasierten Positionsbestimmung. Verschiedene Methoden wurden entwickelt, um eine effiziente Bildverarbeitung und Positionsberechnung bei hohen Bildwiederholungsraten zu ermöglichen. Es wurde erreicht, dass auf einem Low-Power-Prozessor in wenigen Millisekunden mehrere hundert Merkmale verfolgt werden können. Der Einfluss des Öffnungswinkels, die Tracking-Genauigkeit und die Anzahl der Merkmale auf die Positionsbestimmung werden evaluiert. Weiters werden Fehlerabschätzungen für Positionsberechnungs-Methoden hergeleitet, die als Gütemaß für die Fusion von Kameramessungen mit anderen Sensoren Verwendung finden. Zwei verschiedene Kombinationsmöglichkeiten von inertialen Navigationssystemen und Kameras werden vorgestellt. Entweder unterstützt die inertiale Messung das Tracking von Merkmalen oder ein Kalman-Filter fusioniert die inertialen und visuellen Bewegungsschätzungen. Für die zeitliche und räumliche Registrierung der Sensoren werden ebenfalls verschiedene Ansätze vorgeschlagen.

Schließlich findet eine Auswertung der vorgestellten Verfahren mit realen und synthetischen Daten statt. Zudem werden die Algorithmen in verschiedene Applikationen integriert, wie z.B. handgeführtes 3D Scannen, visuelle Umgebungsmodellierung und fahrende sowie fliegende Roboter.

Riassunto

La stima affidabile del moto su piattaforme con risorse limitate è importante per molte applicazioni. Mentre gli insetti risolvono questo problema in modo esemplare, i robot mobili, per garantire la robustezza necessaria, hanno bisogno di svariati sensori e di ingombranti attrezzature per il calcolo. In questa tesi l'obiettivo è di sviluppare un sistema di navigazione efficace ed affidabile e allo stesso tempo indipendente da dispositivi esterni. Il primo approccio è stata la valutazione positiva di concetti biologici adatti a fornire le basi per gli studi seguenti. Grazie a queste intuizioni si è lavorato ad una combinazione di sensori minima, che tuttavia garantisce una navigazione efficiente e robusta, formata da un sensore inerziale ed una videocamera.

Questa dissertazione si concentra specialmente sulla stima della posizione basata sull'analisi delle immagini. Per rendere efficace la loro elaborazione e il calcolo della posizione ad elevata frequenza, sono stati proposti diversi metodi. In questo modo è possibile monitorare diverse centinaia di punti di riferimento con un processore a basso consumo. Inoltre sono stati valutati l'influenza dell'angolo di apertura, l'esattezza del tracking e il numero di punti di riferimento sulla stima della posizione. Si è poi risalito a propagazioni dell'errore di calcolo della posizione, che possono essere usati come indice di qualità per la fusione di misurazioni video con altri sensori. Infine sono state individuate due possibili combinazioni di un'unità di misura inerziale con una videocamera. Queste informazioni inerziali possono sostenere il tracking di punti di riferimento, oppure possono essere combinate in un filtro Kalman con il calcolo del movimento basato su una camera. Per la sincronizzazione spaziale e temporale dei sensori sono state proposte diverse metodiche.

Alla fine i metodi presentati sono stati analizzati con dati reali e sintetici. Gli algoritmi sono stati integrati in diverse applicazioni, come per esempio la scansione manuale 3D, la modellatura visuale dell'ambiente e robot mobili e volanti.

This thesis is dedicated to my parents
for their love, endless support and encouragement.

Acknowledgements

First of all I want to thank Prof. Darius Burschka, who gave me the opportunity to do my PhD in his group. He was a great supervisor, mentor and friend to me. I want to thank him for his time and patience and all the chances he offered me.

Next I want to thank Prof. Gregory D. Hager, who allowed me to visit him and his lab at the *Johns Hopkins University* (JHU) for almost five months. It was a great experience with a lot of insights and fruitful discussions with him and his lab members. Further, I want to thank him for reviewing my thesis.

Special thanks also to Prof. Gerhard Hirzinger and Dr. Michael Suppa of the institute of *Robotics and Mechatronics* at the *German Aerospace Center* (DLR) for the funding and the chance to work in this so inspiring research environment since the early beginning of my PhD.

Thanks also to Dr. Wolfgang Stürzl of the department of neurobiology at the university of Bielefeld and Prof. Jochen Zeil of the *Research School of Biology* (RSB) of the *Australian National University* (ANU) for the chance to visit and talk to the biologists in Canberra. The discussions with them were so inspiring to me. Both greatly supported me with the biological motivation of this thesis and gave me a lot of hints.

Thanks also to the excellence cluster *Cognition for Technical Systems* (CoTeSys) for the funding of the first one and a half years of my work at the TUM.

Further, I want to thank Dr. Gerhard Schrott, Monika Knürr, Amy Bücherl, and Gisela Hibsch for their administrative support at the TUM and for making it to more than a research environment.

I could have never done this work without the support, the exchange of ideas and the discussions with my colleagues and friends at the institute for *Robotics and Embedded Systems* at the TUM, especially the *Machine*

Vision and Perception (MVP) group, and at the institute for *Robotics and Mechatronics* at the DLR, especially the department *Perception and Cognition*. Thank you all for the great time and all the fun we had. At this point I want to mention Oliver Ruepp, my office mate at the TUM, which opened my eyes for some mathematical problems and helped me, *e.g.*, formulating Appendix A.4 in a (as he would say) "less intuitive" way. Further, I want to thank the main project partners for the great collaboration: Werner Meier in the Virtual Environment Modeling project (CoTeSys), Klaus Strobl in the DLR-3D Modeler project, and my office mate at the DLR, Korbinian Schmid, in the DLR multicopter project.

Special thanks belong also to my students Felix Ahlborg, Marcus Augustine, Michael Fleps, Michelle Kruell, Sebastian Riedel, Konstantin Werner, Florian Wilde, and Jonas Zaddach for their great work.

Finally, I want to thank my family Luise, Hans, Sabine and Egon for their support and their patience. I also want to thank Manuela, who gave me strength and energy during the hard-working phases of my PhD, Marco, for being the best housemate I can imagine, and all my other friends which I was neglecting so often but which, nevertheless, were so patient with me.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

Motivation

Since the early beginnings of mobile robotics, researchers work on robots which autonomously move from a starting point A to an end point B. Different questions arise, if one wants to tackle this problem – some major ones are: Which trajectory shall the robot follow to safely reach B? How can the robot be kept on this path? What is the robot's current location relative to B? These three questions are treated in separate research areas. The Motion Planning community tries to answer the first question and, thus, looks for algorithms to compute trajectories under various aspects, such as efficiency, safety, velocity, and so forth. The second question is extensively discussed in control theory. These two questions are beyond the scope of this work, in which we will concentrate on parts of the remaining issue: How can mobile robots efficiently and reliably navigate and localize themselves in 3D space?

The knowledge of its own location with respect to a prior (*localization*) or an aspired position (as part of *navigation*) are crucial for mobile robots[1]. However, in many non-robotic applications similar problems have to be solved, like localization devices for humans, scanning and modeling of objects or the whole environment, surveying, and many more. All these applications need to estimate the absolute or relative position based on one or more sensors. Of course, depending on the applications varying requirements arise.

Up to now, most localization or navigation algorithms with broad impact rely on artificial global references, *e.g.* the Global Positioning System (GPS). Such methods are already available in mass-products, although they suffer from a restricted application space and limited accuracy and are, hence, not feasible for all applications. Man-made global referencing systems will always somehow limit the area of application; even GPS

[1] In general, an approach solving one of these two problems can also solve the other one. Thus, we will not distinguish between them in the following, if not explicitly mentioned.

provides no signal or reduced signal quality for instance in houses, cities, canyons, narrow mountains or forests.

Therefore, on-system solutions are desirable, which do not depend on external devices. Such methods are usually based on the fusion of a large number of sensors to achieve the required reliability, while simultaneously building a map of the world to limit drifting. High processing costs, the price of the sensors and the computational power are the limiting factors of their impact. The commercially available systems are quite expensive or do not have adequate reliability or accuracy. Most approaches provided in literature tackle only the localization problem for a certain application, but do not provide a generic solution. Moreover, due to high memory consumption, these methods do not scale well in general and, thus, restrict the workspace. A local navigation system, which allows for a reliable pose estimation in general environments while providing a scalable map for drift compensation, is of great interest.

Various aspects have to be considered for the decision on which sensors to use. Many applications, like the localization of hand-held devices, exhibit high dynamic motions. Furthermore, increasing dynamics are also noticeable in mobile robots, which generally become more and more agile, solving their tasks more quickly. To deal with such motions, sensors have to be used that provide short integration times and high sampling rates. Another significant characteristic of sensors is whether they are passively capturing the environment or actively emitting signals, *e.g.*, light or sound to perform their measurements. The passive sensors have no direct influence on their measurements and, thus, they completely depend on the scene characteristics, like lighting conditions or object properties. This makes it difficult to develop generic algorithms with universal parameters. However, beside this drawback the passive nature of sensors allows in general for more diverse environments and, thus, for a wider field of application than active sensors. These usually have a rather high power consumption, provide only a limited measurement range, and they suffer from cross-talking as well as easy jamming and detection.

There is a continuous trend to miniaturize platforms, not only to reduce production costs, but also to allow for micro or light-weight systems. To achieve that, compact sensors should be used, their number should be reduced and the algorithms should be optimized for memory and computational efficiency. However, since every sensor has its drawbacks, one should not rely on a single one, but fuse the data acquired by different sensors to yield reliability and robustness. Every additional sensor improves the estimation result, but for the sake of the system's compactness, only as many of them as necessary should be used. A trade-off has to be found between reliability and efficiency.

A short excursion into biology should provide some insights into which senses nature found to be useful for navigation and how this task is solved by insects. The sensor-concept and the methods which we will present in this work are inspired by these biological assessments.

1.1 Efficient Navigation Inspired by Biological Insights

To motivate our idea of an efficient navigation system, we will first try to analyze navigation from a biological point of view. Any life-form has been optimized by the process of evolution over many millions of years. This selection process led to the most efficient and robust navigation systems. Sometimes they have been adapted to a specific ecological niche and sometimes they have been generalized to allow for a large environmental variety. The tasks of most animals may be always specific and rather simple, *e.g.* foraging, mating or moving back home, and, hence, they do not need generic navigation strategies. However, the same species of even rather primitive animals, like insects, can be found in a large variety of environments. Insects are quite resource-limited animals if compared to humans or other mammals - they require a highly efficient but still robust navigation apparatus. Therefore, we let us inspire by their sensory organs and their behavior when trying to answer the questions: *"Which sensors were found by evolution to be necessary for robust navigation?"* and *"How can insects with their small brain reliably find back home after moving thousands of body lengths to a foraging site?"*.

Even though much research has been devoted in biology to answer these questions, the experiments are not always conclusive and, hence, many aspects are still part of current discussion or only theories without proper experimental grounding are available. Nevertheless, we will try to assess various ideas and concentrate on the common assumptions by denoting meaningful experiments. For that we are not looking for any family or even species specific behaviors or physiological details, but we try to extract common concepts. Indeed, the physiological properties relevant for navigation seem to be in general quite similar, even between different families of insects. Especially their long-distance navigation techniques seem to be comparatively coherent. Probably the best studied insects, in terms of long-distance navigation, are bees and ants. Although they act in different spaces, they seem to use common navigation techniques, which should, hence, allow for an environment- and task-independent generalization. Therefore, we will focus on the extensive experiments and theories on ants and bees to find an answer to the aforementioned questions.

The insights in this section should motivate a compact but reliable sensor setup and efficient navigation algorithms. However, an extensive survey and discussion is beyond the scope of this work. Comprehensive overviews of specific aspects, *e.g.* the sensory system, navigation strategies or insect vision, can be found amongst others in [15, 16, 17]. In the following we will first assess the insects' physiology and their navigation behavior to finally make the transition to mobile robotics by mapping the senses and the strategies to technical sensors and navigation algorithms.

1.1.1 Assessing the Sensory Physiology of Insects

The difference in how sophisticated a sensory organ is depends on the importance of the information it provides in a specific environment. Beside that, the physiology between species is quite similar. Ants, *e.g.*, live in forests, deserts, houses, on rocks, basically limited only by the presence of enough food. Of course, different species of ants vary slightly in their physiology and do, therefore, fit more or less well in a specific environment. Many of them use pheromones to mark their path. By varying the intensity or the pattern, recruiting individuals may signal whether the scent trail is promising for foraging or not. However, such a feature would not help desert ants, like the *Cataglyphis bicolor*, because any pheromone would vaporize immediately in the desert's heat [18].

To get an idea of which senses insects may use to solve their navigation task we will start with a list of senses and their possible use for navigation.

- **Orientation sensing**: Animals can estimate the global direction based on their sensitivity for the Earth's magnetic field (*magnetic compass*) [19, 20]. A compass is crucial to ensure direction of heading, accurate path integration and for the alignment of visual representations of the environment. Turtles, lobsters, newts, and birds, for instance, even use a geomagnetic map for navigation [21, 22].
  Insects use also their visual system for orientation estimation (*visual compass*). They use terrestrial information like, *e.g.*, the landmark panorama [23], far distant beacons or celestial information like the polarized skylight [24], the direction of the sun or the moon, the spectral or intensity distribution of skylight and the sky itself (*i.e.* clouds). One dedicated organ seems to be the ocelli [25, 26] and the *dorsal rim area* (DRA) - a specialized dorsal part of the compound eye, which contains receptors that are sensitive to the plane of polarization of light [27].

- **Vision**: The two compound eyes of insects consist of thousands of facets, the so called ommatidia, which are spread almost over the whole sphere. Each ommatidium consists of several individual photoreceptor units. House flies (*Musca*

domestica), for instance, have about 8,000 ommatidia, worker honeybees (*Apis Mellifera*), *e.g.*, have about 10,000 and dragonflies (*Anisoptera*) up to 60,000 ommatidia. For comparison, a camera with a resolution of 320×240 has already 76,800 pixels - more than most insects. The interommatidial angle in honeybees varies from 1.4 to 4.5 and the acceptance angle of the ommatidias has been measured to be about 2.6° [28, 29, 30]. The blind area, covered by the torso of the bee, is approximately an ellipse of about 47° azimuth and 100° elevation. The field of view (FOV) of the two eyes does also overlap by about 30° [29].
Several insects possess a multi-chromatic opponent color processing system (multi-spectral pixels). Honeybees, *e.g.*, have a trichromatic perception for blue, green and ultraviolet. The latter is important for some specific tasks, as, for instance, to perceive patterns on nectar-laden flowers that are invisible to us. However, there might also some navigational aspects be involved, like the fact that in the ultraviolet channel the sky is always bright and far clouds are invisible [31] or the increased contrast for the skyline panorama [32].
The neuronal system provides fast, analog processing of the visual information. In insects, each ommatidium is serviced by ca. 12 neurons in the first layer of processing (*Lamina*) and by up to 60 neurons in the second layer of processing (*Medulla*). There are extensive lateral interactions at many different scales, especially in the medulla. Flies showed in experiments that they can reliably respond to visual changes up to $200\,Hz$ [33]. Flying insects cannot make use of any leg-based odometry. For them visual odometry is crucial for path integration. Visual odometry uses the optical flow[1] to estimate the distance traveled [34].

- **Proprioception**: It is quite likely, that insects are, like humans, aware of the position and motion of their limbs – information that can be gathered from mechanoreceptors in joints and muscles. An important concept of proprioception is leg-based odometry, which is used by various walking insects and is probably their most significant sense for navigation. By counting their steps and by fusing the compass information they could yield accurate and reliable homing vectors even on rough terrain, where the step sizes vary.

- **Tactile Sensing**: The tactile recognition of landmarks, tactile obstacle detection and wall following is crucial for animals with antennae, vibrissæ or other sensible hair that live in dark, confined environments, like the crayfish [35]. However, this sense is not really relevant for long-distance navigation.

[1] Optical flow represents how fast the world moves in the visual system - this concept will be described in Section 2.4 in more detail.

- **Inertial sensing**: Several insects, *e.g.* flies, have modified hind wings, so called *halteres*, which are flapped rapidly and measure the angular velocity (like gyroscopes) based on the Coriolis forces [36]. This information allows animals to keep their head and, thus, their sensors stabilized in all three rotational axes.
  Some insects have a dedicated organ to measure accelerations, the so called *statocysts*. It is a heavy body lying on a bed of mechanoreceptors that are sensitive to shearing forces. Additionally, the force raised by the muscles reveals also applied forces. There is also evidence that the relative angle between thorax and abdomen is used to measure the gravity.
  Further, there is a very widely distributed reflex, the so called *dorsal light reflex*, that is used by many animals to define the vertical direction [37]. It uses the overall light distribution to control the roll orientation of the head.

- **Odor**: Ants and also flying stingless bees use pheromones to mark trails, their nest and food sources. Many other animals mark their territorial boundaries with pheromones. The navigational problem that arises once an odor plume has been detected is to draw conclusions of where the odor comes from, taking into consideration that it has been perturbed by wind and other disturbances [38]. Hence, odor is mainly used to detect artificial markers, *e.g.* pheromones, and as communication channel between individuals, *e.g.* to reveal the mating-state. However, there is some experimental evidence that odors form stable patterns at landscape scale, which are used by pigeons (doves) for navigation [39, 40, 41].

- **Taste**: Similar to odor, the taste sense can be used to detect artificial markers, like pheromones, or for other task-specific communication. One major difference between taste and odor is that a marker-detection can only occur at the actual marker location. This prevents far distant perception, but avoids also the localization problem of odor.

- **Hearing**: Sound is measured on the insect's surface by the so called *tympanal organ* [42]. Special vibration-sensible legs have also shown in experiments to be responsible for sound perception, *e.g.* in scorpions and spiders [43]. Further experiments proof that sound or ultrasound is used as active sense for mate attraction or predator avoidance in flying crickets [44] and bats, *e.g.*, are also known to use acoustic landmarks for navigation [45].

- **Wind sensing**: Flying insects can measure the air speed with their antennae and with filigree hairs on their head. There is no experimental evidence that this

information is fused together with the visual odometry and orientation sensing for path integration, but it would probably help to make use of this sense.

- **Range sensing**: It is not clear to which extend insects use their stereo system for range sensing. The short baseline and the low resolution make stereo triangulation only possible within a few centimeters. Hence, it seems likely that insects do not rely much on stereo triangulation. Only in case of close objects stereo vision would be applicable. Such a binocular spatial localization has been experimentally shown for the praying mantis while catching the pray [46]. However, while insects do not provide a dedicated range sensing organ, other animals, *e.g.* bats, do. The ultrasound for bats is like the tactile sensing for crawling insects - it reveals the geometry of the environment and is used for navigation and obstacle detection.

1.1.2 Biological Insights in the Navigation Behavior of Insects

In this section, we will try to outline how insects solve the navigation task with their senses and their limited processing and memory resources. We will focus on navigation strategies which do not rely on any communication and, hence, we neglect *e.g.* the pheromone trail of ants and the famous "waggle" dance of bees, which is how bees communicate the location of foraging sites in the hive. These aspects are of major interest for research on swarm intelligence and are beyond the scope of this work. The non-interactive navigation strategies rely mainly on path integration by any kind of odometry (proprioceptive navigation) and navigation by environmental perception (exteroceptive navigation).

The following senses can be used to perceive cues in the environment for navigation: vision, odor, taste, hearing and range sensing. We will discuss the importance of each sensor in Section 1.1.3 and conclude that visual cues are sufficient to solve all required tasks for navigation. Several species of ants and bees seem to successfully navigate over long distances only based on their visual system and relying on path integration. Hence, in the following we will only discuss visual navigation concepts.

Path Integration

Path integration is the continuous integration of movements over an insect's journey. Cues that can be used for this process are, for instance, angular velocity, optical flow, compass information (*e.g.* the gravity vector) and any proprioception, like counting the number of steps. The fusion of all this information allows for accurate dead reckoning.

The integration of the path is in general very robust, because it uses mostly motion cues which do not directly rely on the environment and, hence, are not affected by

any external disturbances. Therefore, it is also the major fall-back solution in case the navigation based on environmental cues fails. The result of path integration is a global vector pointing to the starting point, *e.g.* the nest, or, if the path has already been traveled, an arbitrary point of interest, *e.g.* a foraging site. This vector denotes a straight and, hence, the fastest way to the goal. A long exploration path can be abbreviated on the way back just by following this vector. However, some other experiments show, that insects use also local path integration vectors to connect landmarks.

Fiddler crabs (*Uca vomeris*), *e.g.*, reveal the importance of path integration when fighting. They have only a small action space, but they need to find a hole in a flat world from a viewpoint close to the ground. Their path integration is very accurate and the only way to navigate. If such crabs are fighting, they use their large pincer not to hurt each other, but to lift each other from the ground. At the point they loose touch, their path integration fails and they are not able to find and hide in their holes anymore, which exposes them to predators [47, 48].

Visual Exteroceptive Navigation

However, for insects which forage in larger environments, path integration is not accurate enough for long distance explorations. Any small error becomes constantly accumulated and, hence, the divergence of the proper direction and distance at the end of the journey increases with the traveled distance [49]. This is a systematic problem of any dead reckoning based method and cannot be avoided.

The visual system allows insects to follow known paths and to localize within a familiar environment. In case visual navigation fails, insects seem to rely again on path integration. Once they are at the end of the vector and their visual navigation system is still lost, they initiate a species specific search pattern to look for the goal they were heading to, *e.g.* the nest [50]. If an insect is passively displaced by, *e.g.*, a gust of wind or an animal, the computed path integration will miss the goal by the magnitude of the displacement [51].

While path integration is rather well understood and verified, the various visual navigation concepts in insects still pose many open questions. The ocelli are also a visual sense, but their is no evidence that this organ is used for anything else than attitude control (sky compass) and, thus, they are not further discussed at this point. The compound eye is, therefore, the only organ providing detailed visual navigation information. Let us keep in mind, that the view of a bee is completely different than what humans see. As you can see in Fig. 1.1[1], the resolution is less but the field of view

[1] www.zf-laser.com

(a) Omnidirectional view (b) Bee-eye view

Figure 1.1: The left image shows an unwarped omnidirectional view rendered from the data acquired by a Z&F laser-scanner. The right image shows the same view as seen through the eye of a bee. (Both images are by courtesy of Wolfgang Stürzl [30])

covers almost the whole sphere. It is difficult to realize an accurate body stabilization based on such a low resolution. Other organs, *e.g.* the halteres, are more likely to be used for this task. However, optical flow is clearly used to prevent drifting and to keep a certain height [52]. By assuming that the velocity of the bee is constant, it can simply stay at a certain altitude by keeping the optical flow constant. Another effect, which one can note observing insects, is, that if an insect is flying against too strong wind, it will force it to land. Moreover, if insects fly over a clear lake, they may drown, because they try to keep the optical flow of the ground at a certain level. There is also some work in literature showing that by some assumptions it would be possible to estimate the egomotion based on optical flow and by measuring the air speed [53].

For visual navigation there is no need for high accuracy, but it is crucial to provide highly robust and reliable localization. As we will show and discuss in Section 8.2.1 the resolution is the major factor which determines the accuracy and a field of view which is much larger than 120° worsens the precision for a constant resolution, but it helps to gain robustness and reliability. Many questions remain, which visual information is gathered and how it is processed, memorized, and used by the insects brain while navigating through the world.

In the following, we will discuss experiments which will give some insights into the exteroceptive navigation of insects. For that we will formulate three questions and try to find the answers. The statements we will make are mainly based on experiments on bees and ants, because an extensive survey of the problem would go beyond the scope of this work. However, for the sake of clarity we will make general statements without delimiting the validity of every conclusion to the respective species.

Do insects use the whole field of view or only single landmarks for pose estimation?
Humans have an optical system which provides high resolution for a small aperture angle (the *fovea*), improving the recognition of focused objects they are looking for. In contrast, the resolution in the optical system of insects does not vary as much over the whole FOV and is in general much lower. However, a large field of view allows to robustly identify the current location, but does it provide the high accuracy necessary to find specific locations, like the nest entrance or a known feeding source? Experiments have shown, that it is possible to approach a goal based on the gradient of image differences, once a set of images of the trajectory to follow are known [54, 55, 56]. For that each view is substracted from a so called *snapshot*, an image which has been acquired while learning the trail or the nest environment, and by following the resulting gradient the insect reaches the position where the snapshot has been acquired. From there on the next snapshot can be approached until it finds the goal. Such global methods which use similarities between images are easy to realize as a neuronal network in a bee's brain. Further, they would also explain the learning flights of bees, where the insect backs away from the goal in a series of arcs that are roughly centered on the goal [57]. However, it is still not clear according to which criteria they decide when to acquire a new snapshot. A trade-off has to be found between enough similarity between the images to allow for an overlapping catchment area of the gradients and enough differences to be memory efficient.

While the latter conclusions would suggest that insects use their panoramic view to find a location, there are many other experiments showing that visual navigation of insects is strongly related to specific landmarks in the environment [58, 59]. Thinking of an omnidirectional view, there are many possible landmarks and the insect has to determine how much it relies on each feature. Thereby, it would make sense to weight the cues in the image according to their distance and according to how reliable they are. The visual localization accuracy in space is directly related to the distance of the objects in the field of view. An insect can measure the distance to a landmark by speed-based parallax or motion-based parallax [60]. According to [61] landmarks should be weighted based on the following three characteristics: *salience* - landmarks should be unique and easy to distinguish from other part; *permanence or reliability* - landmarks and their position should be constant over time; *relevance* - a landmark should help to recognize important places or decision points. There is experimental evidence which shows that not all landmarks have the same influence on navigation, but a sort of weighting of the information gathered from landmarks occurs [62, 63].

The question remains, whether single landmarks are recognized as objects or whether the location of the corresponding cues is only encoded in the retina (so called *retinotropic*

encoding). It has been empirically shown, *e.g.* with water striders [64] and ants [65], that insects follow a path by keeping a certain edge or point on a specific retinal position. Nevertheless, they are also able to relate landmarks to each other and categorize them as we will see in the next question.

Summarizing the results we retrieve following insights: Insects use the panoramic view to find a location, but it seems that there is also a weighting of the image areas involved. Strong landmarks are predominant and have more impact on the insect's navigation than the "background curtain". The panoramic, far distant background provides a visual compass, but its drawback is that it does not reveal any translational information. Hence, close landmarks or texture is crucial to correct for translational drifts. Honeybees, *e.g.*, almost perfectly stabilize their head while flying, which allows them to fly strict translational and change directions by rotation saccades [66, 67, 68]. Thus, they behaviorally separate rotation from translation, which eases and robustifies vision-based motion computation significantly as we will see in Section 5.1. Further, to keep a specific position they seem to use also so called *visual servoing* techniques, where they keep image cues on predefined retinal locations.

Which cues do insects use as landmarks and how are correspondences between them established?

Insects have been shown to rely on various visual cues, like color [69, 70, 71], apparent size, edge orientation [72] and the vertical center of gravity of a shape [73], but also their spatial arrangement [74]. Usually, all these cues indicate the same origin or direction, but what happens if different cues provide contradictory information? Insects have been shown to not only average or interpolate between cues, but they are also able to generalize and categorize them [75]. They can, *e.g.*, discriminate between different oriented stripes, but at the same time they also generalize a pattern and "identify" it as stripes independent of its orientation.

Experiments show that they do not only navigate based on the resulting probability of the most likely location, but they also take into consideration the likeliness of whether a location or direction is reliable. In case the information from the visual system is interpreted to not yield enough probable navigation information and, hence, the insect is lost, it starts with a search pattern.

Even though apparently the different visual stimuli are fused, there is no proof for a probabilistic fusion of landmark guidance and path integration. The latter serves as a fall-back strategy and scaffold for learning – there is no evidence that ants use landmarks to reset their current global path integration coordinates when they are displaced to a location with familiar visual stimuli [76, 77, 78]. Landmarks are primed by path

integration and path integration is suppressed in landmark rich environments [79]. Due to this lack of a close fusion between path integration and visual landmarks, we can assume that on the one hand landmarks are not used to improve the accuracy of path integration, while on the other hand global path integration is not used to provide landmarks with positional coordinates. There is no evidence that insects build geometric maps of their environment. Nevertheless, they use landmarks to trigger certain local path integration vectors with defined distance and direction, which were stored as a specific part of the route [60].

Summarizing, we can say that the tasks of path integration and navigation are separated. Thus, insects do not combine localization and (geometrical) map building, like it is suggested and implemented by the popular SLAM (*simultaneous localization and mapping*) methods in robotics. Rather topological maps are used to solve this task, whereas a large part of the biologists is of the opinion that insects do not rely on maps at all. Further, many cues of different types are used in a weighted way to allow for a robust navigation. As soon as an insect is lost, it relies on fall-back algorithms.

How are paths or routes stored in the insect's memory?

Landmarks are route specific – insects may use different routes when they go to or come from a certain location even though the omnidirectional view would allow to use the same landmarks [80]. They seem to be not able to inverse a path if it is only known in one direction. An ant, *e.g.*, which is on the way to the foraging site and becomes displaced to a different path which it knows only from going back to the nest, will follow this path to the nest instead of taking the inverse trajectory to the foraging location where it actually was heading to [81]. Due to the panoramic view, the ant would have also seen the landmarks from behind, but apparently it cannot use them in that constellation. Humans seem to build a so called cognitive map which allows to navigate based on the spatial relations of prominent features. Experiments led to the conclusion that bees use their path integration system, piloting system and image matching system in a non-cooperative and competing manner [82]. There is no indication for a level of combinatorial integration of the representations which would force the assumption of the existence of cognitive maps in insects. They rather follow specific routes which they set up during exploration or by following a scout to the foraging location.

However, the location of landmarks influences how insects choose their trajectory [58] and, hence, it is quite probable that individuals of the same species heading to the same location will follow the same trajectory, even if it is not a straight line. This behavior seems to be a trade-off found by nature between memory efficiency and traveling efficiency. Much less abstraction capability is required and less information about

the environment needs to be stored when an insect simply moves from one landmark to the next instead of taking unknown shortcuts.

Further, insects spend a variable amount of memory to recognize locations or paths depending on how often they need it [60]. It seems that they refine their path every time, evaluating the reliability and, thus, the importance of landmarks for navigation. Of course it is more important to find again the nest than a foraging site and, thus, any insect which lives in a hive or a nest spends some time to memorize the close surrounding of the nest before leaving it.

Concluding we have seen that insects have not a complex cognitive map which they can rely on, but they have rather a set of paths which they can follow based on landmarks. The location of the landmark also influences the trajectory, which leads to the assumption that insects store routes as lists of landmarks or views. Landmarks can be related to a certain path, which allows them to set up on a known route if lost, but only in the direction they have stored the path. Following specific, landmark-related exploration strategies reduces the amount of information to be memorized.

1.1.3 Biological Inspired Efficient Navigation System

The selection process in biological evolution guarantees that all the mechanisms we can find nowadays in nature are superior - within the current constraints - to any previous biological solution. However, it should not be our aim to copy the biological systems, because we do not have the same application constraints or the same realization requisites in nowadays technology and we can not provide the same embodiment with our robots. Further, animals have in general other constraints than robotic applications. We should rather try to be inspired by the computational mechanisms which we can observe in nature and translate them to our robots and applications.

Sensory Equipment

Let us start with the question which current technologies could realize the insect's senses and which of them are necessary for a robust but minimal system. In the following we will list the corresponding sensors to the senses in insects and assess them:

- **Orientation sensing**: It is possible to gain relative and absolute compass information by using cameras with or without polarization filter, but it is currently much easier to provide a magnetic compass for a global orientation estimation in robots. However, such compasses have difficulties in the vicinity of magnetic materials like, *e.g.*, the iron in buildings. Furthermore, one can use the gravity

vector measured by accelerometers to determine two degrees of attitude. This method fails as soon as a significant translational acceleration is applied.

- **Vision**: Cameras become more and more popular in robotics due to their low power consumption, compactness and low prices and due to the fact that they are noninvasive and accurate. However, only a few approaches are known, which try to achieve visual systems with similar spatial and temporal resolution characteristics as the compound eye of insects [83, 84]. Most of the technological realizations of optical flow devices are one dimensional arrays of photoreceptors [83]. There are first technological solutions which allow for a parallel processing of two dimensional fields. Up to now, however, they only exist in sizes up to 30×30 [85]. Another approach is to use a regular camera and a field-programmable gate array (FPGA) to realize parallel dense optical flow computation [86, 87, 88, 89] or a graphic processing unit (GPU). FPGA based solutions require long development times, while GPUs suffer from high power-consumption and heat generation. To allow the implementation on resource-limited embedded systems, one should for the time being focus on the development of sequential algorithms. Further, a common drawback of both processing units is that it is hard to realize complex, adaptive algorithms and, hence, they usually only compute the optical flow from image to image based on computationally simple algorithms, *e.g.* [86], and feed the control loop with the outcome [90]. However, this does not prevent drifting, which is actually a main advantage of optical sensors. The stabilization can also be provided by the inertial sensors, but motion estimation based on them inevitably drifts. Thus, one should not focus on visual localization for inner-loop control, which requires high sampling rates and accuracies, but for drift-prevention.
  The catadioptric imaging system presented in [30] provides a 280° field of view, while the panoramic mirror in [91] has a smaller aperture angle but a uniform angular resolution in elevation. In general, cameras on robots are equipped with narrow field of view lenses, trying to reduce the accuracy limits introduced by the pixel discretization. This contradicts the observations of insect eyes. Humans overcome the drawback of a limited field of view at high resolution by extensive head and eye movements and by building an environmental model in their mind, which allows them to navigate, despite seeing only a small part of the world. This is computationally expensive and beyond a minimalist navigation system as insects are. There are several algorithms that were designed for spherical views and, thus, are also more accurate and reliable for large aperture angles [92, 93].

In Section 8.2.1, we will see that the accuracy for a generic algorithm on projective images has an optimum for the focal length and decreases again for too wide angles, assuming a constant resolution. Anyway, a large FOV increases robustness and reliability of motion estimation and especially for global localization. Further, it reduces the aperture problem, improves obstacle and object detection, and allows for the realization of a visual compass as illustrated in Fig. 1.2.

In Section 1.1.1 we mentioned that the insect's eyes process images at a high

Figure 1.2: This sequence shows images acquired by a catadioptric camera mounted on a quadrocopter. It would be straightforward to extract the horizon (note the slight irregularities and texture differences) which could then be used to estimate the full rotation and not only the yaw angle (visual compass).

rate. How relevant is this for visual navigation? If image cues move several pixels between processing instances, it is computationally much more expensive than if only neighboring pixels have to be evaluated for data association. A neuronal network requires less interactions of neighboring neurons and becomes more robust if the optical flow vectors are short. In the machine learning community, *e.g.*, it is well-known that the constitution of a neuronal net prefers a small area of interaction of neighboring neurons than a highly cross-linked one (broad vs. deep neuronal net). The distance of visual cues on the retina or the image plane depends on the velocity of motion and the processing rate. Hence, a fast visual processing makes the computation of correspondences easier and reduces the number of possible matches. Otherwise, assumptions on the motion or the environment have to be made in order to disambiguate within a large set of possible matches.

Hence, for a robust, efficient and reliable motion estimation a camera with a large FOV and high processing rates should be preferred.

- **Proprioception**: The current joint position or the link size of a robot are in general well known. Lately, also impedance-based control became possible due to respective force sensors in the joints [94, 95]. Such robotic concepts are especially interesting for, *e.g.*, safety issues [96]. However, counting the revolutions per

minutes of a wheel does usually not allow for accurate odometry estmiation due to slippage and changing ground conditions.

- **Tactile Sensing**: Tactile sensors are of great interest in robotics [97, 98]. However, they are only of little importance for unconstrained, long distance navigation.

- **Intertial sensing**: Inertial sensors measure the derivatives of motion, which is crucial for high dynamic control. Inertial measurement units (IMUs) consists of a gyroscope and an accelerometer for each of the three spatial axes. Their information can also be used for strapdown calculations (see Section 2.6) to improve path integration.

- **Odor**: Research on the development of odor sensors started in the 1990*s* and became more popular with various promising approaches in the last years [99]. The navigation relevant aspect of odor is quite application specific and not interesting for a general solution. There exists, *e.g.*, some work on autonomous plume tracking robots, which should replace lobsters to sniff out mines [100]. Odor as communicative channel can be easily replaced by any other communication technology, *e.g.* radio.

- **Taste**: The technological realizations of the taste sense are far too complex and time consuming to be used in mobile robotics. Further, taste as navigation strategy would only be relevant, if it is feasible to set artificial markers. Hence, this sense can be neglected for our considerations.

- **Hearing**: Acoustic signals were first used in underwater vehicles for navigation issues [101], but nowadays they are also used on mobile robot platforms to localize relative to sound sources [102]. However, the accuracy which can be achieved is not as good as for visual or range sensing devices and, thus, it is in general only used to drive the attention of the robots rather than localizing them.

- **Wind sensing**: For flying systems it might be of interest to measure the air speed and, thus, their velocity against wind. Such *anemometers* are available as small assemblies. However, depending on the aircraft it might be difficult to measure the wind – the spinning rotors of a helicopter, *e.g.*, make it difficult to measure it.

- **Range sensing**: Range finders are quite common in robotics. Light or laser detection and ranging (LIDAR/LADAR), time-of-flight (TOF) cameras, sonar,

stereo camera systems, are a few sensors which measure the distance relative to their environment. While most of these devices are of large size or need a complex setup, sonar is quite small and lightweight. Although being not quite accurate, its measurements can be used for reliable obstacle detection. Unlike infrared (IR) diodes, sonar can also detect glass and it is not influenced by the presence of sunlight. Range measurements can be used in a straightforward manner to build geometric maps, which are easy to understand and to navigate on. However, most of them are active sensors and they all suffer from a limited range of measurement.

Three crucial tasks of a mobile robot system in a general environment can be identified to be application and system independent. They are: *motion estimation* to control the robot, *navigation* in its environment and *obstacle detection* and avoidance. In the next three paragraphs we will evaluate the relevance of the mentioned sensors for these tasks.

Low-level motion estimation

IMUs are the most important sensors for high dynamic control in mobile robots. The compactness, low power consumption and high sample rates of these sensor units make them superior to any other available sensor on the market for high dynamic control. The major drawback of this sensor is that it only measures the motion derivatives as angular velocity and translational acceleration which makes it prone to drifts. The roll and pitch angle can be stabilized using the gravity vector measured by the accelerometers. Wind sensing is, except for special tasks, not useful for ground robots and is, therefore, disregarded by our considerations. Range sensors measure the distance to objects within a certain range. If these objects are static, a mobile system can estimate its motion based on these measurements. A monocular camera allows to estimate the rotation and the translation up to scale. The scale of the translation depends on the distance of the visible landmarks. The crucial requirement is texture - if there is no visible structure an image-based motion estimation becomes impossible. A camera is actually an instrument which measures angles (*goniometer*). Due to the projection of the 3D world on a 2D image surface, one dimension gets lost and can only be partly recovered by using several measurements. This loss of one dimension and the large amount of data makes visual motion estimation algorithms in general complex, error-prone and slow. Orientation sensors, like a magnetic compass or visual orientation estimation techniques, can further improve the robustness of motion estimation. A magnetic compass, *e.g.*, can estimate the yaw angle which can not be determined based on the IMU measured gravitation vector. For mobile robots, proprioception is only relevant for odometry estimation in case the robot is ground-based.

High-level navigation

There are various visual cues which can be used for landmark-based navigation. Furthermore, there are algorithms known which use image differences to follow certain paths. Odor and taste are only useful if artificial markers are available. Passive acoustic localization requires also external sound sources which are not always provided. We are aiming for a more generic solution and, hence, we neglect these senses. Haptic sensors are interesting for close distance exploration, narrow spaces or manipulation, but they are not so useful for navigation in air, *e.g.*, for flying systems. The ground-based odometry estimates of proprioceptive sensors or prone to drift, but they have the advantage that they do not rely on the environment and, thus, are always available. Geometric cues from range sensors are usually not as reliable and robust for navigation, because either they provide too less variation on a noise-robust level of detail.

Obstacle detection

Visual systems provide obstacle detection in terms of the *time to impact* defined by the length of the optical flow vectors. However, again only objects with texture can be seen. Probably the most important directive for a mobile robot is not to harm a human being. Despite that, they are usually rather fragile and do not have a chitinous exoskeleton like insects, which allows them to bump into obstacles if vision fails (in which case tactile sensing would start playing a role). Hence, a fall-back sensor for obstacle avoidance would be useful, which replaces the chitin-shell. Obstacle detection is easiest provided by range sensing devices. For that we do not need a high resolution or accuracy - a noisier but compact and energy efficient sensor like sonar is sufficient for this purpose and could provide a sort of "virtual chitinous skin", which allows some software to detect a "virtual bumping" into an object and prevent the real contact.

Concluding, we can say, that a camera alone could solve all three tasks in a generic environment, but only based on one camera it is not possible to estimate the proper scale which is required by the control loop. Moreover, vision is not applicable to high dynamic systems due to motion blur and rather small sampling rates. Like every sensor, also cameras have some drawbacks, *e.g.*, environments with changing light conditions, poor texture or repetitive pattern which make the feature association quite error-prone. Hence, we define a minimal sensory equipment to reliably and robustly solve the navigation task by an IMU and a camera, as seen in insects. Cameras allow for a relative referencing, which prevents drifts as long as the same landmarks are visible, while IMUs permit robust motion estimates also at high dynamics. Further, IMUs may drift, but they never fail and always provide measurements at a constant rate. They also allow to

estimate the proper motion scale. Both are compact and passive sensors which perfectly enhance each other due to their complementary nature. Really high camera frame rates would be necessary to replace an IMU by a visual system and one would also loose the reliability gained by a fail-safe sensor. Anyway, high frame rates should be preferred to ease the data association problem in landmarks tracking. A wide aperture angle, as in omnidirectional cameras, provides a robust and reliable navigation and obstacle detection.

Range sensors provide a very accurate localization and, thus, a reliable motion estimation in indoor or object-dense environments (*e.g.* forests, cities, etc.), where enough measurements are provided by close landmarks. To increase the robustness of the system in case of failure, an obstacle detection based on range sensors, *e.g.* sonar, is recommendable. Any further sensor, like, *e.g.*, a magnetic compass or barometer, would of course improve the motion estimation and increase robustness but is beyond a minimalist sensor concept. To navigate between two points in space only as much accuracy in odometry estimation is necessary as it is required to provide a robust landmark association for exteroceptive navigation. Any drift of the egomotion estimation can then be compensated by the navigation algorithms.

Design of Biologically Inspired Algorithms

Unlike algorithms proposed and implemented by the SLAM community, insects seem to keep the task of motion estimation and landmark or map based navigation separated. They have techniques to estimate the egomotion and they have methods to memorize paths based on various landmarks and cues. They use basic principles, like global and local path integration, to achieve a high robustness and reliability in navigation, while keeping the processing costs low. Further, they do not build a geometric map, but rather store their paths separately and access them independently. This allows for a memory efficient navigation which is highly scalable. However, it is not efficient anymore as soon as there are too many, partly overlapping trajectories - some landmarks would be stored redundantly. In such cases a cognitive map would be more flexible (as, *e.g.*, proposed by [103]).

One major characteristic of the visual navigation of insects is that they adapt their motion to their neuronal processing ("algorithms"). They plan their trajectory continuously based on their current view. This allows to relax the motion problem and when repeating the path an insect does not even have to memorize how it moved the first place, but it just has to follow the same usual strategy and it will move the same way as before. It is not clear from the experiments, whether insects have implemented such

a concept which would ease memorization a lot. An indication for it is, that different ants choose almost the same paths to a certain destination, which solely depends on the available landmarks. On the other side, the experiments also show that insects memorize local vectors which describe the distance and direction from each landmark on. A combination of both techniques increases robustness and reliability.

We stated that a visual system should run at high speed, because the sensory system has to fit the dynamics of the motion, which eases processing. While insects evaluate every photoreceptor ("pixel") in parallel, conventional CPUs process the large amount of image-data sequentially. Hence, it is crucial to reduce the number of pixels to process, by selecting only pixels which contain relevant information. There is no need to process pixels showing a white wall, because no motion information can be gathered from it. By a sparse image evaluation, the processing costs per image are reduced, which allows for much higher frame rates with all the advantages mentioned before.

By physically separating rotation from translation, the motion estimation in insects becomes much easier than in the general case. Such a separation would significantly reduce the processing costs for motion estimation and increase its accuracy.

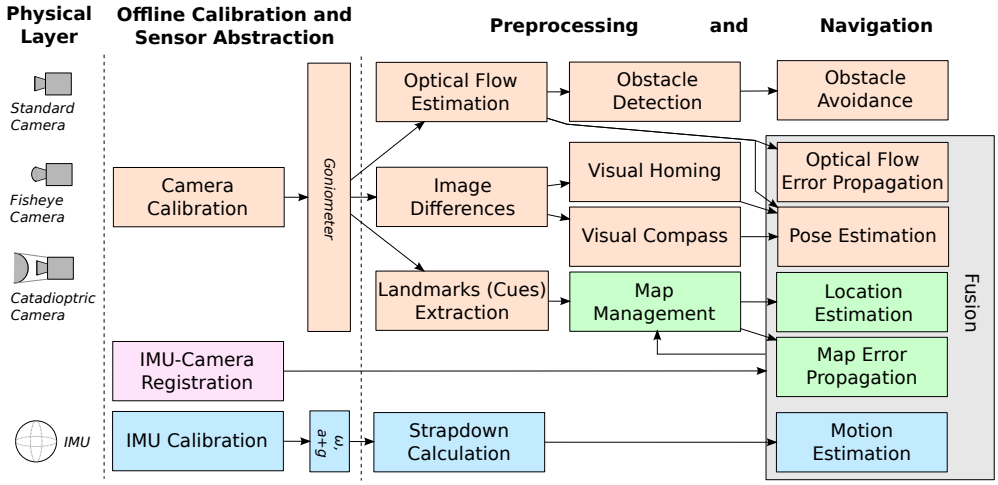


Figure 1.3: This flow diagram illustrates significant software modules for an efficient autonomous robot system which is based on a camera (red) and an IMU (blue). Basically all tasks can be achieved by processing the camera measurements only. Whereas, the IMU increases accuracy and robustness especially at high dynamics, and the systems reliability, because it is a fail-safe sensor which does not depend on the environment. The navigation map can be based on arbitrary sensors and, thus, these modules are colored in green.

Fig. 1.3 gives an overview about required software modules for our proposed insect inspired navigation. In this work, we are going to focus on the motion estimation

task which involves feature detection and tracking, IMU-camera calibration and sensor fusion with the aim to achieve high frame rates also on resource-limited mobile platforms by developing efficient image processing algorithms. A more detailed overview is given in the next section.

1.2 Outline – Towards Efficient Inertial-Visual Navigating Robots

In this thesis, we do not reason about how to implement special hardware, but from an algorithmic point of view we start at the lowest level evaluating available methods and compare them to our own solutions. Keeping the proposed, biologically-inspired minimal system in mind (see Fig. 1.3), we focus on the following modules: IMU-camera registration and fusion, optical flow computation and the pose estimation from it as well as its error propagation. Furthermore, we touch on the topics visual homing, obstacle detection and its avoidance based on optical flow. Our algorithms have been developed to allow for a reliable navigation also on resource-limited mobile platforms. They are designed under the premise of efficiency and reliability - aiming for minimalist systems providing adequate robustness. To preserve the generic aspects of the various algorithms, we will introduce them in a top-down manner.

In order to accurately fuse an IMU and a camera, they have to be temporally and spatially registered. We discuss the importance of proper temporal alignment and present solutions for it. Moreover, a closed-form solution for the spatial alignment is introduced, which can be used for a rough registration or the initialization of our novel batch-optimization based approach, which then is compared to state-of-the-art methods, proving the accuracy of the algorithm. The sensor measurements are fused in twofold ways, either by aiding landmark tracking or in an extended Kalman filter.

Aiming for an efficient image-based pose estimation, we analyze our own, biologically motivated approach, and various variants of the eight-point algorithm. We evaluate them with respect to robustness and accuracy under the aspects of aperture angle, the number of features and their detection accuracy. Furthermore, we derive a first order error-propagation, which estimates the expected motion accuracy based on the available feature correspondences.

We have identified high frame-rate image processing as one major key for robust and reliable visual motion estimation. The bottleneck in image-based motion computation is usually the low-level processing of the image. Only by an efficient detection of feature correspondences, we can provide high motion update rates. Hence, we will

again enhance state-of-the-art tracking algorithms and evaluate own solutions. The latter are developed based on the outcome of our accuracy evaluations and under the constraint of high frame rates. This new feature tracker can handle several hundreds of features in few milliseconds on low-power processing units.

The presented algorithms are used in various applications, like online 3D-modeling, virtual environment modeling for surprise detection and autonomous quadrocopter flights.

This yields the following structure for the rest of this thesis. First off, we introduce some mathematical concepts in Chapter 2 and related work is discussed in Chapter 3. Next, we start with our contributions, presenting techniques for IMU-camera registration and fusion in Chapter 4. In Chapter 5, we introduce our image-based motion estimation methods and derive the first order error propagations in Chapter 6. The last theoretic chapter, Chapter 7, tackles our low-level image processing with feature extraction and tracking. In Chapter 8, we experimentally validate the proposed methods and present some applications based on our algorithms in Chapter 9. Chapter 10 concludes this thesis and considers future work.

Chapter 2

Mathematical Framework and Concepts

In the following, the mathematical framework is introduced and some required mathematical concepts are sketched. The mathematical notation is explained and summarized in the same named section after the appendix. In addition, the illustration of samples by boxplots is explained in Appendix A.2.

2.1 Rigid Body Transformations and Coordinate Frames

Let $\boldsymbol{P}$ be a point in the three-dimensional Euclidean space, which is represented globally by a Cartesian coordinate frame, such that $\boldsymbol{P} \in \mathbb{R}^3$ and the elements of $\boldsymbol{P} = \begin{pmatrix} x_{\boldsymbol{P}} & y_{\boldsymbol{P}} & z_{\boldsymbol{P}} \end{pmatrix}^T$ denote the Cartesian coordinates. Any rigid transformation in 3D space can then be represented by a rotation and a translation [104, 105]:

$$\boldsymbol{P}' = \boldsymbol{R}\boldsymbol{P} + \boldsymbol{t}\,, \tag{2.1}$$

where $\boldsymbol{R} \in \mathrm{SO}(3, \mathbb{R})$ represents the rotation and $\boldsymbol{t} \in \mathbb{R}^3$ the translation. By extending the point representation to the so called homogeneous form $\boldsymbol{P} = \begin{pmatrix} x_{\boldsymbol{P}} & y_{\boldsymbol{P}} & z_{\boldsymbol{P}} & 1 \end{pmatrix}^T$ the rigid body transformation can be expressed by a single transformation matrix:

$$\boldsymbol{P}' = \boldsymbol{T}\boldsymbol{P} \quad \text{, with} \quad \boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} \tag{2.2}$$

Because it will always be clear from the context we renounce of marking point representations according to whether they are homogeneous or not.

Within this work we assume that the motion of all points in space is the same and, thus, they are all part of a rigid body. The question remains, which coordinate frame is used as reference for the description of motion or indication of points. For example, each sensor measures relative to its own coordinate system. In order to combine different measurements, the alignment between these coordinate frames has to be known. First, let us assume a global (world) coordinate frame $\mathcal{W}$ defining the global origin and the global x-, y-, and z-directions by its X-, Y-, and Z-axis. The alignment between each couple of coordinate frames can be described by a transformation matrix ${}^{A}\boldsymbol{T}_{B}$, where this matrix denotes the transformation from $\mathcal{B}$ to $\mathcal{A}$.

Leading superscripts denote the frame of reference, while successive superscripts denote the sensor which acquired the measurements. Hence, *e.g.* the rotational velocity at time instance t, measured by a camera but expressed in the IMU coordinate system $\mathcal{I}$ is described as ${}^{I}\boldsymbol{\omega}_{t}^{C}$. In case of translations, we require two successive subscripts to express the start and end frame of the translation. Thus, *e.g.* ${}^{W}\boldsymbol{t}_{CI}$ denotes the translation from the camera frame to the IMU frame expressed in the world frame.

In Eq. 2.1 we assume that the points move relative to the frame of reference. However, in mobile robot navigation the main task is to estimate the motion of the robot relative to some fixed landmarks. Thus, the landmarks are assumed to be static, while the sensors on the robot experience the motion. Hence, the motion of the points can be explained by the inverse of the motion of the sensor which measures the landmarks, leading to the following motion equation and the inverse transformation matrix

$$\boldsymbol{P'} = \boldsymbol{R}^{T}\left(\boldsymbol{P} - \boldsymbol{t}\right) \quad \text{and} \quad \boldsymbol{T}^{-1} = \begin{pmatrix} \boldsymbol{R}^{T} & -\boldsymbol{R}^{T}\boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix} . \tag{2.3}$$

We will choose between the representations in Eq. 2.1 and Eq. 2.3 depending on which formulation is more adequate and eases the understanding.

2.2 Rotation Representations

There are several ways how a rotation may be expressed. The different representations and their conversions are fairly well described in literature [105]. Hence, we will focus on the formulas and relations required in the next chapters. Following representations will be used: Euler angles, direction cosine matrices (DCMs), Euler angle-axis, Tsai angle-axis and quaternions.

To solve the ambiguity of sign in the angle-axis representations, the absolute angle is chosen to be always positive. The Tsai angle-axis representation $\boldsymbol{p}$ (as described

in [106]) of a rotation with absolute angle θ and unit length rotation axis $\hat{\boldsymbol{p}}$ can be computed by

$$\boldsymbol{p} = 2 \sin\left(\frac{\theta}{2}\right) \hat{\boldsymbol{p}} \quad | \quad \theta \geqslant 0 \tag{2.4}$$

A DCM is denoted by $\boldsymbol{R}$, which has the characteristic to be orthonormal, while a quaternion is usually represented by $\boldsymbol{q}$ and has the property to have unit length. The fourth element of the quaternions $\boldsymbol{q}_4$ represents $\cos\left(\frac{\phi}{2}\right)$, where ϕ is the absolute rotation angle, and, thus, the only real parameter. A bar over a quaternion denotes the inverse rotation

$$\bar{\boldsymbol{q}} = \begin{pmatrix} -q_1 \\ -q_2 \\ -q_3 \\ q_4 \end{pmatrix} . \tag{2.5}$$

The operator $\odot$ denotes the quaternion product so that ${}^{C}\boldsymbol{q}_A = {}^{C}\boldsymbol{q}_B \odot {}^{B}\boldsymbol{q}_A$.

The conversion from an arbitrary rotation representation into a DCM or a quaternion is described by the functions $\boldsymbol{R} = \mathrm{DCM}(\ldots)$ and $\boldsymbol{q} = \mathrm{Q}(\ldots)$, respectively.

Small Angle Approximations

Sometimes it is helpful to use approximations for small rotations. In the following we list a few relations which will be used later on:

$$\boldsymbol{\delta}_{\boldsymbol{q}} = \begin{pmatrix} x_{\boldsymbol{\sigma}} \sin\left(\frac{\sigma}{2}\right) \\ y_{\boldsymbol{\sigma}} \sin\left(\frac{\sigma}{2}\right) \\ z_{\boldsymbol{\sigma}} \sin\left(\frac{\sigma}{2}\right) \\ \cos\left(\frac{\sigma}{2}\right) \end{pmatrix} \underset{\sigma \approx 0}{\approx} \begin{pmatrix} \frac{x_{\boldsymbol{\sigma}}}{2} \\ \frac{y_{\boldsymbol{\sigma}}}{2} \\ \frac{z_{\boldsymbol{\sigma}}}{2} \\ 1 \end{pmatrix} , \tag{2.6}$$

whereas σ is the absolute rotation angle and $\hat{\boldsymbol{\sigma}}$ is the unit length rotation axis of the rotation $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}_{\boldsymbol{q}}$ denotes a small quaternion rotation. Such quaternions can then be represented by only their imaginary parts

$$\boldsymbol{\delta}_{\tilde{\boldsymbol{q}}} \overset{!}{=} \begin{pmatrix} \boldsymbol{\delta}_{\boldsymbol{q};1} \\ \boldsymbol{\delta}_{\boldsymbol{q};2} \\ \boldsymbol{\delta}_{\boldsymbol{q};3} \end{pmatrix} \underset{\sigma \approx 0}{=} \begin{pmatrix} \frac{x_{\boldsymbol{\sigma}}}{2} \\ \frac{y_{\boldsymbol{\sigma}}}{2} \\ \frac{z_{\boldsymbol{\sigma}}}{2} \end{pmatrix} . \tag{2.7}$$

Moreover, we use

$$\boldsymbol{\Delta}_{\boldsymbol{R}} \underset{\sigma \approx 0}{\approx} \boldsymbol{I}_3 + [\boldsymbol{\sigma}]_\times \underset{\sigma \approx 0}{\approx} \boldsymbol{I}_3 + 2\,[\delta_{\tilde{\boldsymbol{q}}}]_\times \quad \text{and} \quad \boldsymbol{\Delta}_{\boldsymbol{R}}^T \underset{\sigma \approx 0}{\approx} \boldsymbol{I}_3 - [\boldsymbol{\sigma}]_\times \underset{\sigma \approx 0}{\approx} \boldsymbol{I}_3 - 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}}}]_\times , \tag{2.8}$$

where $\boldsymbol{\Delta}_{\boldsymbol{R}}$ is a small rotation expressed as DCM.

2.3 Camera Model

In this work, we make use of the pinhole camera model [107, 108]. A pinhole camera provides a perspective projection from 3D to 2D. Any point $\boldsymbol{P}$ in 3D space is projected onto the image plane according to following formula

$$\tilde{\boldsymbol{r}} = f\frac{\boldsymbol{P}}{z_{\boldsymbol{P}}} = f\begin{pmatrix}\frac{x_{\boldsymbol{P}}}{z_{\boldsymbol{P}}}\\ \frac{y_{\boldsymbol{P}}}{z_{\boldsymbol{P}}}\\ 1\end{pmatrix} \stackrel{def}{=} f\boldsymbol{r} \tag{2.9}$$

where $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$ denote the projections of the 3D point onto the unit focal or real image plane in the camera coordinate system respectively.

The exact pixel location in the image coordinate frame, with pixel coordinates u and v, can, thus, be computed by

$$\boldsymbol{p} = \begin{pmatrix}u\\ v\end{pmatrix} = \begin{pmatrix}\frac{f}{s_x}\frac{x_{\boldsymbol{P}}}{z_{\boldsymbol{P}}} + o_x\\ \frac{f}{s_y}\frac{y_{\boldsymbol{P}}}{z_{\boldsymbol{P}}} + o_y\end{pmatrix} = \begin{pmatrix}f_x\frac{x_{\boldsymbol{P}}}{z_{\boldsymbol{P}}} + o_x\\ f_y\frac{y_{\boldsymbol{P}}}{z_{\boldsymbol{P}}} + o_y\end{pmatrix} \tag{2.10}$$

with s_x, s_y being the pixel size in x- or y-direction, while o_x, o_y represents the pixel-offset of the principle point to the upper left corner, which is the origin of the image coordinate frame $\mathcal{C}$. This relation can also be written in matrix form, resulting in the so called intrinsic camera parameter matrix

$$\boldsymbol{K} = \begin{pmatrix}f_x & s & o_x\\ 0 & f_y & o_y\\ 0 & 0 & 1\end{pmatrix}. \tag{2.11}$$

The skew factor s, which allows to model a relation between the X- and Y-axis can usually be neglected, because such a relation only exists if the chip is not mounted parallel to the lens as it usually only occurs in cameras of low quality. Hence, this factor has not been listed in Eq. 2.10 for the sake of clarity. However, there is a further step necessary to provide the nonlinear mapping of the division:

$$\boldsymbol{p} = \begin{pmatrix}\frac{u'}{w'}\\ \frac{v'}{w'}\end{pmatrix}, \text{ with } \begin{pmatrix}u'\\ v'\\ w'\end{pmatrix} = \boldsymbol{K}\boldsymbol{r}\,. \tag{2.12}$$

The *intrinsic parameters* consist also of the radial distortion introduced by the lens. In this work, we assume in general already undistorted points, if not mentioned differently. The transformation parameters, a rotation ${}^{C}\boldsymbol{R}_{W}$ and a translation ${}^{C}\boldsymbol{t}_{W}$ or a

homogeneous transformation matrix ${}^{I}\boldsymbol{T}_{W}$, denoting the spatial alignment of a camera to its frame of reference (*e.g.* the world coordinate frame, a second camera, a robot, or an other device) are called *extrinsic parameters*. Fig. 2.1 sketches the pinhole camera model with the involved coordinate frames.

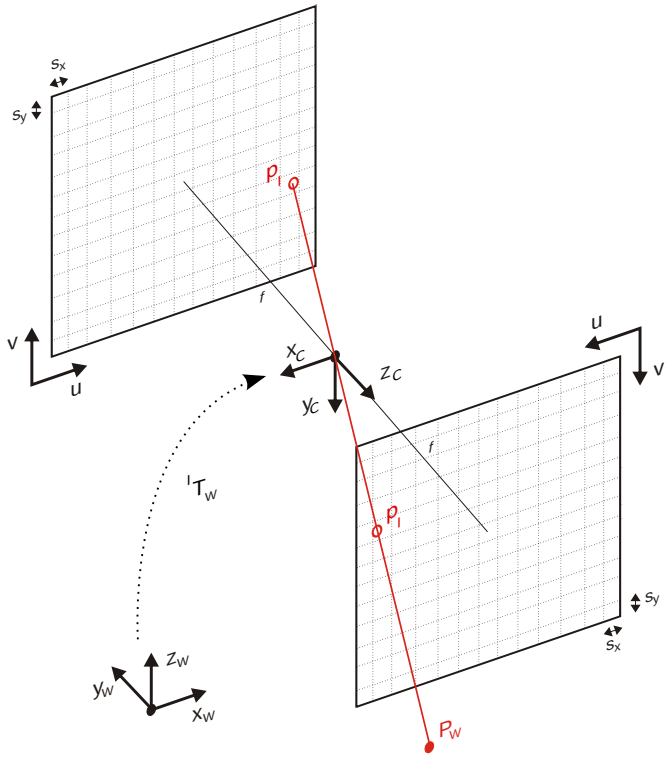


Figure 2.1: This drawing illustrates the camera projection with the intrinsic and extrinsic camera parameters. While the real image plane is behind the optical center (the pinhole, the point through which all the rays go), it is usually easier for imagination to assume a plane which is in front of the optical center. In this plane all the objects are upright and not mirrored as in the real image plane.

We calibrate the intrinsic and extrinsic parameters of the camera with *CalDe* and *CalLab* [109].

2.4 Optical Flow

Optical flow is the pattern of apparent motion in a visual scene caused by the relative motion between an observer, *e.g.* eye or camera, and the scene [107, 108]. Any element of the observer gets connected to the element where the projected point in space appeared in the last time-step, which yields a dense vector field between two images. These so

called *optical flow vectors* contain information about the relative rotation, translation and distance of the observer to any object in the scene. The length of an optical flow vector is defined by the ratio between the relative velocity of the projected point and its distance and, thus, represents the *time-to-collision*. This information can be used for obstacle detection. Sparse optical flow denotes the variant where the optical flow is computed only for a few elements (*e.g.* pixels of an image) and not for the whole projection (dense optical flow).

Assuming a static environment this concept can also be used for motion estimation. If only translation is applied, all optic flow vectors intersect in two points on a sphere, the *focus of expansion* (FOE) and *focus of contraction* (FOC), whereas at most one of them is visible in pinhole cameras where it is called *epipole*. The axis through the optical center and the FOE denotes the direction of motion, thus, the orientation of the optical flow vectors relative to the epipole reveals the sign of the direction of motion. If the vectors are pointing away from the epipole, the motion is towards it (epipol equals FOE), and otherwise away from it (epipol equals FOC). If a general motion is applied, the computation of rotation and translation becomes more complicated, but it is still possible by closed form solutions as we will see in Section 3.3.

2.5 RANSAC

The *Random Sampling Consensus* algorithm (RANSAC) is an iterative framework to find parameters for a given model from a data set including outliers [110]. In each iteration, a smallest possible data subset is randomly chosen to calculate the model parameters. This model is then applied to the residual data set and the elements are split into fitting elements (inliers) and non-fitting elements (outliers). This step is repeated several times and the best model, according to the number of inliers and their residual error, is chosen. Preemptive RANSAC is a variant which allows the algorithm to leave the loop in case that a certain quality criterion applies also before the maximum number of iterations is reached.

2.6 IMU Strapdown Computation

An inertial measurement unit measures only 3D motion derivatives: the angular velocity, as the first derivative of the attitude, is provided by the gyroscopes and the translational acceleration, as the second derivative of the position, is gathered by the accelerometers. These measurements are acquired in the IMU-frame (or *body-frame*) $\mathcal{I}$, the coordinate frame which is rigidly attached to the sensor unit and, thus, to the

moving platform. The measurements are in respect to a fixed reference frame $\mathcal{F}$, the so called *inertial-frame*, which has its origin in the center of the Earth and a rigid orientation in respect to the fixed stars. Another important reference frame is the *navigation-frame* $\mathcal{N}$. Its origin corresponds to the one of $\mathcal{I}$, but the X- and Y-axis are aligned with the North and East direction, respectively, while the Z-axis points to the Earth center.[1]

The accelerometers, however, measure not only the forces due to translational acceleration, but also the earth's gravity force and the Coriolis force, which results from the Earth's rotation. In a strapdown computation, one computes the pose from the measurements of an IMU [111]. For that the rotation and the translational velocity has to be propagated from the former measurements and the Coriolis and gravity force compensated motion derivatives are integrated. In presence of vibrations and if the strapdown computation runs slower than the IMU sampling rate, also other effects, like *coning* and *sculling*, play an important role (the interested reader is referred to [111] for further explanations). For the sake of compactness, weight and price we will use sensors which are based on the technology of *micro-electro-mechanical systems* (MEMS), but which are more noisy than, *e.g.*, optical IMUs and suffer from biases. We assume to sample the IMU at its maximum rate and, hence, we will neglect aforementioned effects as well as the influence of Earth rotation in our strapdown computation. This results in following strapdown equations.

The attitude quaternion at time-step $k+1$ can be computed by

$$ {}^{N}\boldsymbol{q}_{I;k+1} \;=\; {}^{N}\boldsymbol{q}_{I;k} \odot {}^{I;k}\boldsymbol{q}_{I;k+1} \tag{2.13} $$

where ${}^{I;k}\boldsymbol{q}_{I;k+1}$ results from the measured rotational velocity ${}^{I}\boldsymbol{\omega}^{I}$, which is assumed to be constant for the sample-time Δt, by the small-angle approximation

$$ {}^{I;k}\boldsymbol{q}_{I;k+1} \;=\; \begin{pmatrix} \cos\left(\frac{\|\boldsymbol{\sigma}\|}{2}\right) \\ \frac{\boldsymbol{\sigma}}{\|\boldsymbol{\sigma}\|}\sin\left(\frac{\|\boldsymbol{\sigma}\|}{2}\right) \end{pmatrix} \quad \text{with} \quad \boldsymbol{\sigma} \;=\; {}^{I}\boldsymbol{\omega}^{I}\,\Delta t\;. \tag{2.14} $$

The new position of the body-frame ${}^{N}\boldsymbol{t}_{FI;k+1}$ can be computed by integrating the current velocity ${}^{N}\boldsymbol{v}_{FI;k}$ and the measured translational acceleration ${}^{N}\boldsymbol{a}_{FI;k+1}$, which is also assumed to be constant over the sampling time,

$$ {}^{N}\boldsymbol{t}_{FI;k+1} \;=\; {}^{N}\boldsymbol{t}_{FI;k} \;+\; {}^{N}\boldsymbol{v}_{FI;k}\,\Delta t \;+\; \frac{1}{2}\,{}^{N}\boldsymbol{a}_{FI;k+1}\,\Delta t^{2} \tag{2.15} $$

[1] For the sake of simplification a fixed world reference frame $\mathcal{W}$ might be assumed in other chapters with an arbitrary origin and orientation.

where

$$ {}^{N}\boldsymbol{v}_{FI;k+1} = {}^{N}\boldsymbol{v}_{FI;k} + \left(\boldsymbol{I}_3 + {}^{N}\boldsymbol{R}_{I;k+1}\frac{1}{2}\left[\boldsymbol{\sigma}\right]_{\times}\right){}^{N}\boldsymbol{a}_{FI;k+1}\,\Delta t \tag{2.16} $$

and

$$ {}^{N}\boldsymbol{a}_{FI;k+1} = {}^{N}\boldsymbol{R}_{I;k+1}{}^{I}\boldsymbol{a}^{I}{}_{k+1} + {}^{N}\boldsymbol{g} \tag{2.17} $$

with ${}^{I}\boldsymbol{a}^{I}{}_{k+1}$ the measured acceleration and ${}^{N}\boldsymbol{g} = (0\ 0\ 9.81)^{T}$ the gravitation vector.

For a derivation of these equations please refer to [111].

2.7 Kalman Filter Equations

The Kalman filter is an optimal filter for linear problems with zero-mean, white Gaussian noise [112]. A lot of work can be found in literature about this data fusion concept [111, 113]. In this section, we will only mention the two steps, the propagation and the update, for the indirect (or error-state) Kalman framework and its general equations which we will in the following refer to. Any Kalman filter variation applied in this work will be introduced and discussed at the point we will use it.

Propagation

The Kalman filter requires a linear dynamic model which describes the transition from the error state $\boldsymbol{\delta}^{+}_{\boldsymbol{x};k}$ of time instance k to $\boldsymbol{\delta}^{-}_{\boldsymbol{x};k+1}$

$$ \boldsymbol{\delta}^{-}_{\boldsymbol{x};k+1} = \boldsymbol{F}_{k+1}\boldsymbol{\delta}^{+}_{\boldsymbol{x};k} + \boldsymbol{G}_{k+1}\boldsymbol{n}_{\boldsymbol{x},k}\ , \tag{2.18} $$

where $\boldsymbol{F}_{k+1}$ is the state transition matrix and $\boldsymbol{G}_{k+1}$ maps the system noise $\boldsymbol{n}_{\boldsymbol{x},k}$ to the state space. The covariance matrix $\boldsymbol{P}^{+}_{k}$ propagates as follows:

$$ \boldsymbol{P}^{-}_{k+1} = \boldsymbol{F}_{k+1}\boldsymbol{P}^{+}_{k}\boldsymbol{F}^{T}_{k+1} + \boldsymbol{G}_{k+1}\boldsymbol{Q}_{k}\boldsymbol{G}^{T}_{k+1} \tag{2.19} $$

with $\boldsymbol{Q}_k$ denoting the system noise matrix.

Update

As soon as a measurement $\boldsymbol{z}_{k+1}$ is acquired, the residual $\boldsymbol{r}_{k+1}$ between it and the estimated measurement $\widehat{\boldsymbol{z}}_{k+1}$ is computed by

$$ \boldsymbol{r}_{k+1} = \boldsymbol{z}_{k+1} - \widehat{\boldsymbol{z}}_{k+1}\ . \tag{2.20} $$

Now, an update step can be performed, which fuses the propagated state and the acquired measurement. For that we compute the innovation $\boldsymbol{S}_{k+1}$ by the measurement matrix $\boldsymbol{H}_{k+1}$, which represents the mapping between the state space and the measurement space, and the covariance matrix $\boldsymbol{L}_{k+1}$ of the new measurement

$$\boldsymbol{S}_{k+1} = \boldsymbol{H}_{k+1}\boldsymbol{P}^{-}_{k+1}\boldsymbol{H}^{T}_{k+1} + \boldsymbol{L}_{k+1} \, . \tag{2.21}$$

The so called Kalman gain $\boldsymbol{K}_{k+1}$

$$\boldsymbol{K}_{k+1} = \boldsymbol{P}^{-}_{k+1}\boldsymbol{H}^{T}_{k+1}\boldsymbol{S}^{-1}_{k+1} \tag{2.22}$$

is then used for the covariance update

$$\boldsymbol{P}^{+}_{k+1} = \boldsymbol{P}^{-}_{k+1} - \boldsymbol{K}_{k+1}\boldsymbol{H}_{k+1}\boldsymbol{P}^{-}_{k+1} \tag{2.23}$$

as well as the state update

$$\boldsymbol{\delta}^{+}_{\boldsymbol{x};k+1} = \boldsymbol{\delta}^{-}_{\boldsymbol{x};k+1} + \boldsymbol{K}_{k+1}\,\boldsymbol{r}_{k+1} \, . \tag{2.24}$$

Chapter 3

Related Work

In this section, we are going to discuss relevant approaches in literature and state-of-the-art solutions for the different topics we are treating within this work. These include image features and their tracking techniques, motion estimation algorithms and sensor fusion relevant methods.

3.1 Image Features

Thinking of a camera as a sensor for motion estimation, one prerequisite is inevitable: a relation between two images has to be established. The way how such a relation is determined bears probably the most significant aspect regarding robustness, reliability and accuracy in image based navigation. Either every pixel of the whole image is used, which is only feasible on parallel processing units, or with the reduction of the amount of pixels to process by evaluating their relevance. The properties commonly considered for this purpose are intensity, color, and texture. An exception is the feature proposed by Kovesi [114, 115], which is based on the phase congruency of the image's frequencies. These features are lighting insensitive, but require an evaluation of the image in the frequency domain, which is processing intensive. The detection of color or texture features bears in general a computational overhead and, thus, we do as well not consider them here.

In Section 1.1.2, we have seen, that from a biological point of view there are three characteristics of a possible landmark which have to be considered: salience, permanence and reliability. Salience is probably the most crucial property. The more salient a gradient is, the more accurate its location and, hence, the motion can be determined. The permanence of a feature is an aspect which can only be evaluated over time and, thus, is only crucial for landmark-based navigation but not for visual odometry. The

reliability of feature detectors is usually referred to as *repeatability* by the computer vision community. This value represents how often, and, thus, how reliable, a feature can be found in a sequence of images.

Compared to edges and color cues, corners are more accurate and provide a two dimensional constraint. Considering corners as intersection of two edges, these features have no spatial extension and, therefore, there is no ambiguity in their location. Of course, this aspect is only valid, if the locality of a corner is preserved and the response of a detector is as close as possible to the real corner location. Several different approaches of corner extraction are known in literature. All aim for an efficient, accurate and/or reliable detection – three rather conflicting characteristics. In the following, we will mention the most popular feature detectors and explain more in detail the ones related to our proposed solutions. For a more comprehensive survey please refer to [116].

The Harris corner detection algorithm is probably one of the most popular corner extraction methods [117]. It is based on the first order Taylor expansion of the second derivative of the local sum of squared differences (SSD). The eigenvalues of this linear transformation reveal how much the SSD approximation varies if the patch would be shifted along the image axes. There are solutions which interpret the eigenvalues based on a threshold [118] or without [119].

So called global matching algorithms allow features to be detected within the whole image. Therefore, a corner detector has to provide a high repeatability, so that it detects the same features also after large affine transformations. The global tracker SIFT [120, 121] (*scale invariant feature transform*) uses difference of Gaussians (DoG), while the faster SURF [122, 123] (*speeded up robust features*) uses a Haar wavelet approximation of the determinant of the Hessian. A generalization of the DoG detector for box- and octagon-like shapes is presented as CenSurE [124] (*center surround extrema*).

Smith developed the so called *smallest uni-value segment assimilating nucleus* test (SUSAN) [125] for corner detection. The brightness of the center pixel, the nucleus, is compared to its circular pixel neighborhood, and the area of the uni-value segment assimilating nucleus (USAN) is computed. Corner and edges can be detected by evaluating this area, or it can be used for noise reduction, too. The advantages of this approach are, that neither noise sensitive derivation nor other computationally expensive operations have to be performed. In [125], a circular disc with diameter 3.4 is used, which yields a total area of 37 pixels.

In the last decade, the processing power of standard computers has become fast enough to provide corner extraction at video rate. However, running conventional corner detection (*e.g.* the Harris corner detector) and performing other intensive tasks, is still computationally infeasible on most single processor. With the introduction of

recent techniques such as Rosten's *features from accelerated segment test* (FAST) [126], feature extraction has seen a significant performance increase for real-time Computer Vision applications. While being efficient, this method has proven in several applications to be reliable due to high repeatability [127].

In this thesis, we are going to present a novel corner detection approach, which is based on the same corner criterion as FAST, but which provides a significant *performance increase* for *arbitrary* images. Unlike FAST, the corner detector does not have to be trained for a specific scene, but it *dynamically adapts* to the environment while processing an image. For a better understanding of our approach, we will sketch FAST in more detail in the next section.

3.1.1 FAST Revisited

The FAST principle is based on the SUSAN corner detector. Again, the center of a circular area is used to determine brighter and darker neighboring pixels. However, in the case of FAST, not the whole area of the circle is evaluated, but only the pixels on the discretized circle describing the segment. Like SUSAN, FAST uses a Bresenham's circle of diameter 3.4 pixels as test mask. Thus, for a full accelerated segment test 16 pixels have to be compared to the value of the nucleus. To prevent this extensive test, the corner criterion has been even more relaxed. The criteria for a pixel to be a corner according to the accelerated segment test (AST) is as follows: there must be at least S connected pixels on the circle, which are brighter or darker than a threshold determined by the center pixel value. The values of the other $16 - S$ pixels are disregarded. Therefore, the value S defines the maximum angle of the detected corner. Keeping S as large as possible, while still suppressing edges (edges result in $S = 8$), increases the repeatability of the corner detector. Thus, FAST with segment size 9 (FAST-9) is usually the preferred version, and is also used in our experiments unless otherwise stated. The AST applies a minimum difference threshold (t), when comparing the value of a pixel on the circular pattern with the brightness of the nucleus. This parameter controls the sensitivity of the corner response. A large t-value results in few but strong corners, while a small t-value yields also corners with smoother gradients. In [127] it is shown, that the AST with $S = 9$ has a high repeatability, compared to other corner detectors as, *e.g.*, Harris, DoG, or SUSAN. The repeatability of a corner detector is a quality criterion which measures the capability of a method to detect the same corners of a scene from varying viewpoints.

One question still remains, namely which pixel to compare first, second, third, and so forth. Obviously, there is a difference in speed, whether one consecutive pixel

after another is evaluated or, *e.g.*, bisection on the circle pattern is used to test if the corner criterion applies or cannot apply anymore. This kind of problem is known as constrained twenty questions paradigm. When to ask which question results in a decision tree with the aim to reduce its average path length. In [128], Rosten uses ID3 [129], a machine learning method, to find the best tree based on training data of the environment where FAST is applied. Doing so, it is not guaranteed that all possible pixel configurations are found (see Section 8.1.2). Already small rotations of the camera may yield pixel configurations which have not been measured in the test images. Even if all the pixel configurations are present, a small rotation about the optical axis would drastically change the probability distribution of the measured pixel configurations. This may result in an incorrect and slow corner response. To learn the probabilistic distribution of a certain scene is, therefore, not applicable, unless only the same viewpoints and the same scene are expected. Please note that the decision tree is optimized for a specific environment and has to be re-trained every time it changes to provide the best performance.

The decision tree learning used by the FAST algorithm builds a *ternary* tree with possible pixel states "darker", "brighter" and "similar". At each learning step, *both* questions, "is brighter" and "is darker", are applied for all remaining pixel and the one with the maximum information gain is chosen. Hence, the state of each pixel can be one of four possibilities: unknown (u), darker (d), brighter (b) or similar (s). In the following, we call a combination of N states a pixel *configuration*. The size of the configuration space is therefore 4^N, which yields $4^{16} \approx 4 \cdot 10^9$ possible configurations for $N = 16$. For the rest of this thesis, we refer to this model as restricted or four states configuration space.

FAST-ER, the most recent FAST derivation, has even a slightly increased repeatability, compared to FAST-9, at the cost of computational performance [127]. The main difference is the thickness of the Bresenham's circle, that has been increased to 3 pixels. This results again in a more SUSAN-like algorithm, which spans a circular area of 56 pixels, disregarding the inner 3x3 pixels. Again, ID3 is used to build the decision tree, restricting the evaluation to only a small part of the 47 pixels.

3.2 Feature Trackers

In the previous section, we discussed image cues, which are salient enough to be re-detected in different images. Feature trackers describe techniques how features can be followed from image to image and how the data association problem between the

feature sets of two images can be solved. There is a large number of approaches available in literature, which can be categorized into dense or sparse optical flow methods. Similar concepts arise for both types. The only difference is that the dense approaches, *e.g.* [130], can use the proximity constraint to solve ambiguities and/or to smooth the result, *e.g.* by using total variation [131]. Further, feature trackers can be divided into

- **global trackers**, which consist of a detector and a descriptor. They can find features in images which have been acquired from two quite different viewpoints (*e.g.* SIFT [120, 121], SURF [122, 123], BRISK [132], *etc.*) and
- **local trackers**, which are much more efficient and use *gradient descent techniques* to find the new feature location (KLT [133], SSD [134], *etc.*) or *tracking by matching* methods, which provide similar to global trackers a detection and description stage, but the descriptor can be significantly relaxed, because they also use the local proximity condition of the features (*e.g.* PTAM [135]).

Global tracking methods use computationally expensive descriptors and, hence, their matching is in general not real-time capable. Only recently, an efficient global matching algorithm, BRISK [132] (*binary robust invariant scalable keypoints*), has been proposed, which uses an efficient corner detector, namely AGAST (which we will present in Section 7.2.1), and, thus, allows online feature tracking at 25 Hz on a conventional computer. The descriptor is a smart combination of the rotation-invariant BRIEF [136] (*binary robust independent elementary features*) and the multi-scale DAISY [137]. BRIEF is the most efficient global feature descriptor, a derivative of LBP [138] (*local binary patterns*), which allows an efficient matching due to binary operations on gray-scale comparisons. It is not rotation invariant, but has proven to yield comparable results to upright SURF (U-SURF), a rotation-invariant variation of SURF, while being about 25 times faster than U-SURF. However, even though BRISK allows for an image processing at video-rates, it is not feasible for high frame-rates on embedded systems. For the sake of efficiency and generality, we are aiming for sparse image processing of subsequent images of a sequence, focusing only on salient pixels or pixel-groups. Hence, we do not require global features and, thus, we will not further discuss global tracking algorithms and their descriptors but focus on local tracking approaches, which are relevant for our work.

The Kanade-Lucas-Tomasi (KLT) tracker uses gradient images to approach the corresponding image patch by an iterative gradient descent method. For the derivation of the original formulas refer to [133, 139]. In general, Shi-Tomasi features are used for KLT tracking, which are Harris corners with the smallest eigenvalue larger than a

specified threshold [118]. These features have been derived to optimally fit the KLT tracker. The so called Bouguet-implementation of KLT is a more efficient, pyramidal implementation of the tracker, which is based on a few assumptions and modifications of the derivation [140].

Davison's *MonoSLAM*[1] algorithm uses Shi-Tomasi features to implement tracking by matching [141], while Klein's *parallel tracking and mapping* (PTAM)[1] algorithm uses FAST features for this purpose [135]. FAST features are more efficient to compute, which allows to process much more features. However, a large number of features worsens the data association problem and, hence, PTAM uses image pyramids with several levels to limit the search areas for the features. In both cases SSD, is used to evaluate whether the features correspond or not.

3.3 Motion Estimation Techniques

Once feature correspondences between two images are known and the environment can be assumed to be rigid, the relative motion of the camera between the locations of image acquisition can be estimated. However, this information is not directly accessible due to the perspective projection from 3D to 2D of a regular camera. As introduced in Section 2.3, one degree of freedom, namely the distance, is lost.

An efficient way of computing the motion between two images are closed form solutions. However, the aforementioned missing information gives space to ambiguities and error sensitivities, which we will evaluate in Section 8.2. Solutions based on nonlinear optimization achieve a higher accuracy, but they are computationally more expensive and require in general a measurement history. If the 3D structure of the landmarks is known for both images, the problem becomes reduced to a registration of two 3D point-clouds. If the 3D structure is only available for one image, the task can be solved by iterative methods. Such solutions are much more stable, but require a landmark initialization.

There exist also egomotion estimation techniques based on dense optical flow, *e.g.* [142], which we disregard at this point, because they are computationally too expensive on generic, sequential platforms. In addition, special motion estimation algorithms have been developed for omnidirectional cameras [143, 92, 93]. These algorithms, however, significantly lose performance as soon as the aperture angle becomes reduced. Even if we aim for camera systems with a large FOV, we can not guarantee, that there is adequate texture in the environment which makes the full view usable. Hence, we will focus on generic solutions, which work for arbitrary aperture angles.

[1] We will discuss these approaches more in detail in Section 3.3.3.

In the following, we derive the well-known eight-point algorithm and some of its variants, which we will evaluate in Section 8.4.1. Furthermore, we will discuss state-of-the-art optimization and iterative approaches. At the end of this part, we will briefly describe the concept of active matching.

3.3.1 Closed Form Solutions

According to Eq. 2.1, which represents the rigid body transformation, and the camera projection, described in Eq. 2.9, the components $\boldsymbol{Rr}$, $\boldsymbol{r}'$ and $\boldsymbol{t}$ must be coplanar and, thus, linear dependent. Consequently, their scalar triple product must be zero:

$$\boldsymbol{r}'^{T}\left(\boldsymbol{t}\times\boldsymbol{Rr}\right)=\boldsymbol{r}'^{T}\left[\boldsymbol{t}\right]_{\times}\boldsymbol{Rr}=0 \tag{3.1}$$

This yields a fundamental linear mapping between two images: the *Essential matrix* $\boldsymbol{E}$, which describes the mapping between two sets of points on the unit focal image plane in the camera coordinate frame and the *Fundamental matrix* $\boldsymbol{F}$, which denotes the mapping between two sets of points in the image coordinate frame [144]

$$\boldsymbol{K}^{T}\boldsymbol{F}\boldsymbol{K}\;=\;\boldsymbol{E}\;=\;\left[\boldsymbol{t}\right]_{\times}\boldsymbol{R}\,, \tag{3.2}$$

with $\boldsymbol{K}$ introduced in Eq. 2.11. At this point, we want to mention that the mapping $\boldsymbol{Fp}\;=\;\boldsymbol{e}$ yields the so called *epipolar line* $\boldsymbol{e}$ and the point $\boldsymbol{p}'$ needs to lie on this line resulting in a dot product of zero. An extensive derivation of this mapping and a summary of the properties of the Fundamental and Essential matrix can be found in [108]. We are interested in estimating the relative motion in 3D and, thus, in the following, we assume calibrated cameras and undistorted image features, which allows us to estimate the Essential matrix $\boldsymbol{E}$.

By mapping the matrix $\boldsymbol{E}=\{E_{ij}\}$ column-wise into a single column vector

$$\boldsymbol{E}\rightarrow\boldsymbol{e}=\begin{pmatrix}E_{11} & E_{21} & E_{31} & E_{12} & E_{22} & E_{32} & E_{13} & E_{23} & E_{33}\end{pmatrix}^{T}=\begin{pmatrix}e_{1} & \dots & e_{9}\end{pmatrix}^{T} \tag{3.3}$$

we get a linear system of equations

$$\boldsymbol{A}\;=\;\begin{pmatrix}u_{\boldsymbol{r}}u_{\boldsymbol{r}'} & u_{\boldsymbol{r}}v_{\boldsymbol{r}'} & u_{\boldsymbol{r}} & v_{\boldsymbol{r}}u_{\boldsymbol{r}'} & v_{\boldsymbol{r}}v_{\boldsymbol{r}'} & v_{\boldsymbol{r}} & u_{\boldsymbol{r}'} & v_{\boldsymbol{r}'} & 1\end{pmatrix} \tag{3.4}$$

with $\boldsymbol{Ae}=0$. Furthermore, the notation $u_{\boldsymbol{r}}$ and $v_{\boldsymbol{r}}$ denotes the x- and y-coordinates in the image, corresponding to the image ray $\boldsymbol{r}$. To solve this system of equations one

needs at least eight point correspondences stacked on top of each other

$$\boldsymbol{A} = \begin{pmatrix} u_{\boldsymbol{r}_1} u_{\boldsymbol{r}'_1} & u_{\boldsymbol{r}_1} v_{\boldsymbol{r}'_1} & u_{\boldsymbol{r}_1} & v_{\boldsymbol{r}_1} u_{\boldsymbol{r}'_1} & v_{\boldsymbol{r}_1} v_{\boldsymbol{r}'_1} & v_{\boldsymbol{r}_1} & u_{\boldsymbol{r}'_1} & v_{\boldsymbol{r}'_1} & 1 \\ u_{\boldsymbol{r}_2} u_{\boldsymbol{r}'_2} & u_{\boldsymbol{r}_2} v_{\boldsymbol{r}'_2} & u_{\boldsymbol{r}_2} & v_{\boldsymbol{r}_2} u_{\boldsymbol{r}'_2} & v_{\boldsymbol{r}_2} v_{\boldsymbol{r}'_2} & v_{\boldsymbol{r}_2} & u_{\boldsymbol{r}'_2} & v_{\boldsymbol{r}'_2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{\boldsymbol{r}_N} u_{\boldsymbol{r}'_N} & u_{\boldsymbol{r}_N} v_{\boldsymbol{r}'_N} & u_{\boldsymbol{r}_N} & v_{\boldsymbol{r}_N} u_{\boldsymbol{r}'_N} & v_{\boldsymbol{r}_N} v_{\boldsymbol{r}'_N} & v_{\boldsymbol{r}_N} & u_{\boldsymbol{r}'_N} & v_{\boldsymbol{r}'_N} & 1 \end{pmatrix} \tag{3.5}$$

As Eq. 3.1 reveals, the Essential matrix has only five degrees of freedom. Hence, less than eight points should be necessary to estimate E depending on which and how many constraints of the Essential matrix are used. There exist so called seven-, six- and five-point algorithms [145, 146, 147]. These algorithms are, in general, computationally more expensive and more sensitive to measurement errors [148]. Therefore, we will focus on the eight-point algorithm and its variants [149, 150].

Eight-Point Algorithm

The eight-point algorithm basically consists of solving the homogeneous linear system denoted in Eq. 3.5. Due to measurement noise, this system will in general not be zero. Thus, we need to solve for

$$\boldsymbol{e} = \arg\min_{\boldsymbol{e}} \left(\|\boldsymbol{A}\boldsymbol{e}\| \right) \tag{3.6}$$

subject to the constraint $\|\boldsymbol{e}\| = 1$. The solution to this constrained minimization problem, obtained by means of Lagrangian multipliers, is given by the eigenvector corresponding to the smallest eigenvalue of $\boldsymbol{A}^T\boldsymbol{A}$ [151]. Hence, the solution can be found by a singular value decomposition of $\boldsymbol{A}$ into

$$\mathrm{SVD}\left(\boldsymbol{A}\right) = \boldsymbol{U}_{\boldsymbol{A}} \boldsymbol{\Sigma}_{\boldsymbol{A}} \boldsymbol{V}_{\boldsymbol{A}}^T \, . \tag{3.7}$$

and using the last column of $\boldsymbol{V}_{\boldsymbol{A}}$

$$\boldsymbol{e} = \boldsymbol{V}_{\boldsymbol{A},-9} \, . \tag{3.8}$$

In some literature, you find also a factor of $\sqrt{2}$ in above formula. This factor would enforce the Essential Matrix to have $\|\boldsymbol{E}\|_F = \sqrt{2}$ according to

$$\|\boldsymbol{E}\|_F^2 = \|\boldsymbol{E}^T\|_F^2 = \mathrm{tr}\left(\boldsymbol{E}\boldsymbol{E}^T\right) = \mathrm{tr}\left([\boldsymbol{t}]_\times \boldsymbol{R}\boldsymbol{R}^T [\boldsymbol{t}]_\times^T\right) = \mathrm{tr}\left([\boldsymbol{t}]_\times \left(-[\boldsymbol{t}]_\times\right)\right) = 2 \tag{3.9}$$

with $\boldsymbol{t}$ having unit length. We will take this normalization into consideration in a few steps.

A major improvement of the conventional eight-point algorithm was first proposed by Hartley [149]. He observed, that the image coordinates are usually of different

magnitude than the focal length, which leads to an ill-posed singular value decomposition. Hence, Hartley suggested, in defense of the eight-point algorithm, to transform the input data before estimating the Essential matrix. Introducing two transformation matrices, $\boldsymbol{S}$ and $\boldsymbol{S}'$, we are able to manipulate the input of the algorithm, which improves the computational condition for estimating $\boldsymbol{E}$:

$$\begin{aligned} \boldsymbol{r}'^{T}\boldsymbol{E}\boldsymbol{r} &= 0 \qquad (3.10)\\ \left(\boldsymbol{S}'\boldsymbol{r}'\right)^{T}\ \boldsymbol{S}'^{-T}\boldsymbol{E}\boldsymbol{S}^{-1}\ \boldsymbol{S}\boldsymbol{r} &= 0\\ \boldsymbol{r}'^{T}\boldsymbol{S}'^{T}\ \boldsymbol{S}'^{-T}\boldsymbol{E}\boldsymbol{S}^{-1}\ \boldsymbol{S}\boldsymbol{r} &= 0\\ \hat{\boldsymbol{r}}'^{T}\hat{\boldsymbol{E}}\hat{\boldsymbol{r}} &= 0\ . \qquad (3.11) \end{aligned}$$

The matrix $\boldsymbol{E}$ can then be retrieved by

$$\boldsymbol{E} = \boldsymbol{S}'^{T}\hat{\boldsymbol{E}}\boldsymbol{S}\ . \qquad (3.12)$$

In order to make the computation as robust as possible according to total least squares estimation, the following matrices for $\boldsymbol{S}$ and $\boldsymbol{S}'$ are chosen [152]. $\boldsymbol{S}$ provides an anisotropic normalization of $\boldsymbol{r}$

$$\boldsymbol{S} = \mathrm{chol}(\boldsymbol{M})^{-1} \quad \text{with} \quad \boldsymbol{M} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}_i\boldsymbol{r}_i^{T} \qquad (3.13)$$

and the Cholesky decomposition $\mathrm{chol}(\boldsymbol{M}) = \boldsymbol{K}$, whereas $\boldsymbol{K}$ is an upper triangular matrix with $\boldsymbol{M} = \boldsymbol{K}\boldsymbol{K}^{T}$. N denotes the number of available point correspondences. $\boldsymbol{S}'$ describes a shift of the origin and an isotropic scaling for $\boldsymbol{r}'$

$$\boldsymbol{S}' = \begin{pmatrix} s & 0 & -s\boldsymbol{c}_1\\ 0 & s & -s\boldsymbol{c}_2\\ 0 & 0 & 1 \end{pmatrix} \quad \text{with} \quad \boldsymbol{c} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}'_{i;1:2} \quad \text{and} \quad s = \frac{\sqrt{2}}{\frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{r}'_{i;1:2} - \boldsymbol{c}\right\|}\ , \qquad (3.14)$$

thus, $\boldsymbol{c}$ denotes the centroid and s an isotropic scaling factor averaging the length of each point to the length of $\begin{pmatrix}1 & 1 & 1\end{pmatrix}^{T}$.

As next, we enforce the singularity constraint of the Fundamental Matrix, replacing the smallest singular value by zero. Furthermore, the non-zero singular values of the

Essential matrix have to be one, thus, the modifications would look like

$$\bar{\boldsymbol{F}} = \boldsymbol{U_F}\ \mathrm{diag}(\sigma_{F;1}, \sigma_{F;2}, 0)\ \boldsymbol{V_F^T} \quad \text{with} \quad \mathrm{SVD}(\boldsymbol{F}) = \boldsymbol{U_F}\ \mathrm{diag}(\sigma_{F;1}, \sigma_{F;2}, \sigma_{F;3})\ \boldsymbol{V_F^T} \tag{3.15}$$

$$\bar{\boldsymbol{E}} = \boldsymbol{U_E}\boldsymbol{\Sigma}_{\bar{\boldsymbol{E}}}\boldsymbol{V_E^T} = \boldsymbol{U_E}\ \mathrm{diag}(1, 1, 0)\ \boldsymbol{V_E^T} \quad \text{with} \quad \mathrm{SVD}(\boldsymbol{E}) = \boldsymbol{U_E}\boldsymbol{\Sigma_E}\boldsymbol{V_E^T} \tag{3.16}$$

where $\mathrm{diag}(d_1, \ldots, d_n)$ denotes a diagonal matrix with the entries $d_1, \ldots, d_n$. This modification allows to neglect the aforementioned factor $\sqrt{2}$ in Eq. 3.8.

In [152], a further concept for robustification is proposed. The SVD based solution of an overdetermined linear system in Eq. 3.7 minimizes the error for each entry of the matrix $\boldsymbol{A}$. In Section 6.1, we will derive the first order perturbation of the matrix $\boldsymbol{A}^T$. To ease understanding at that point, we will solve for the transposed Essential matrix which leads to a permuted matrix $\boldsymbol{\Delta}^T_{\mathring{\boldsymbol{A}}}$ of $\boldsymbol{\Delta}^T_{\boldsymbol{A}}$. This matrix looks like

$$\begin{aligned}\boldsymbol{\Delta}^T_{\mathring{\boldsymbol{A}}} &= \underset{\text{row } (1,4,7,2,5,8,3,6,9)}{\text{perm}} \left(\boldsymbol{\Delta}^T_{\boldsymbol{A}}\right) \\ &\cong \begin{pmatrix} \delta_{u_1}u'_1 + \delta_{u'_1}u_1 & \delta_{u_2}u'_2 + \delta_{u'_2}u_2 & \ldots & \delta_{u_N}u'_N + \delta_{u'_N}u_N \\ \delta_{v_1}u'_1 + \delta_{u'_1}v_1 & \delta_{v_2}u'_2 + \delta_{u'_2}v_2 & \ldots & \delta_{v_N}u'_N + \delta_{u'_N}v_N \\ \delta_{u'_1} & \delta_{u'_2} & \ldots & \delta_{u'_N} \\ \delta_{u_1}v'_1 + \delta_{v'_1}u_1 & \delta_{u_2}v'_2 + \delta_{v'_2}u_2 & \ldots & \delta_{u_N}v'_N + \delta_{v'_N}u_N \\ \delta_{v_1}v'_1 + \delta_{v'_1}v_1 & \delta_{v_2}v'_2 + \delta_{v'_2}v_2 & \ldots & \delta_{v_N}v'_N + \delta_{v'_N}v_N \\ \delta_{v'_1} & \delta_{v'_2} & \ldots & \delta_{v'_N} \\ \delta_{u_1} & \delta_{u_2} & \ldots & \delta_{u_N} \\ \delta_{v_1} & \delta_{v_2} & \ldots & \delta_{v_N} \\ 0 & 0 & \ldots & 0 \end{pmatrix}\end{aligned} \tag{3.17}$$

and for the case where no error in the first image can be assumed, the perturbation matrix looks like

$$\boldsymbol{\Delta}^T_{\mathring{\boldsymbol{A}}} \cong \begin{pmatrix} \delta_{u'_1}u_1 & \delta_{u'_2}u_2 & \ldots & \delta_{u'_N}u_N \\ \delta_{u'_1}v_1 & \delta_{u'_2}v_2 & \ldots & \delta_{u'_N}v_N \\ \delta_{u'_1} & \delta_{u'_2} & \ldots & \delta_{u'_N} \\ \delta_{v'_1}u_1 & \delta_{v'_2}u_2 & \ldots & \delta_{v'_N}u_N \\ \delta_{v'_1}v_1 & \delta_{v'_2}v_2 & \ldots & \delta_{v'_N}v_N \\ \delta_{v'_1} & \delta_{v'_2} & \ldots & \delta_{v'_N} \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \end{pmatrix}. \tag{3.18}$$

This is, *e.g.*, the case using KLT (see Section 7.1), which looks for matches of a specific patch and, thus, is error-free by definition.

The conventional SVD does not consider, that the last or last three rows of the error

matrix are zero, respectively. A generalization of the Eckart-Young-Mirsky approximation theorem, first mentioned in [153], allows to account for it and to find the smallest perturbation of a submatrix which lowers the rank of a matrix [154]. Let $\boldsymbol{C} \in \mathbb{R}^{N\times(9-z)}$ and $\boldsymbol{D} \in \mathbb{R}^{N\times z}$ denote the error-prone and error-free submatrix of $\mathring{\boldsymbol{A}}$ respectively, with z being the number of error-free columns in $\mathring{\boldsymbol{A}}$

$$\mathring{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} . \tag{3.19}$$

Then, by row compression, we yield

$$\boldsymbol{U}_{\boldsymbol{D}}^T \mathring{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{C_1} & \boldsymbol{D_1} \\ \boldsymbol{C_2} & 0 \end{pmatrix} \quad \text{with} \quad \boldsymbol{U_D \Sigma_D V_D^T} = \text{SVD}(\boldsymbol{D}) \tag{3.20}$$

which allows us to estimate the matrix $\boldsymbol{\Delta_C}$, the smallest perturbation which lowers the rank of matrix $\mathring{\boldsymbol{A}}$ to $\text{rk}\left(\tilde{\boldsymbol{A}}\right) = 8$ and

$$\tilde{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{C} + \boldsymbol{\Delta_C} & \boldsymbol{D} \end{pmatrix} , \tag{3.21}$$

whereas

$$\begin{aligned} \boldsymbol{\Delta_C} = \boldsymbol{U_D} \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\Delta}_{\boldsymbol{C_2}} \end{pmatrix} \quad \text{with} \quad \boldsymbol{\Delta}_{\boldsymbol{C_2}} &= -\boldsymbol{U}_{\boldsymbol{C_2}} \operatorname{diag}\left(0, \ldots, 0, \sigma_{8-\text{rk}(\boldsymbol{D})+1}, \ldots, \sigma_{9-z}\right) \boldsymbol{V}_{\boldsymbol{C_2}}^T \\ &\text{and } \boldsymbol{U}_{\boldsymbol{C_2}} \operatorname{diag}(\sigma_1, \ldots, \sigma_{9-z}) \, \boldsymbol{V}_{\boldsymbol{C_2}}^T = \text{SVD}(\boldsymbol{C_2}) , \end{aligned} \tag{3.22}$$

assuming that $N \geq 9$. In case the features are tracked throughout an image sequence, the initial features are error-free and, thus, $z = 3$. While in case of tracking-by-matching, as described in Section 7.2.1, the number of error-free columns becomes reduced to the last one and, hence, $z = 1$.

Decomposition of the Essential Matrix

Once the Essential matrix has been computed, it can be decomposed into translation and rotation. By modifying the polar decomposition of $\bar{\boldsymbol{E}}$ we yield

$$\begin{aligned} \bar{\boldsymbol{E}} = \boldsymbol{U_E \Sigma_{\bar{E}} V_E^T} &= \boldsymbol{U_E \Sigma_{\bar{E}}} \underbrace{\boldsymbol{W Z U_E^T}}_{[\boldsymbol{t}]_\times} \quad \underbrace{\boldsymbol{U_E Z W V_E^T}}_{\boldsymbol{R}} \\ &= [\boldsymbol{t}]_\times \boldsymbol{R} \end{aligned} \tag{3.23}$$

with

$$\boldsymbol{W} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det\left(\boldsymbol{U_E V_E^T}\right) \end{pmatrix} , \quad \tilde{\boldsymbol{Z}} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.24}$$

and

$$\boldsymbol{Z} = \begin{cases} \tilde{\boldsymbol{Z}}^T & \text{if} \quad \left(\sum_i \boldsymbol{A}_i^+ \boldsymbol{t}\right)^T \mathbf{1}_2 < 0 \quad \text{with} \quad \boldsymbol{A}_i = \begin{pmatrix} -\boldsymbol{R}\boldsymbol{r}_i & \boldsymbol{r}_i' \end{pmatrix} \\ \tilde{\boldsymbol{Z}} & \text{else} \end{cases} . \tag{3.25}$$

Furthermore, $\boldsymbol{A}_i^+$ denotes the Moore-Pennrose pseudoinverse of $\boldsymbol{A}_i$. This case distinction disambiguates whether the points lie in front or behind the image plane. Again, only one correspondence has to be checked in the noise free case, but otherwise, several feature correspondences should be tested to provide a reliable result. Assuming only a small rotation, the following more efficient test for $\boldsymbol{Z}$ can be used, where only the first two diagonal elements are checked to be positive:

$$\boldsymbol{Z} = \begin{cases} \tilde{\boldsymbol{Z}}^T & \text{if} \quad \boldsymbol{R}_{1,1} + \boldsymbol{R}_{2,2} < 0 \\ \tilde{\boldsymbol{Z}} & \text{else} . \end{cases} \tag{3.26}$$

The translation vector can be easily found as the unit vector minimizing

$$\tilde{\boldsymbol{t}} = \arg\min_{\boldsymbol{t}} \left(\left\| \bar{\boldsymbol{E}}^T \boldsymbol{t} \right\| \right) , \tag{3.27}$$

which corresponds to finding the unit eigenvector of $\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T$ associated with the smallest eigenvalue

$$\tilde{\boldsymbol{t}} = \boldsymbol{U}_{\boldsymbol{E};-,3} . \tag{3.28}$$

It remains to solve for the ambiguity of sign:

$$\boldsymbol{t} = \begin{cases} -\tilde{\boldsymbol{t}} & \text{if } \sum_i \left(\boldsymbol{R}\,\boldsymbol{r}_i \times \boldsymbol{r}_i'\right)^T \bar{\boldsymbol{E}}\boldsymbol{r}_i < 0 \\ \tilde{\boldsymbol{t}} & \text{else} \end{cases} \tag{3.29}$$

Actually, only one correspondence has to be checked in the noise free case, but with noise, far distant features, which are translation invariant, would cause this test to be more or less random. Thus, in practice one needs to use a few correspondences to ensure a reliable result or just one close feature which has a large measurable translational component.

3.3.1.1 Implementation Variants

The aforementioned variation concepts are evaluated in four different implementation variants, which are described in the following and named according to their inventors:

- **original eight-point (Longuet-Higgins variant)**: this implementation consists of the unmodified image rays of Eq. 3.10 and, thus, uses the $\boldsymbol{A}$-matrix as

described in Eq. 3.5.

- **Hartley variant**: this variant uses isotropic scaling for both sets of image rays, $\boldsymbol{S}$ and $\boldsymbol{S}'$, as denoted in Eq. 3.14.
- **Mühlich variant**: this variant uses anisotropic scaling for the first image rays and isotropic scaling for the rays of the second image as introduced in Eq. 3.14 and 3.13.
- **Mühlich TLS-FC variant**: this variant uses the same normalization as the one before, but implements also the fixed-column perturbation estimation as seen in Eq. 3.21.

3.3.2 Iterative Solutions

Other approaches exist, which solve for the rotation and translation in an iterative way [143, 155]. In the following, we will sketch one such iterative method, *Visual-GPS* (VGPS), more in detail, because we will describe an enhanced version of it in Section 5.2.1.

VGPS estimates the camera motion from a set of tracked landmarks, knowing the distance of the features up to scale in the reference frame [155, 156]. The method assumes a reference image $\mathcal{C}_0$ and, thereby, solves the relative orientation problem of determining the pose of the camera ${}^{C_0}\boldsymbol{T}_t$ at time-instance t with respect to $\mathcal{C}_0$. For that it requires an internal 3D model (a set of N points, $\boldsymbol{P}_i$, $i \in \{1..N\}$, in the scene) attached to $\mathcal{C}_0$. This model can be constructed in an arbitrary way (e.g. stereo triangulation, see Section 7.1.1). The exterior orientation between the current frame and the reference 3D model is computed as follows: an additional tentative 3D model $\widehat{\boldsymbol{P}}'_i$ is generated from the 2D projections in the images $\boldsymbol{r}'_i$ by using approximated ranges only. These ranges are estimated from the preceding pose estimation in $t-1$. Thus, the problem is reduced to finding the relative orientation between these two sets of points $\boldsymbol{P}_i$ and $\widehat{\boldsymbol{P}}'_i$. This can be solved in closed form using the singular value decomposition.

In a nutshell: Relative translation and rotation are estimated separately. The origins of the sets of points are set to their respective centroids without modifying their orientations

$$\begin{aligned} \boldsymbol{C} &= \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{P}_i & \boldsymbol{C}' &= \frac{1}{N}\sum_{i=1}^{N}\widehat{\boldsymbol{P}}'_i \\ \boldsymbol{P}_i^* &= \boldsymbol{P}_i - \boldsymbol{C} & \widehat{\boldsymbol{P}}_i'^* &= \widehat{\boldsymbol{P}}'_i - \boldsymbol{C}' \,. \end{aligned} \tag{3.30}$$

The camera rotation between these sets of points of the same model corresponds to the relative rotation between the camera and the reference frames and can be calculated by maximizing the trace of the inertia matrix of the matched set:

$$ {}^{C_0}\boldsymbol{R} = \arg\max_{\boldsymbol{R}} \left(\text{trace}\left(\boldsymbol{R}^T \boldsymbol{M}\right)\right) \quad \text{with} \quad \boldsymbol{M} = \sum_{i=1}^{N} \widehat{\boldsymbol{P}}_i'^{*} \, \boldsymbol{P}_i^{*} \, . \tag{3.31} $$

The solution to this equation can be found in [157] by

$$ {}^{C_0}\boldsymbol{R} = \boldsymbol{V}_{\boldsymbol{M}} \boldsymbol{U}_{\boldsymbol{M}}^T \quad \text{with} \quad \text{SVD}(\boldsymbol{M}) = \boldsymbol{U}_{\boldsymbol{M}} \boldsymbol{\Sigma}_{\boldsymbol{M}} \boldsymbol{V}_{\boldsymbol{M}}^T \, . \tag{3.32} $$

The camera translation can also be computed in closed-form as the distance between the rotation compensated centroids of the two 3D point sets:

$$ {}^{C_0}\boldsymbol{t} = \boldsymbol{C} - \frac{1}{n} \, {}^{C_0}\boldsymbol{R} \sum_{i=1}^{n} \boldsymbol{P}_i' \, . \tag{3.33} $$

Since the tentative 3D model $\widehat{P}_i'$ may differ from the reference one, the final solution is found iteratively by optimizing the unknown ranges of the tentative model at the same time. The algorithm terminates whenever sufficient consistency between the reference and the original set of points is achieved. The convergence rate depends on the measurement errors, the location of the features in 3D space and the quality of the initial parameters.

3.3.3 Least Squares and Maximum Likelihood Solutions

Optimization based solutions are in general computationally complex and suffer from limited workspaces. However, these kind of approaches become more and more popular due to the increasing computational power. However, we do not aim to solve a task with the most powerful processing platforms available, but we want to estimate the motion reliably with as less resources as possible to allow for other parallel processes and to be also feasible for smaller, low-current, and resource-limited platforms. Nevertheless, the approaches we are going to mention in this section are real-time state-of-the-art solutions, which solve the task of motion and scene structure estimation in a highly robust way.

MonoSLAM uses an extended Kalman filter (EKF) to predict and fuse the measurements [141]. While this approach works fine for a small number of features, it does not scale well. The filter state vector and the covariance matrices are not real-time feasible anymore as soon as too many features are used.

In contrast, PTAM is a keyframe based approach, which looks for feature correspondences between the current and the keyframe image [135]. As soon as the overlap is below a certain threshold, a new keyframe is acquired. The relative locations of neighboring keyframes are estimated in an optimization framework based on the common features. Thus, two processes, real-time feature tracking and online[1] keyframe mapping, run in parallel and asynchronous. In a later work, this algorithm has been extended to support also so called *edgelets* (pieces of lines) [158]. In this way, it became more robust against motion blur. Furthermore, the huge map sizes limit the algorithm to small rooms. To overcome this, the maps can be swapped from the physical memory to the hard drive to allow for larger environments [159].

The question, whether filter or batch-optimization based approaches should be preferred, has been answered in [160]. The outcome of the experiments shows, that keyframe optimization yields higher accuracy per processing unit than filter based approaches. Only if small processing budges are available, filter should be preferred to bundle adjustment methods.

Most recently a bundle adjustment based motion estimation and scene reconstruction has been proposed which processes the dense optical flow of the full image [161]. This became feasible by implementing the processing expensive parts of the optimization framework on a GPU. No tracking is required due do the use of every pixel in the image.

3.3.4 Active Matching

Thinking of efficient feature tracking and pose estimation the concept of *active matching* (AM) cannot be neglected [162, 163]. The idea is to not search for all feature correspondences before estimating the motion, but track feature per feature and use the gained information to estimate the possible motion, which reduces the search space for the remaining ones. By sequentially choosing the features with the highest innovation gain, the search space can be narrowed down drastically by each iteration reducing the tracking time significantly. However, the question remains, whether this speed up compensates for the overhead of computing the search spaces and the innovation gains for the remaining features in each step. Nevertheless, the concept is highly interesting and we will make use of it in an application presented in Section 9.1.

[1] With the word *online* we denote a process, which runs not necessarily in real-time, but processes the data at run-time.

3.4 Fusion of IMU and Camera Measurements

Each sensor has its drawbacks and weaknesses. As motivated in Section 1.1, one sensor is not reliable enough for a general application. For instance, a camera facing a featureless wall cannot provide navigation information, a laser does not detect glass, an IMU drifts and suffers from biases, sonar is not accurate due to echoes, GPS and magnetometers do not work indoor, and so forth. Hence, only by fusing various sensors, we will achieve a robust and reliable pose estimation for a variety of environments and, thus, applications. A requirement for any sensor fusion is, that their relative pose is calibrated online or in an offline registration step.

In this work, we focus on the combination of cameras and IMUs. Before we list different fusion techniques for IMU-camera setups, we will first discuss sensor registration methods available in literature with focus on IMU-camera setups.

3.4.1 IMU-Camera Registration

For a proper fusion of the measurements, the spatial alignment of the sensors has to be known. However, most IMU-camera calibration approaches neglect a further important registration issue: the temporal alignment. Especially for highly dynamic motions, where the signal to noise ratio of the IMU becomes practical, an accurate alignment is crucial. If the sensors contain a clock and provide a time-stamp for their samples, one can just synchronize the clocks or the time-stamps to temporally align the measurements. Several solutions exist to synchronize clocks (an overview is given in [164]) or the time-stamps (a survey is given in [165]). Unfortunately, cameras and IMUs provide in general just the sensor data and sometimes also a sample counter, but the first real time-stamp is usually only set by the driver on the host computer. Hence, a different synchronization concept must be applied.

Temporal Alignment

Various approaches exist which describe how to fuse measurements which suffer from time delays in filters. Some of them assume, that the delay is known apriori [166, 167, 168]. Li and Leung estimate the time delay in an unscented Kalman filter [169]. Julier and Uhlmann show how measurements with random and unknown delays can be fused using the covariance union algorithm [170]. Tungadi and Kleeman estimate the time delay between a laser-range sensor and the odometry measurements of a mobile robot by computing the phase shift of a periodic motion [171]. They do not consider any

error introduced by slippage or outlier of the range samples. Furthermore, for their method the spatial alignment of the measurements is required.

Recently, Kelly and Sukhatme presented an approach to estimate the time delay between a proprioceptive and an exteroceptive sensor [172]. The so called Time Delay Iterative Closest Point (TD-ICP) algorithm tries to minimize the distance between the orientation trajectory in 3D space by estimating the angular and temporal alignment between the sensors. Estimating the angular trajectory by the IMU requires an error-prone strap-down computation and also the camera orientation would suffer from drift if no global landmarks, *e.g.*, a checkerboard, are provided. Moreover, in this work, any jitter of the time-stamps is neglected.

The time delay could also be estimated within an optimization framework, as, *e.g.*, the one we will use for the spatial alignment and which is introduced in Section 4.2. For that the batch-based optimization algorithm has to be enhanced by a further optimization parameter which denotes the time delay. However, the computational complexity would increase significantly because a temporal dependency prevents several assumptions which speed up computation. In the case at hand, the B-spline matrices representing the trajectories would not be constant anymore and the convergence capability would probably be reduced due to an additional degree of freedom.

Spatial Alignment

Two sensors can be spatially aligned by relating their relative measurements of at least two motions. Such a relation of a motion in two different coordinate systems is a well-known problem in robotics. It appears every time a flange mounted sensor has to be calibrated relative to the robot's wrist. This problem is known as "$\boldsymbol{AX} = \boldsymbol{XB}$" problem, where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the known motions and $\boldsymbol{X}$ the unknown transformation between the respective coordinate systems. It has first been motivated in the 1980's – today, several closed form solutions exist [173, 174, 106, 175, 176]. Especially the registration of a camera relative to a robot, the so called hand-eye calibration, has been intensively discussed [177, 178, 179, 180].

The problem in IMU-camera registration[1] is a little bit more tricky. A camera can estimate the proper motion, as long as the scale of some objects in the scene is provided. IMUs, in contrast, do not directly measure the motion, but only the first derivative of the angular motion, the angular velocity, and the second derivative of the translational motion, the acceleration. Furthermore, its measurements are rather noisy and, hence, high dynamics are necessary to provide an appropriate signal to noise ratio (S/N) for

[1] In the style of "hand-eye calibration" this problem would be called "vestibule-eye calibration".

an accurate registration. Both, the gyroscopes and the accelerometers, suffer from a bias, which can only be estimated, if properly fused with another sensor. Finally, the accelerometer measures also the Earth's gravitation, which makes it generally impossible to process its measurements without knowing the current orientation relative to the Earth's center. Hence, without any assumptions, there is no other way than to estimate the trajectories of the IMU and, thus, to fuse IMU and camera already at the registration stage. However, by special setups and measurement arrangements the registration problem can be reduced to allow for a direct estimation of the rotation and translation.

Several inertial-visual calibration techniques came up in the last years. They can be categorized in three different classes: closed-form solutions by reducing the system's complexity, Kalman filter based approaches and methods which make use of optimization techniques.

Lobo and Dias make use of the gravitation vector, measured by the accelerometers, and a vertical calibration pattern to estimate the rotational alignment of the inertial-visual system [181, 182]. The translation between the devices is then estimated by using a turntable. The accelerometers have to lie in the center of the turntable, so that they become the center of rotation. In this way, simplified equations known from hand-eye calibration can be used to solve for the translation. While this approach has the advantage to be static and does not need dynamic motions, the necessary system setup is rather cumbersome and error-prone.

The most popular solution is to use an extended or unscented Kalman Filter (EKF, UKF) to estimate the sensor alignment. Approaches of this category implement the following states in their filter: position, orientation, linear and angular velocity, bias of the gyroscopes, bias of the accelerometer and, finally, translation and rotation between IMU and camera.

Kelly and Sukhatme present an UKF based calibration algorithm which can be based on artificial or natural landmarks [183, 184]. The idea is to allow a robot to simultaneously explore the environment while calibrating its sensors. If the calibration step is done offline, the computational overhead of the UKF becomes irrelevant, while it provides a higher order approximation than the EKF and should, thus, be preferred.

Mirzaei and Roumelioutis register the camera and IMU by fusing the corner locations of a checkerboard in the camera images with the measurements of the gyroscopes and the accelerometers [185]. For that they use an error-state (indirect) Kalman Filter. An initialization stage to find good start values precedes the filtering. Moreover, they also analyse the observability of the Kalman Filter, with the result that only

rotations in two degrees of freedom are necessary to estimate the IMU camera transformation [186]. As benchmark for their experiments, they use an optimization framework where they minimize the error between measurements and state estimates as described in [187]. The states and the covariance matrices are linearly propagated as within an EKF, which is similar to the approach described next.

Hol et al. estimate the spatial alignment using a common system identification technique [188]. The innovation in an EKF is minimized by standard gradient descent methods. Their experiments are promising, but one drawback is that the EKF linearizes the motion model which yields significant errors, especially with large rotations.

Lang and Pinz use a constrained nonlinear optimization method to calculate the angular misplacement between gyroscopes and a camera [189]. For that they compare the angles between two camera frames with the integrated gyroscope samples.

3.4.2 Fusion Concepts

There are several approaches known in literature how inertial and visual information can be fused. These approaches can be split into two groups. The first are batch-optimization approaches, where the inertial information and the image location of visual landmarks are fused by bundle-adjustment methods [190]. We will make use of such techniques in our novel IMU to camera registration approach, described in Section 4.2. However, these methods are only feasible for offline processing and, thus, are not discussed further.

Almost all real-time solutions known in literature rely on Kalman filter concepts. Some solutions make use of the more accurate but also more computationally complex UKF [184, 191, 192], while the more familiar, because more efficient, framework is the EKF [193, 194, 195]. Depending on the application also other derivatives might be of advantage [196]. There are two different concepts how the visual information can be fused with the inertial measurements, which let us distinguish between loosely or tightly coupled systems. Tightly coupled systems model the cross-relations of all the observed landmarks in the video stream, resulting in a SLAM framework, together with the inertial measurements [193, 197]. For that they add the landmark location to the state vector which limits the workspace of these approaches due to a quickly increasing complexity of $\mathcal{O}(n^2)$, with n being the number of landmarks. Loosely coupled systems, on the other hand, fuse the inertial measurements with the visual odometry computed from the feature correspondences. Some of them only use the information of the gravity vector [198]. In any case, the state vector remains constant and does not grow with the number of features, which results in a constant complexity of $\mathcal{O}(n)$.

An interesting approach in between loosely and tightly coupled techniques is the one of Mourikis and Roumeliotis which, similar to the aforementioned Klein's PTAM, add the camera viewpoints of selected keyframes to the state vector [194]. Due to an intelligent formulation of the filter problem they achieve high accuracy while still being real-time capable.

However, for an online sensor fusion on resource-limited platforms a loosely measurement fusion seems to be the only alternative. Independent of the fusion framework and the measurement and motion model formulation one has to tackle two major issues if designing such a filter. One is to estimate the proper scale from the acceleration measurements of the IMU and the other is to model the measurement uncertainties properly. Nützi et al. implemented a window based spline fitting for scale estimation [199]. They model the scale parameter already in the motion model which yields redundancies in the state vector due to the representation of the scale as the norm of the position and the extra scale variable. Further, the spline fitting is not real-time capable. Weiss and Siegwart modeled the scale as measurement variable [195]. They use PTAM for motion estimation which allows them to estimate the absolute poses, while it delimits the workspace. They model the visual motion sensor as black box and, hence, they do not take care of a proper model for the measurement uncertainties in the camera domain, where they actually occur.

We are interested in long distance navigation and do not want to restrict the robots workspace. We aim for fast motions in an arbitrary large application area and accept small drifts while moving, as it is the case for any odometry based approaches. Hence, we have relative visual motion measurements. Such information can be modeled by so called stochastic cloning as introduced by Roumeliotis in [200]. With a camera we can measure the direction of translation and we can try to keep the relative translational scale.

In [201], an overview of different measurement models is given. According to that, there are four ways to set up a filter. One can use the sensor coordinate frame to model also the motion which results in pseudo-accelerations and highly nonlinear transitions matrices. Another way is to transform the measurements in the Cartesian frame which results in tricky uncertainty correlations [202]. The most familiar approach is to use a hybrid approach mixing the Cartesian and measurement coordinate frames, *e.g.* using pseudo-measurements [203]. The last and most rare solution in literature is to use a completely different coordinate frame which depends on the application.

Chapter 4

Fusion of IMU and Camera

As motivated in Section 1.1, multimodal perceptions should be combined to compensate for the drawbacks of single sensors and, thus, achieve a reliable estimation. The same is valid for cameras and IMUs. The former depend on the environment, requiring textured scenes and proper lightening conditions, while the latter suffer from long-term drifting. These characteristics can be overcome by combining the sensors. Cameras provide drift-free references and IMUs are fail-safe sensors, providing measurements at equidistant time instances. Hence, IMUs and cameras are rather complementary sensors which gratefully enrich each other if properly fused. However, this characteristic makes the registration of these sensors problematic and complex. Before we describe the fusion of the measurements, we will focus on the temporal and spatial registration, which is crucial for an accurate fusion of any sensor pair.

Estimating the temporal and spatial alignment simultaneously allows the delay estimate to compensate for the spatial alignment error. It seems reasonable to avoid such a compensation and assume, that the temporal and the spatial alignment are not correlated. By computing the time delay in a pre-processing step of the spatial registration, as shown in Section 4.1, rejects such a correlation and prevents any influence of the time lag estimation by the spatial alignment error.

In the following the letters C, G, I, and W denote quantities relative to the camera ($\mathcal{C}$), gyroscopes ($\mathcal{G}$), IMU ($\mathcal{I}$ – in general $\mathcal{I} = \mathcal{G}$, but $\mathcal{G}$ is more specific) and the global (world) reference frame ($\mathcal{W}$) respectively, or measurements of a specific sensor.

4.1 Temporal Alignment

Any calibration step starts with the data acquisition. While the calibration approaches for most sensors are based on static acquisition, gyroscopes need some dynamics to

provide a reasonable signal to noise ratio. Hence, the temporal alignment of the sensors becomes crucial. Most sensors do not provide any time-synchronization mechanism (a prototype for a hardware-synchronized IMU-camera setup is presented in Appendix A.5), nor do they provide a time-stamp in their data packages. Thus, the only hint to "guess" the measurement time is the time-stamp from the driver on the host PC when the sample is received. However, this time-stamp suffers from

- **delay**, due to the acquisition time, buffers on the sensors or/and on the host interface and due to the bus used for transmission,
- **jitter**, the driver on the host side has to be scheduled in order to set the time-stamp,
- **data jams**, the processor is busy with tasks of higher priority and cannot process the arriving samples - these are buffered and all get almost the same time-stamp as soon as the driver becomes active again.

Thus, the problem arises how to interpret these time-stamps and whether it is reasonable at all to make use of them. In three steps we correct the time-stamp sequences of both sensors and align them: first, the sampling period has to be estimated, then missing samples and data jams have to be detected and fixed so that, finally, the sample sequences can be aligned.

One assumption can be made which holds for most sensors, namely, that the sensor itself acquires the measurements with a constant frequency. Calibration sequences are usually rather short, thus, temperature dependent variations can be neglected at this point. Hence, the first thing to do is to estimate the sampling period. For that we compute the median $med_{\Delta t}$ of all differences between two consecutive time-stamps. This rough, but robust, estimate of the sample period is then used to detect all valid sample differences, where valid is defined as a time interval Δ_i between the time-stamps t_{i-1} and t_i which varies only up to half the sampling period. Thus, we define the indicator function $v(\Delta_i)$ which detects valid samples as

$$v(\Delta_i) = \begin{cases} 1, & \text{if } \frac{1}{2} med_{\Delta t} < \Delta_i < \frac{3}{2} med_{\Delta t} \\ 0, & \text{else} \end{cases} . \tag{4.1}$$

Doing so, excessively long and short acquisition times resulting from missing samples and data jams respectively are rejected while allowing for up to 50 % jitter. The acquisition period $\bar{m}_T$ can then be computed as the mean of all valid time differences.

Now, that we know the acquisition period, we can remove jitter, look for missing samples and sample jams and, thus, provide equidistant time-stamps. Data jams are

characterized as long gaps between the time-stamps with a sequence of extremely short time differences following. A sample jam can only be recovered if the number of samples after the gap until the first valid sample interval corresponds to the number necessary to fill the gap. In case there are too few samples to fill the gap, all samples from the jam have to be rejected, because it is not possible to figure out which samples have not been buffered. This follows the strategy to reject samples rather than use false measurements. In case the sensor provides a counter, it can be used to detect missing samples and to partially recover data jams even in the absence of some measurements. The residual gaps in the time-stamp sequence represent missing samples and both, the time and measurement sequence, can be filled with values "*NA*" at this points, denoting unavailable samples.

The sequences have now temporally equidistant samples with period $\bar{m}_T$. We can estimate the jitter-free time-stamp of the first sample, $\bar{t}_0$, by computing the mean of all deviations

$$\bar{t}_0 \;=\; \frac{1}{|\mathcal{S}_v|} \sum_{i \in \mathcal{S}_v} \left(t_i \;-\; i\,\bar{m}_T\right) , \tag{4.2}$$

with $\mathcal{S}_v = \{i \,|\, i \in \{1..N-1\} \;:\; v\,(t_i - t_{i-1}) = 1\}$ denoting the set of all valid time-stamps and N being the number of samples. Finally, we can align the sequences.

In the following ${}^{C}\boldsymbol{R}^{C}_{\Delta_i}$ denotes the inter-frame DCM resulting from the image based motion estimation and ${}^{G}\dot{\phi}^{G}_{t}$, ${}^{G}\dot{\chi}^{G}_{t}$ and ${}^{G}\dot{\psi}^{G}_{t}$ represents the angular velocity measured by the gyroscopes at time instance t. While the frames in which the camera and the gyroscopes measure the rotations may be different, the measured absolute angular velocities $\dot{\theta}_t$ are frame independent and, hence, equal up to an unknown measurement error e_G and e_C

$$\dot{\theta}^{G}_{t} + e_G \;=\; \dot{\theta}^{C}_{t} + e_C \; . \tag{4.3}$$

The absolute rotational velocity of the camera at the temporal midpoint between two images, $t_{i-0.5}$, can be computed by

$$\dot{\theta}^{C}_{t_{i-0.5}} = \frac{\theta^{C}_{\Delta_i}}{\bar{m}_{T;C}} \; , \text{ with } \quad t_{i-0.5} = t_i - \frac{\bar{m}_{T;C}}{2} \; . \tag{4.4}$$

In the following, we present two different approaches for the temporal alignment of a camera and an IMU. Both methods are only based on the measurements of the gyros and, thus, could also only be used for a systems which does not provide accelerometers.

4.1.1 Temporal Alignment by Cross-Correlation

One way to find the time delay between the sensors is to solve following problem:

$$\delta t_{G2C} = \arg\max_{\delta t} \left(\sum_{i \in \mathcal{S}_v} \dot{\theta}^G_{t_i + \delta t} \, \dot{\theta}^C_{t_i} \right) \tag{4.5}$$

A brute-force solution to this problem is to use cross-correlation, which has the nice property to be robust against white noise. The conventional cross-correlation allows sequences to be aligned only up to sampling accuracy. To overcome this problem, a higher sampling resolution with sampling interval Δ_t can be used by interpolating the sequences. In the following we assume, without loss of generality, that the time interval of the acquired gyro measurements is longer than the one of the camera. The problem to solve becomes

$$\delta t_{G2C} = \Delta_t \, \arg\max_{k} \left(\{ \mathrm{xcorr}\,(k) \}_{k=-N_x}^{N_x} \right) \tag{4.6}$$

with $N_x = N_C \frac{\bar{m}_{T_C}}{\Delta_t}$ and N_C denoting the total number of valid and invalid camera samples. The cross-correlation function is defined as

$$\mathrm{xcorr}\,(k) = \begin{cases} \frac{1}{N_x} \sum_{i=0}^{K} \dot{\theta}^G_{i\,\Delta_t + k\,\Delta_t} \quad \dot{\theta}^C_{i\,\Delta_t} & \text{if} \quad k \geq 0 \\ \frac{1}{N_x} \sum_{i=0}^{K} \dot{\theta}^G_{i\,\Delta_t} \quad \dot{\theta}^C_{i\,\Delta_t + k\,\Delta_t} & \text{else} \end{cases} , \tag{4.7}$$

whereas $K = \frac{\bar{m}_{T_C}(N_C - 1) - k\,\Delta_t}{\Delta_t}$, which clips the overlapping sequences. The interval for k can also be reduced to speed up processing. The interpolation causes an error which is proportional to the kind of the chosen approximation. Furthermore, if the temporal displacement is too large, cross-correlation may not find the optimal fit anymore because the overlap of the sequences becomes too small.

4.1.2 Temporal Alignment by Phase Congruency

Another way to address this problem is to evaluate the measurement sequence in frequency domain. For that the measurements have to be transformed in the frequency domain $\mathcal{F}(\omega)$. The amplitude and the phases of the signal can be computed by

$$A(\omega) \; = \; |\mathcal{F}(\omega)| \quad , \quad \varphi(\omega) \; = \; \arg\left(\mathcal{F}(\omega)\right) \tag{4.8}$$

The phase shift between the common frequencies reveals the temporal alignment, ${}^G\delta t_C$. Considering the amplitude, high frequencies consist mainly of the measurement noise and the lower frequencies contain the bias of the gyroscopes. Furthermore, frequency

bin 0 has a large spectral leakage because the absolute angles are defined to be only positive. Therefore, we ignore all frequencies before the first minimum in the spectrum. To suppress outliers and noise we introduce the following normalized weighting function which amplifies similar measurements with large amplitude

$$w(\omega) = \left(\sum_{\omega \in \mathcal{F}_v} \frac{1}{w'(\omega)} \right) w'(\omega) \quad \text{with} \tag{4.9}$$

$$\begin{aligned} w'(\omega) &= (1 - \frac{max_{A(\omega)} - min_{A(\omega)}}{max_{A(\omega)}}) \, min_{A(\omega)} \\ &= \frac{min^2_{A(\omega)}}{max_{A(\omega)}}, \\ max_{A(\omega)} &= \max \left({}^{G}A(\omega) \, , \, {}^{C}A(\omega) \right), \\ min_{A(\omega)} &= \min \left({}^{G}A(\omega) \, , \, {}^{C}A(\omega) \right) \end{aligned} \tag{4.10}$$

and $\mathcal{F}_v$ being the set of all valid frequencies. Of course, to prevent ambiguities it is only possible to compute delays up to $\frac{\pi}{\omega}$, which is half the period of the respective frequency. Converting the difference of the sensor specific phases to time we yield following weighted time difference

$$\delta t_{G2C} = \sum_{\omega \in F_v} w(\omega) \, \frac{\left(k_\omega 2\pi + {}^{G}\varphi(\omega) - {}^{C}\varphi(\omega) \right)}{\omega}, \tag{4.11}$$

where k_ω is the factor which brings the respective frequency in the range of the delay. These factors have to be computed in a second iteration. A manually chosen maximum delay is used to pick all the valid frequencies with a period smaller than this threshold. Based on the phases and amplitudes of these frequencies the time delay is computed and the factors k_ω for the rest of the frequencies are estimated. Now all frequencies can be used to refine the estimate. In our experiments in Section 8.5.1, we compare also an alternative where we rely only on the most significant phase shift corresponding to the frequency bin with the maximum weight $\max(w(\omega))$. This method is based on the assumption that less noise is involved in the estimate, even though it should be less robust.

A conventional Fast or Discrete Fourier Transformation (FFT, DFT) does not allow for gaps in the signal. A generalization of the DFT which can also deal with such gaps is presented in [204] and is called Extended Discrete Fourier Transform (EDFT):

$$\mathcal{F}_{\alpha}(\omega) = \sum_{k=0}^{K-1} x(kT)\ \alpha(\omega, kT) \tag{4.12}$$

where, in general, $\alpha(\omega, kT) \neq e^{-j\omega kT}$. It is an iterative algorithm which tries to find a transform basis function which is applicable to a band-limited signal registered in a finite time interval and providing the results as close as possible to the Fourier transform. Based on this transformation we can compute the magnitudes and the phases even for sequences with gaps.

4.1.3 Online Adaption

As already mentioned earlier, changes or drifts of the sensor clocks may be neglected for the registration because of the short time span of the sequence. In case an application has to run for hours or even longer, both sample periods have to be synchronized in order to ensure proper temporal alignment and, thus, accurate data fusion, *e.g.*, in a Kalman filter. This can be done by adapting the assumed sample rate $\bar{m}_{T;I} = \bar{m}_{G;I}$ and $\bar{m}_{T;C}$ online, *e.g.*, by a low-pass filter on the valid sample times. In this way, one should always provide equidistant time-stamps, whereas the period is adapted by the low-pass filter. Furthermore, it is crucial to always read the complete buffer to detect data jams and, thus, to avoid reading deprecated samples instead of current ones.

We compare the two presented approaches in Section 8.5.1 and evaluate the importance of proper temporal alignment.

4.2 Spatial Alignment

Once the time-line is fixed and the offset between the time-stamps of the IMU and the camera measurements is known, one can estimate the spatial alignment between the sensors. We propose to model the problem of determining the relative pose and orientation of camera and IMU as a nonlinear batch-optimization problem. Contrary to the EKF-based methods, we are not propagating the states and covariances throughout a measurement sequence, but we optimize based on the real nonlinear motion by modeling the trajectory of the sensors. Any filter-based estimation is performed sequentially instead of using the whole batch of data as it is the case in our solution. Hence, our registration considers all available information at the same time and does not estimate the constant IMU-camera alignment on a sample-by-sample basis. Within this optimization framework, we seek to determine the trajectory of the unit together with the calibration parameters for some calibration data sequence. Since the problem is of very

high complexity and many parameters and characteristics of the employed sensors are unknown, it is necessary to introduce reasonable assumptions.

In our case, the assumption is that the trajectory associated with the calibration sequence can be modeled by means of twice differentiable smooth parametric curves ${}^{I}\mathbf{p}(t)$ and ${}^{I}\mathbf{r}(t)$ describing position and orientation in the IMU coordinate frame at time t, respectively. The IMU coordinate frame has been chosen for the sake of computational efficiency, as we will see later. Quaternions will be used to represent orientations, thus ${}^{I}\mathbf{r}(t)$ is a four-dimensional curve, while ${}^{I}\mathbf{p}(t)$ is intuitively a three-dimensional one.

The basic idea of our method is as follows: IMU and camera can be seen as sensors that deliver measurements that are a result of the same motion, observed from different coordinate systems. Given accurate measurements as well as accurate calibration data, we would be able to align the measurements based on the different base coordinate frames of the sensors. Conversely, we can evaluate the quality of calibration values by measuring the alignment between measurements. This leads directly to the idea of optimization of calibration parameters by maximizing the alignment.

The measurements delivered by the IMU are the rotational velocity vector $\boldsymbol{\omega}^I \in \mathbb{R}^3$ and the linear acceleration $\boldsymbol{a}^I \in \mathbb{R}^3$. Depending on whether the camera observes unknown or known landmarks, it can provide relative or even absolute measurements. For the sake of generality, we assume relative translational and rotational measurements, which we interpret as linear and rotational velocities, $\boldsymbol{v}^C \in \mathbb{R}^3$ and $\boldsymbol{\omega}^C \in \mathbb{R}^3$, by knowing the sampling period.

4.2.1 Objective Function

Relationships between the measurements of the different sensors can be established as follows [105, 111]:

$$ {}^{I}\hat{\boldsymbol{\omega}}_t \;=\; {}^{I}\boldsymbol{\omega}_t^I - {}^{I}\boldsymbol{e}_{\boldsymbol{\omega}^I,t} \;=\; {}^{I}\boldsymbol{R}_C \left({}^{C}\boldsymbol{\omega}_t^C - {}^{C}\boldsymbol{e}_{\boldsymbol{\omega}^C,t} \right) \tag{4.13} $$

and

$$ {}^{I}\boldsymbol{a}_t^I - {}^{I}\boldsymbol{e}_{\boldsymbol{a}^I,t} + {}^{I}\boldsymbol{R}_W {}^{W}\boldsymbol{g} \;=\; {}^{I}\boldsymbol{R}_C \left({}^{C}\dot{\boldsymbol{v}}_t^C - {}^{C}\dot{\boldsymbol{e}}_{\boldsymbol{v}^C,t} \right) - {}^{I}\hat{\boldsymbol{\omega}}_t \times \left({}^{I}\hat{\boldsymbol{\omega}}_t \times {}^{I}\boldsymbol{t}_{IC} \right) - {}^{I}\dot{\hat{\boldsymbol{\omega}}}_t \times {}^{I}\boldsymbol{t}_{IC} \tag{4.14} $$

In above formulæ, ${}^{I}\hat{\boldsymbol{\omega}}_t$ denotes the real, error-free angular velocity, $\boldsymbol{e}_x$ represents the error of a specific measurement x, and ${}^{W}\boldsymbol{g}$ is the gravitation vector $(0, 0, -9.81)^T$. Measurements of cheap MEMS gyroscopes and accelerometers are not only affected by white noise, but also by a bias, which has to be modeled explicitly. We will denote the bias values for gyroscope and accelerometer by $\boldsymbol{b}_{\boldsymbol{\omega}^I}$ and $\boldsymbol{b}_{\boldsymbol{a}^I}$, respectively. Since the

bias values change extremely slowly over time, we adopt the common assumption that they are constant during the registration sequence.

Unfortunately, a direct comparison as outlined above is not possible, because the sensor measurements occur at different time instances. Hence, we need a model which allows to interpolate the motion at arbitrary points in time. In our optimization approach, we want to minimize the differences between the measurements and, thus, we aim to minimize the error between all samples and a motion model which inherently interpolates these measurements. The measurement errors $\boldsymbol{\delta}_{x;i}$ at time-instance i can be formulated by following equations:

$$\boldsymbol{\delta}_{\boldsymbol{\omega}^I;i} = {}^I\boldsymbol{\omega}_i^I - \boldsymbol{b}_{\boldsymbol{\omega}^I} - {}^I\boldsymbol{\omega}(t_i) \tag{4.15}$$

$$\boldsymbol{\delta}_{\boldsymbol{a}^I;i} = {}^I\boldsymbol{a}_i^I - \boldsymbol{b}_{\boldsymbol{a}^I} + {}^I\mathbf{r}(t_i) \odot \left(\boldsymbol{I}_{vq}^T {}^W\boldsymbol{g}\right) \odot {}^I\bar{\mathbf{r}}(t_i) - {}^I\ddot{\mathbf{p}}\,(t_i) \tag{4.16}$$

$$\boldsymbol{\delta}_{\boldsymbol{\omega}^C;i} = {}^I\boldsymbol{R}_C {}^C\boldsymbol{\omega}_i^C - {}^I\boldsymbol{\omega}(t_i) \tag{4.17}$$

$$\boldsymbol{\delta}_{\boldsymbol{v}^C;i} = {}^I\boldsymbol{R}_C {}^C\boldsymbol{v}_i^C + {}^I\boldsymbol{\omega}(t_i) \times {}^I\boldsymbol{t}_{IC} - {}^I\dot{\mathbf{p}}\,(t_i) \tag{4.18}$$

with

$${}^I\boldsymbol{\omega}(t) = 2\,\boldsymbol{I}_{vq}\,\left({}^I\bar{\mathbf{r}}(t) \odot {}^I\dot{\mathbf{r}}\,(t)\right)\,, \tag{4.19}$$

$$\boldsymbol{I}_{vq} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\,, \tag{4.20}$$

$${}^I\bar{\mathbf{r}}(t) = \mathrm{diag}\,(-1,-1,-1,\ 1)\ {}^I\mathbf{r}(t) \tag{4.21}$$

and t_i representing the measurement time of sample i. The operator $\odot$ denotes the quaternion multiplication. Please note that the equations are simpler if the trajectory is modeled relative to the IMU frame. The only measurement revealing information about the real orientation of the IMU relative to the world coordinate frame is the gravity vector. However, this becomes irrelevant if we also estimate the orientation of the gravity vector relative to the initial pose of the IMU ${}^{I_0}\boldsymbol{g}$. Hence, the trajectory starts at the origin of the IMU coordinate frame, aligned to its axes and can be estimated independently of the world coordinate frame. That way, ${}^W\boldsymbol{g}$ in Eq. 4.16 can be replaced by ${}^{I_0}\boldsymbol{g}$.

The errors are weighted with the pseudo Huber cost function $\mathrm{h}(\boldsymbol{\delta})$ [108], which is differentiable and robust against outliers. It is defined as

$$\mathrm{h}(\boldsymbol{\delta}_{x,i}) = 2b_x^2 \left(\sqrt{\frac{\boldsymbol{\delta}_{x,i}^T \boldsymbol{\delta}_{x,i}}{b_x^2} + 1}\ - 1\right). \tag{4.22}$$

A common choice for b_x is $3\sigma_x$, where σ_x denotes the standard deviation of the noise and x the kind of measurements for $\boldsymbol{\omega}^I$, $\boldsymbol{a}^I$, $\boldsymbol{\omega}^C$ or $\boldsymbol{v}^C$, respectively.

Concatenating the matrices of all the error vectors $\boldsymbol{\Delta}_x = (\boldsymbol{\delta}_{x,1} \ldots \boldsymbol{\delta}_{x,N_x})$, with N_x the respective number of measurements, yields the total error matrix $\boldsymbol{\Delta} = (\boldsymbol{\Delta}_{\boldsymbol{\omega}^I}\ \boldsymbol{\Delta}_{\boldsymbol{a}^I}\ \boldsymbol{\Delta}_{\boldsymbol{\omega}^C}\ \boldsymbol{\Delta}_{\boldsymbol{v}^C})$. Thus, the objective function $\mathrm{g}(\boldsymbol{\Delta})$ can be formulated as

$$\mathrm{g}(\boldsymbol{\Delta}) = \sum_{x \in \{\boldsymbol{\omega}^I, \boldsymbol{a}^I, \boldsymbol{\omega}^C, \boldsymbol{v}^C\}} \left(\frac{1}{\sigma_x^2} \sum_{i=1}^{N_x} \mathrm{h}(\boldsymbol{\delta}_{x,i}) \right). \tag{4.23}$$

4.2.2 B-Spline Based Trajectory Modeling

The model for our trajectory representation needs to be twice differentiable to provide the second derivative of the position for acceleration estimation ${}^I\ddot{\mathbf{p}}(t)$ (see Eq. 4.16). Therefore, we use a cubic B-spline curve in our approach as continuous, twice differentiable trajectory model. Using this curve model certainly constitutes a restriction, since it imposes a number of constraints on the trajectory. Note however, that it is still more general than the commonly encountered assumption of piecewise linearity of motion, which is usually made in filter-based approaches. We are now going to introduce our notation used for B-spline curves. For more detailed information about B-splines, see [205] or [206]. We define a B-spline curve as

$$\mathbf{s}(t) = \sum_{i=1}^{M} \boldsymbol{c}_i\ \mathrm{b}_i^k(t), \tag{4.24}$$

where $\boldsymbol{c}_i \in \mathbb{R}^d$ are the d-dimensional control point values, $\mathrm{b}_i^k(t)$ denotes the i-th basis function of order k and M is the number of knots. The B-spline order k can be chosen arbitrarily as long as it is at least four. This is required because the trajectory curve has to be at least twice continuously differentiable, since we also need to compute the acceleration from the position spline. We use a cubic B-spline that is of order $k = 4$, which is equivalent to a piecewise linear approximation of the acceleration measurements. With $\boldsymbol{c}$, we denote the vector $\left(\boldsymbol{c}_1^T\ \boldsymbol{c}_2^T \ldots \boldsymbol{c}_M^T\right)^T \in \mathbb{R}^{Md}$ of concatenated control point values. Furthermore, we assume that an equidistant knot sequence is used.

It is well-known, that B-splines are linear in parameters, which means that the evaluation of above equation at several parameter locations $\boldsymbol{t} = (t_0\, t_1 \ldots t_n)$ is equivalent to computing the matrix-vector product $\mathbf{B}(\boldsymbol{t})\,\boldsymbol{c}$ for a suitable basis matrix $\mathbf{B}(\boldsymbol{t})$. More

formally, this is expressed as

$$\begin{pmatrix}\mathbf{s}(t_1)^T & \mathbf{s}(t_2)^T & \dots & \mathbf{s}(t_n)^T\end{pmatrix}^T = \mathbf{B}(\boldsymbol{t})\,\boldsymbol{c}\,. \tag{4.25}$$

For d-dimensional control point values, the basis matrix has the shape

$$\mathbf{B}(\boldsymbol{t}) = \begin{pmatrix} \mathrm{b}_1^k(t_1) & \mathrm{b}_2^k(t_1) & \dots & \mathrm{b}_m^k(t_1) \\ \mathrm{b}_1^k(t_2) & \mathrm{b}_2^k(t_2) & \dots & \mathrm{b}_m^k(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{b}_1^k(t_n) & \mathrm{b}_2^k(t_n) & \dots & \mathrm{b}_m^k(t_n) \end{pmatrix} \otimes \boldsymbol{I}_d, \tag{4.26}$$

where $\otimes$ denotes the Kronecker matrix product, and $\boldsymbol{I}_d$ is the $d \times d$ identity matrix. It is obvious, that if the vector of parameter locations $\boldsymbol{t}$ remains constant, so does the matrix $\boldsymbol{B} = \mathbf{B}(\boldsymbol{t})$. The time-stamps of the measurements are constant due to the previous temporal alignment and, hence, the matrix $\boldsymbol{B}$ has to be computed only once. Furthermore, it is well-known that B-spline derivatives are again B-splines, and as such are again linear in parameters. In our optimization process, we are going to evaluate the spline and its derivatives at the respective measurement time stamps. This means that spline derivatives can also be computed by simply evaluating $\boldsymbol{B'c}$ for some appropriate matrix $\boldsymbol{B'}$ representing the basis matrix of the derived spline.

In our implementation we need a B-spline of dimension $d = 7$ and, thus, $\boldsymbol{c}_i \in \mathbb{R}^7$ to model the IMU pose

$${}^I\mathbf{s}(t) = \begin{pmatrix}{}^I\mathbf{p}(t)^T & {}^I\mathbf{r}(t)^T\end{pmatrix}^T\,. \tag{4.27}$$

Note that the quaternions are constrained to be of unit length, as described in Section 4.2.3, which yields the expected six degrees of freedom for rigid body motion.

The control point vector of the B-spline is part of the parameter vector $\boldsymbol{\theta}$ subject to optimization. Furthermore, this vector contains the two IMU bias terms, the initial direction of the gravity vector, the quaternion and the vector describing the translation between the IMU and camera coordinate system. For the sake of generality, we also model the scale factor α of the measured camera velocity, assuming that natural landmarks are used and the scale of translation is unknown and, thus, is also estimated in our optimization:

$$\boldsymbol{\theta} = \begin{pmatrix}(\boldsymbol{c}_1^T \boldsymbol{c}_2^T \dots \boldsymbol{c}_M^T) & \boldsymbol{b}_{\boldsymbol{\omega}^I}^T & \boldsymbol{b}_{\boldsymbol{a}^I}^T & {}^I\boldsymbol{q}_C^T & {}^{I_0}\boldsymbol{g}^T & {}^I\boldsymbol{t}_{IC}^T & \alpha\end{pmatrix}^T\,. \tag{4.28}$$

4.2.3 Constraints and Optimization Details

There are some constraints on a few parameters which have to be satisfied for the optimization. The unit property of the control points of the B-spline and of ${}^{I}\boldsymbol{q}_{C}$ has to be ensured. Furthermore, the gravity vector has to be of length 9.81 and the first control point of the spline is constrained to represent zero rotation and zero translation to avoid dependencies in the optimization parameters. This is because both the direction of the gravity vector and the pose of the IMU are going to be optimized – at the beginning of the trajectory one of these parameters has to be fixed to prevent redundant degrees of freedom. All these requirements can be formulated as equality constraints which have to be zero:

$$\boldsymbol{r}_{eq;1} = \sum_{i=1}^{M} (\|\boldsymbol{c}_i\|) - M \tag{4.29}$$

$$\boldsymbol{r}_{eq;2} = \left\|{}^{I}\boldsymbol{q}_{C}\right\| - 1 \tag{4.30}$$

$$\boldsymbol{r}_{eq;3} = \left\|{}^{I_0}\boldsymbol{g}\right\| - 9.81 \tag{4.31}$$

$$\boldsymbol{r}_{eq;4} = \boldsymbol{c}_1 - \begin{pmatrix} \boldsymbol{t}_0 \\ \boldsymbol{q}_0 \end{pmatrix} \tag{4.32}$$

with $\boldsymbol{t}_0 = \boldsymbol{0}_3$ and $\boldsymbol{q}_0 = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$.

We use sequential quadratic programming (SQP) as optimization algorithm, because it is known to work well for nonlinear optimization problems and it allows to implement equality contraints [207].

For optimization purposes it is generally necessary to compute the gradient of the objective function, as well as an appropriate Hessian approximation. The gradient of the objective function can be computed by applying the chain rule to Eq. 4.23 as

$$\frac{\partial \mathrm{g}(\boldsymbol{\Delta})}{\partial \boldsymbol{\theta}} = \boldsymbol{J} \sum_{x \in \{\boldsymbol{\omega}^I, \boldsymbol{a}^I, \boldsymbol{\omega}^C, \boldsymbol{v}^C\}} \left(\frac{1}{\sigma_x^2} \sum_{i=1}^{N_x} \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}} \right), \tag{4.33}$$

with $\boldsymbol{J}$ being the Jacobian of the error matrix $\boldsymbol{\Delta}$ relative to the parameter vector $\boldsymbol{\theta}$. The derivative of the pseudo Huber cost function is stacked by

$$\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}} = \left(\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{\omega}^I}}^T \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{a}^I}}^T \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{\omega}^C}}^T \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}_{\boldsymbol{v}^C}}^T \right)^T \tag{4.34}$$

with

$$\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\Delta}_x} = \left(\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\delta}_{x,1}}^T \cdots \frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\delta}_{x,N_x}}^T \right)^T \tag{4.35}$$

and

$$\frac{\partial \mathrm{h}(\boldsymbol{\delta}_{x;i})}{\partial \boldsymbol{\delta}_{x,i}} = \frac{-2}{\sqrt{\frac{\boldsymbol{\delta}_{x,i}^T \boldsymbol{\delta}_{x,i}}{b_x^2} + 1}} \boldsymbol{\delta}_{x,i} \, . \tag{4.36}$$

The Jacobian $\boldsymbol{J}$ can be described by following components

$$\boldsymbol{J} = \frac{\partial \boldsymbol{\Delta}}{\partial \boldsymbol{\theta}} = \begin{pmatrix} \boldsymbol{J_c} & \boldsymbol{J_{b_{\omega}{}^I}} & \boldsymbol{J_{b_a{}^I}} & \boldsymbol{J_{I_{q_C}}} & \boldsymbol{J_{I_0 g}} & \boldsymbol{J_{I_{t_{IC}}}} & \boldsymbol{J_\alpha} \end{pmatrix} , \tag{4.37}$$

which can be computed in a straight-forward manner. The main part of the Jacobian consists of the control points $\boldsymbol{J_c} = \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{c};\boldsymbol{\omega}^I}^T & \boldsymbol{J}_{\boldsymbol{c};\boldsymbol{a}^I}^T & \boldsymbol{J}_{\boldsymbol{c};\boldsymbol{\omega}^C}^T & \boldsymbol{J}_{\boldsymbol{c};\boldsymbol{v}^C}^T \end{pmatrix}^T$ where each $\boldsymbol{J}_{\boldsymbol{c};x}$ looks like

$$\boldsymbol{J}_{\boldsymbol{c};x} = \begin{pmatrix} \frac{\partial \boldsymbol{\delta}_{x;1}}{\partial \boldsymbol{c}_1} & \frac{\partial \boldsymbol{\delta}_{x;1}}{\partial \boldsymbol{c}_2} & \cdots & \frac{\partial \boldsymbol{\delta}_{x;1}}{\partial \boldsymbol{c}_M} \\ \frac{\partial \boldsymbol{\delta}_{x;2}}{\partial \boldsymbol{c}_1} & \frac{\partial \boldsymbol{\delta}_{x;2}}{\partial \boldsymbol{c}_2} & \cdots & \frac{\partial \boldsymbol{\delta}_{x;2}}{\partial \boldsymbol{c}_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{\delta}_{x;N_x}}{\partial \boldsymbol{c}_1} & \frac{\partial \boldsymbol{\delta}_{x;N_x}}{\partial \boldsymbol{c}_2} & \cdots & \frac{\partial \boldsymbol{\delta}_{x;N_x}}{\partial \boldsymbol{c}_M} \end{pmatrix} \tag{4.38}$$

and each $\frac{\partial \boldsymbol{\delta}_{x;i}}{\partial \boldsymbol{c}_j}$ has two parts:

$$\frac{\partial \boldsymbol{\delta}_{x;i}}{\partial \boldsymbol{c}_j} = \begin{pmatrix} \frac{\partial \boldsymbol{\delta}_{x;i}}{\partial \boldsymbol{c}_{\boldsymbol{p};j}} & \frac{\partial \boldsymbol{\delta}_{x;i}}{\partial \boldsymbol{c}_{\boldsymbol{r};j}} \end{pmatrix} \tag{4.39}$$

with $\boldsymbol{c}_{\boldsymbol{p};j}$ and $\boldsymbol{c}_{\boldsymbol{r};j}$ the position and rotation component of the control points modeling $\mathbf{p}(t_j)$ and $\mathbf{r}(t_j)$, respectively. For approximating the Hessian of the system, the popular Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [207] is used.

Nonlinear optimization does not guarantee to find the global optimum. It depends on the convexity of the problem and the initial values of the parameters to optimize, whether a local or the global optimum is found. We do not want to relax the problem to become convex which comes with relaxation errors, but we rather try to find good starting points for the parameters which also speeds up the optimization process. The initialization of the optimization parameters is a three-step procedure. First of all, we find the time offset between the IMU and camera measurements as described in Section 4.1 and an initial estimate of the orientation between IMU and camera coordinate frame ${}^I\hat{\boldsymbol{q}}_C$ as shown in the next block, Section 4.2.4. The second step is to determine the number of B-spline control points to be used and good starting points for them. Finally, the gravity vector and the bias terms have to be initialized. Both

will be described in Section 4.2.5.

4.2.4 Initialization of the Spatial Alignment

In the following we present a closed form initialization for the gyroscopes-camera alignment and briefly discuss the translation initialization. The orientation estimation can also be used as only registration step without optimization in case that only a less accurate alignment is required.

Angular Alignment

The angular registration of a camera and a gyroscope can be seen as solving the rotational part of the well-known hand-eye calibration problem. To compute the rotational alignment ${}^{G}\boldsymbol{R}_{C}$ between the gyros and the camera, we compute the relative rotations, ${}^{G}\boldsymbol{R}_{\Delta_i}$ and ${}^{C}\boldsymbol{R}_{\Delta_i}$, between two consecutive images $i-1$ and i. The rotations of the gyroscopes, measured in their coordinate frame $\mathcal{G}$, are simply integrated between two camera time-stamps.

The task is now to find the rotation ${}^{G}\boldsymbol{R}_{C}$ which rotates the measurements from frame $\mathcal{C}$ to frame $\mathcal{G}$ according to the well-known hand-eye calibration equation

$$ {}^{G}\boldsymbol{R}_{\Delta_i} = {}^{G}\boldsymbol{R}_{C}\, {}^{C}\boldsymbol{R}_{\Delta_i}\, {}^{G}\boldsymbol{R}_{C}^{T} \,. \tag{4.40} $$

To solve for the best fitting spatial alignment ${}^{G}\tilde{\boldsymbol{R}}_{C}$ following least-squares problem over all camera measurements N has to be solved:

$$ {}^{G}\tilde{\boldsymbol{R}}_{C} = \arg\max_{\boldsymbol{R}} \sum_{i=1}^{N} \operatorname{tr}\left({}^{G}\boldsymbol{R}_{\Delta_i}\, \boldsymbol{R}\, {}^{C}\boldsymbol{R}_{\Delta_i}^{T}\, \boldsymbol{R}^{T}\right) \quad | \, R \in SO(3) \tag{4.41} $$

A closed form solution for the rotation estimation is described and derived in [106] and can be computed as follows. Let ${}^{C}\hat{\boldsymbol{p}}_{\Delta_i}$, ${}^{G}\hat{\boldsymbol{p}}_{\Delta_i}$ and ${}^{G}\hat{\boldsymbol{p}}_{C}$ denote the real eigenvectors to the eigenvalue 1 of ${}^{C}\boldsymbol{R}_{\Delta_i}$, ${}^{G}\boldsymbol{R}_{\Delta_i}$ and ${}^{G}\boldsymbol{R}_{C}$ respectively. Hence, they represent the rotation axes of these DCMs. Using this representation, the system of linear equations to solve for ${}^{G}\boldsymbol{p}_{C}$, consisting of ${}^{G}\hat{\boldsymbol{p}}_{C}$ and ${}^{G}\theta_{C}$ according to Eq. 2.4, can be set up by all measurement pairs, such that

$$ \left[{}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i}\right]_{\times} {}^{G}\boldsymbol{p}'_{C} = {}^{C}\hat{\boldsymbol{p}}_{\Delta_i} - {}^{G}\hat{\boldsymbol{p}}_{\Delta_i} \,, \tag{4.42} $$

with $[\,\cdot\,]_{\times}$ denoting the skew symmetric matrix of a vector.

The angle-axis form of the rotation ${}^{G}\boldsymbol{R}_{C}$ can then be computed from ${}^{G}\boldsymbol{p}'_{C}$ by

$$ {}^{G}\hat{\boldsymbol{p}}_{C} = \frac{{}^{G}\boldsymbol{p}'_{C}}{\|{}^{G}\boldsymbol{p}'_{C}\|} \quad \text{and} \quad {}^{G}\theta_{C} = 2\tan^{-1}\left(\|{}^{G}\boldsymbol{p}'_{C}\|\right) . \tag{4.43} $$

If all rotations are measured about the same axis or the sensor coordinate frames are rotated by 0° or 180°, this system is singular and there is no unique solution. However, if there are at least two rotations about different axes and these axes are not colinear, a unique solution exists. We detect the colinear-exception by inspecting the singular values σ_1, σ_2 and σ_3 of $\left[{}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i}\right]_{\times}$, where $\sigma_1 \geqslant \sigma_2 \geqslant \sigma_3$. If $\frac{\sigma_3}{\sigma_2} < k \ll 1$, where k denotes a threshold which defines the stack of skew symmetric matrices to be rank-deficient, a rotation close to 0° or 180° between the sensor frames is expected. In this case the angular alignment is estimated using SQP optimization, where the objective function is defined as

$$ \mathrm{c}_{\mathrm{SQP}}\left({}^{G}\boldsymbol{R}_{C}\right) = \sum_{i=1}^{N_C} \left\| {}^{G}\boldsymbol{R}_{C}\, {}^{C}\hat{\boldsymbol{p}}_{\Delta_i} - {}^{G}\hat{\boldsymbol{p}}_{\Delta_i} \right\| . \tag{4.44} $$

As starting point for the estimation we use

$$ {}^{G}\boldsymbol{R}_{C} = \mathrm{DCM}(\tilde{\boldsymbol{p}}_{\Delta_i}, 0°) \quad \text{or} \quad {}^{G}\boldsymbol{R}_{C} = \mathrm{DCM}(\tilde{\boldsymbol{p}}_{\Delta_i}, 180°) , \tag{4.45} $$

depending on whether the signs of the delta rotation axes correspond or not, where

$$ \tilde{\boldsymbol{p}}_{\Delta_i} = \frac{\tilde{\boldsymbol{p}}'_{\Delta_i}}{\left\|\tilde{\boldsymbol{p}}'_{\Delta_i}\right\|} \quad \text{and} \quad \tilde{\boldsymbol{p}}'_{\Delta_i} = \mathrm{med}\left(\left\{ {}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i} \right\}_{i=1}^{N_C}\right) . \tag{4.46} $$

Hence, $\tilde{\boldsymbol{p}}_{\Delta_i}$ denotes the normalized element-wise median of the sum of rotation axes, which should be quite close to the global optimum and, thus, prevent SQP from finding local ones.

An alternative approach would be to modify the measurements of the camera or IMU by a constant rotation, *e.g.*, 90°, so that the closed-form solution is non-singular again and consider this rotational offset in the computed angular alignment. However, in this case it is crucial to choose an axis which does not correspond to the actual axis of the rotational alignment and, therefore, we propose to chose an axis perpendicular to $\tilde{\boldsymbol{p}}_{\Delta_i}$.

In the context of gyro-camera calibration, the closed-form solution needs some adaptions to achieve adequate robustness. In [106], critical factors affecting the accuracy and robustness have been discussed and it has been observed, that the accuracy is

proportional to the magnitude of the measured rotations. In the case of camera to gyroscope calibration, the sensitivity for errors due to relative small inter frame rotations becomes crucial and may lead to wrong results. Furthermore, Eq. 4.42 minimizes the steady measurement error of a conventional robot-camera setup, but it does not consider any time correlation between the sensors. It is necessary to overcome this problem, if we want to apply this method for bias prone gyroscopes.

If Eq. 4.42 is applied without any modification, each inter frame rotation would affect the outcome equally - independent of whether there is no rotation, and the measurement consists only of noise, or in presence of an outlier. Therefore, the absolute angles of the inter frame rotations should be used to weight the respective equation based on following observations: Small rotations suffer from a small signal to noise ratio and, further, a large discrepancy between the sensor measurements implies an erroneous sample pair. This leads to following weights for both sides of Eq. 4.42 and the summands in Eq. 4.44

$$w_i = (1 - \frac{max_{\theta_i} - min_{\theta_i}}{max_{\theta_i}})\, min_{\theta_i} = \frac{min_{\theta_i}^2}{max_{\theta_i}} \tag{4.47}$$

with max_{θ_i} and min_{θ_i} analogously defined to Eq. 4.10.

Calibration sequences are usually short and, therefore, a common assumption is that the IMU biases are constant. The bias estimation is free of constrains and, thus, we use a Levenberg-Marquardt optimization for its computation. In the experiments of Section 8.5.2, we compare the following two cost functions. In the first approach we simply try to find the bias by making the absolute angles as similar as possible, while disregarding outliers. This is achieved by applying the Blake-Zisserman cost function [208], which has been chosen due to its outlier handling:

$$\mathrm{c_{BZ}}\left(\delta\right) = -\ln\left(e^{-\delta^2} + \epsilon\right). \tag{4.48}$$

The crossover point from inliers to outliers is given by the threshold α in $\epsilon = e^{-\alpha^2}$. Thus, this optimization problem may be written as

$$\tilde{\boldsymbol{b}} = \arg\min_{\boldsymbol{b}} \sum_{i=1}^{N} \mathrm{c_{BZ}}\left({}^{G}\theta'_{\Delta_i} - {}^{C}\theta_{\Delta_i}\right) \tag{4.49}$$

with ${}^{G}\theta'_{\Delta_i}$ being the absolute angle corresponding to

$$ {}^{G}\boldsymbol{R}'_{\Delta_i} = \prod_{t=t_{i-1}}^{t_i} {}^{G}\boldsymbol{R}'_t \,, \tag{4.50} $$

where

$$ {}^{G}\boldsymbol{R}'_t = \mathrm{DCM}\left({}^{G}\bar{m}_T \left(\begin{bmatrix} {}^{G}\dot{\phi}_t \\ {}^{G}\dot{\chi}_t \\ {}^{G}\dot{\psi}_t \end{bmatrix} - \boldsymbol{b} \right) \right) \tag{4.51} $$

and $\tilde{\boldsymbol{b}}$ being the estimated bias.

An other approach is to minimize the trace of the covariance matrix, $\boldsymbol{P}_{{}^{G}\boldsymbol{p}_C}$, of ${}^{G}\boldsymbol{p}_C$. The covariance matrix for ${}^{G}\boldsymbol{p}'_C$ results from the over-constrained linear system (see Eq. 4.42) as

$$ \boldsymbol{P}_{{}^{G}\boldsymbol{p}_C} = \boldsymbol{V}_{[\cdot]_\times} \boldsymbol{\Sigma}^{-2}_{[\cdot]_\times} \boldsymbol{V}^T_{[\cdot]_\times} \quad \text{with} \quad \boldsymbol{U}_{[\cdot]_\times} \boldsymbol{\Sigma}_{[\cdot]_\times} \boldsymbol{V}^T_{[\cdot]_\times} = \mathrm{SVD}\left(\boldsymbol{B}^{C+G}_{[\cdot]_\times} \right) \tag{4.52} $$

and $\boldsymbol{B}^{C+G}_{[\cdot]_\times}$ representing the vertical stack of the skew matrices of all the vector sums, $\left[{}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i} \right]_\times$. Depending on the length of the calibration sequence, this matrix may become quite large. To reduce the processing time, one can also use

$$ \boldsymbol{P}_{{}^{G}\boldsymbol{p}'_C} = \boldsymbol{V}_T \left(\mathrm{tr}\left(\boldsymbol{\Sigma}^2_T \right) \boldsymbol{I} - \boldsymbol{\Sigma}^2_T \right)^{\frac{1}{2}} \boldsymbol{V}^T_T \tag{4.53} $$

with $\boldsymbol{U}_T \boldsymbol{\Sigma}_T \boldsymbol{V}^T_T = \mathrm{SVD}\left(\boldsymbol{B}^{C+G}_T \right)$ and $\boldsymbol{B}^{C+G}_T$ being the row-wise stack of the vectors $\left({}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i} \right)^T$. In this way, one has only to compute the SVD of $\boldsymbol{B}^{C+G}_T$ and not of $\boldsymbol{B}^{C+G}_{[\cdot]_\times}$ which is three times larger. A derivation of this equation is shown in Appendix A.4.

Minimizing the covariance matrix of ${}^{G}\boldsymbol{p}'_C$ does not necessarily mean to minimize the covariance of ${}^{G}\boldsymbol{p}_C$. Therefore, we propagate the covariance and the estimate for ${}^{G}\boldsymbol{p}'_C$ through the nonlinear equation

$$ {}^{G}\boldsymbol{p}_C = \frac{2\, {}^{G}\boldsymbol{p}'_C}{\sqrt{1 + \left| {}^{G}\boldsymbol{p}'_C \right|^2}} \tag{4.54} $$

by using sigma-points, like in the Unscented Kalman filter [209]. Sticking to the notation of that paper, the weight $W^{(0)}$ is set to 0 according to the formula which computes

the optimum in case of a Gaussian distribution

$$W^{(0)} = 1 - \frac{N_S}{3} \tag{4.55}$$

with N_S denoting the vector length, which is three in our case. Thus, the objective function is this time

$$\tilde{\boldsymbol{b}} = \arg\min_{\boldsymbol{b}} \left(\operatorname{tr}\left(\boldsymbol{P}_{G_{\boldsymbol{p}_C}}\right) \right) . \tag{4.56}$$

Translational Alignment

An initialization of the translation is rather complicated to achieve. Either one can extract a first estimate for ${}^{I}\boldsymbol{t}_{IC}$ from a CAD drawing or a complex setup as described in [182] has to be used. However, experiments have shown, that once the angular alignment is properly initialized, the translation converges reliably and, hence, it does not have to be initialized accurately and can be set to zero too.

The initialization of the residual parameters for the batch-optimization of the spatial alignment is described in the following.

4.2.5 Optimization Parameter Initialization

The next step is to determine the number of B-spline control points to be used and initialize them. A high number of control points means less smoothing and higher flexibility, but also increases the complexity of the computation, and the possibility of over-fitting. For our experiments in Section 8.5.2, we use $0.6\,N_C$ control points. This amount provides a good compromise between computational efficiency and accuracy, as we will show there. Nevertheless, a more general solution would be to evaluate the measurements of the accelerometers and the gyroscopes to adapt the knot vector and the number of control points to satisfy the requirements given by the motion dynamics.

The final step is the initialization of the spline control parameter vector $\boldsymbol{c}$. A good initial estimate can be found by calculating ${}^{I}\hat{\mathbf{r}}(t) = {}^{I}\hat{\mathbf{q}}_W(t)$ and ${}^{I}\hat{\mathbf{p}} = {}^{I}\hat{\mathbf{t}}_{IW}(t)$ from the camera measurements according to the following equations

$$\begin{aligned}
{}^{W}\hat{\mathbf{q}}_C(t_i) &= \prod_{j=1}^{i}{}_{\odot} \, \mathrm{Q}\left({}^{C}\boldsymbol{\omega}_j^C \, \bar{m}_C\right) && (4.57)\\
{}^{I}\hat{\mathbf{q}}_W(t_i) &= {}^{I}\hat{\boldsymbol{q}}_C \odot {}^{W}\bar{\hat{\mathbf{q}}}_C(t_i) && (4.58)\\
{}^{I}\hat{\mathbf{t}}_{IW}(t_i) &= {}^{I}\hat{\boldsymbol{t}}_{IC} - \mathrm{DCM}\left({}^{I}\hat{\mathbf{q}}_W\,(t_i)\right) \sum_{j=1}^{i} \mathrm{DCM}\left({}^{W}\hat{\mathbf{q}}_C\,(t_j)\right) \, {}^{C}\boldsymbol{v}_j^C \, \bar{m}_C \, , && (4.59)
\end{aligned}$$

whereas $i \in \{1..N_C\}$.

Subsequently, the B-spline can be approximated by following least-squares fitting equation:

$$\hat{\boldsymbol{c}} = \left(\mathbf{B}(\boldsymbol{t}_C)^T \mathbf{B}(\boldsymbol{t}_C)\right)^{-1} \mathbf{B}(\boldsymbol{t}_C)^T \begin{pmatrix} \hat{\mathbf{p}}(\boldsymbol{t}_C) \\ \hat{\mathbf{r}}(\boldsymbol{t}_C) \end{pmatrix} \tag{4.60}$$

with $\boldsymbol{t}_C = \begin{pmatrix} t_1 & \dots & t_{N_C} \end{pmatrix}$.

The last step is the initialization of the bias terms $\boldsymbol{b}_{\boldsymbol{\omega}^I}$ and $\boldsymbol{b}_{\boldsymbol{a}^I}$, the gravity vector ${}^{I_0}\boldsymbol{g}$ and the translation scale α. Both bias values are initialized with zero vectors. In general there is no prior knowledge of the initial orientation of the gravity vector ${}^{I_0}\boldsymbol{g}$ and thus, we assume that the acceleration during the first few IMU measurements is negligible small so that the sensor measurements consists mainly of the gravity force. Thus, an initial estimate can be achieved by computing the negative mean of the first few, L, samples:

$${}^{I_0}\hat{\boldsymbol{g}} = -\frac{1}{L} \sum_{i=1}^{L} {}^{I}\boldsymbol{a}_i^I \ . \tag{4.61}$$

If there is no knowledge about the used landmarks, the scale factor can be of any size. One approach to find a proper initial guess is to use the maximum measured velocity $\boldsymbol{v}^C$ of unknown scale and choose an α which yields a reasonable maximum velocity as assumed to be applied during the data acquisition. If the scale of the landmarks is known, also the translational scale is known ($\alpha = 1$) and, hence, can be removed from the parameter vector.

4.3 IMU Supported Tracking

Once the IMU-camera setup is registered the measurements can be fused. In the following, we will describe two fusing concepts: In this section, we discuss IMU supported tracking and in the next section we describe a loose Kalman fusion. These methods can then be applied together or separately.

For the sake of efficiency one should use local feature trackers if a dense image sequence is provided. However, these will in general fail if the translational or especially the rotational velocities become too high. Hence, tracking should be supported by the motion estimates of an IMU which returns reliable estimates at such high dynamics. The IMU rotation expressed in the camera frame ${}^{C}\boldsymbol{R}^I$ can be easily applied to landmark propagation by rotating each feature vector $\boldsymbol{r}$ representing the point on the unit-focal

image plane in direction to the tracked point

$$\hat{\boldsymbol{r}}_{t+1} = \frac{\hat{\boldsymbol{r}}'_{t+1}}{z_{\hat{\boldsymbol{r}}'_{t+1}}} \quad \text{with} \quad \hat{\boldsymbol{r}}'_{t+1} = {}^{C}\boldsymbol{R}^{I}_{t+1}\boldsymbol{r}_t \,. \tag{4.62}$$

In case the scale of the features is known, we could also apply the translation integrated by the acceleration measurements of the IMU after gravity subtraction. However, experiments have shown, that the double integration of the noise and bias prone acceleration measurements do not yield translation estimates which are accurate enough for landmark propagation. Hence, we propose to use a 2D linear translation propagation $\boldsymbol{d_t}$ for each feature as we will describe in Section 7.1. To get only the translational part of the optical flow, we need to subtract the rotational component from it, which we can simply compute applying ${}^{C}\boldsymbol{R}^{I}$ to the reference location

$$\hat{\boldsymbol{d}}_{\boldsymbol{t}_{t+1}} = \boldsymbol{d}_{\boldsymbol{t}_t} = \boldsymbol{d}_t - \boldsymbol{d}_{\boldsymbol{R}_t} \quad \text{with} \quad \boldsymbol{d}_{\boldsymbol{R}_t} = \hat{\boldsymbol{r}}_{t+1} - \boldsymbol{r}_t \,, \tag{4.63}$$

where $\boldsymbol{d}_t$ is the measured optical flow vector. Finally, the landmark propagation consists of

$$\hat{\boldsymbol{p}}_{t+1} = \boldsymbol{p}_t + \hat{\boldsymbol{d}}_{\boldsymbol{R}_{t+1}} + \hat{\boldsymbol{d}}_{\boldsymbol{t}_{t+1}} \,. \tag{4.64}$$

Thus, the rotation becomes propagated based on the IMU measurement while the translation is predicted in 2D, as illustrated in Fig. 4.1. The camera measurements and its motion estimation are consequently completely separated which prevents an accumulation of the estimation errors.

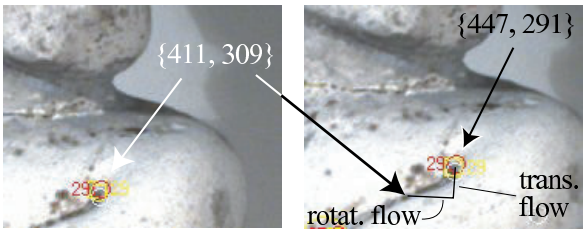


Figure 4.1: Two corresponding features in consecutive images acquired during high dynamic motions at a frame rate of 25 Hz. The feature drifts by 40.2 pixels, whereas 37 pixels result from rotation and 17 pixels from translation (some pixels cancel out). The red circle shows the feature propagation by the IMU and translation prediction in 2D. The yellow square illustrates the tracking result. The estimated location of the feature corresponds well enough to allow for local tracking.

Using such a prediction step, the bottleneck of successful tracking during fast motions (in the experiments we measured successful tracking up to $75\,°/s$ and $0.5\,m/s$)

are not anymore the large feature displacements between two images (up to 50 pixels in the experiments), but mainly the motion blur[1]. The shutter time of the cameras can not be arbitrarily reduced, because short exposure times force the camera gain to be increased, which amplifies the image noise and impedes proper tracking. In Section 8.5.3, we will illustrate a trajectory of a high dynamic motion and the successful prediction by fusing IMU measurements.

4.4 Kalman Filter Based Fusion

While an IMU aided tracking improves the robustness of image processing significantly, it does not improve the motion estimation directly. One way to fuse both motion estimates is by means of a Kalman filter. Such filters convince by their computational efficiency, while being optimal for linear problems with zero-mean, white Gaussian noise. Unfortunately, our problem is highly nonlinear. These nonlinearities can be approximated by making use of Kalman filter derivatives, *e.g.* the EKF or the UKF. We chose to realize our fusion as a loosely coupled EKF for the sake of an efficient solution which can run at high rates on a resource-limited platform. To improve the linearization properties, we make use of an error-state EKF [210, 111]. In the following filter, we aim at combining relative measurement updates with the camera uncertainties modeled in the respective domain as well as a fast convergence of the translational scale.

4.4.1 Dynamic Model

Similar to most solutions presented in literature, our dynamic model for an inertial-visual fusion consists of the following variables: position ${}^{W}\boldsymbol{t}_{WI}$, velocity ${}^{W}\boldsymbol{v}_{WI}$, attitude (as quaternion) ${}^{W}\boldsymbol{q}_{I}$, bias of the accelerometers ${}^{I}\boldsymbol{b}_{\boldsymbol{a}^{I}}$ and the bias of the gyroscopes ${}^{I}\boldsymbol{b}_{\boldsymbol{\omega}^{I}}$. The uncertainties, $\boldsymbol{n}_{\boldsymbol{x}}$, in the pose estimation are modeled as additive zero-mean, white Gaussian noise on the translational acceleration ${}^{W}\boldsymbol{n}_{\boldsymbol{a}}$ and on the rotational velocity ${}^{W}\boldsymbol{n}_{\boldsymbol{\omega}}$. The biases are modeled as random walk processes which are also driven by additive white Gaussian noise, ${}^{I}\boldsymbol{n}_{\boldsymbol{b}_{\boldsymbol{a}}}$ and ${}^{I}\boldsymbol{n}_{\boldsymbol{b}_{\boldsymbol{\omega}}}$. The pose and the velocity are estimated as motion of the IMU frame $\mathcal{I}$ relative to the coordinate frame $\mathcal{W}$, expressed in $\mathcal{W}$. The biases are estimated in $\mathcal{I}$. For the sake of readability, we will drop the frame indices from now on and only mention them in cases where they are required for understanding.

[1] A short video is available at http://www6.in.tum.de/~maire/videos/3dMo_imuTracking.avi

To reduce the linearization errors of the EKF, we represent the dynamic model in the error state space with the error state vector $\boldsymbol{\delta_x}$, such that

$$\boldsymbol{\delta_x} = \begin{pmatrix} \boldsymbol{\delta_t}^T & \boldsymbol{\delta_v}^T & \boldsymbol{\delta_{\tilde{q}}}^T & \boldsymbol{\delta_{b_a}}^T & \boldsymbol{\delta_{b_\omega}}^T \end{pmatrix}^T . \tag{4.65}$$

This representation also allows to represent a quaternion as the vector of its three imaginary elements, $\boldsymbol{\delta}_{\tilde{\boldsymbol{q}}}^T$, by applying a small angle approximation according to Eq. 2.6. The real part of the quaternion is approximated as one and can, thus, be neglected (see Eq. 2.7). As further advantage we yield a minimal representation of the rotational DOF, which results in $\boldsymbol{\delta_x} \in \mathbb{R}^{15}$.

The continuous-time transition can, thus, be modeled as follows:

$$\dot{\boldsymbol{\delta}}_{\boldsymbol{x}} = \boldsymbol{F}\,\boldsymbol{\delta_x} + \boldsymbol{G}\,\boldsymbol{n_x} , \tag{4.66}$$

whereas

$$\boldsymbol{F} = \begin{pmatrix} \mathbf{0}_{3\times3} & \boldsymbol{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -2\left[{}^W\boldsymbol{a}\right]_\times & -{}^W\boldsymbol{R}_I & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\frac{1}{2}{}^W\boldsymbol{R}_I \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix} , \tag{4.67}$$

$$\boldsymbol{G} = \begin{pmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ {}^W\boldsymbol{R}_I & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & 2{}^W\boldsymbol{R}_I & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{I}_3 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{I}_3 \end{pmatrix} \tag{4.68}$$

and

$$\boldsymbol{Q} = \mathrm{Diag}(\boldsymbol{Q_a}, \boldsymbol{Q_\omega}, \boldsymbol{Q_{b_a}}, \boldsymbol{Q_{b_\omega}}) . \tag{4.69}$$

The noise matrices $\boldsymbol{Q_a}$, $\boldsymbol{Q_\omega}$, $\boldsymbol{Q_{b_a}}$, and $\boldsymbol{Q_{b_\omega}}$ correspond to the expectation value of the noise variables

$$\boldsymbol{Q}_x = \mathrm{E}\left[\boldsymbol{n}_x\right] \mid x \in \{\boldsymbol{a}, \boldsymbol{\omega}, \boldsymbol{b_a}, \boldsymbol{b_\omega}\} . \tag{4.70}$$

In a real system we do not have continuous but discrete time. Hence, we compute the discrete-time transition and introduce the prior and posterior index to get

$$\boldsymbol{x}_{k+1}^- = \boldsymbol{\Phi}_{k+1}\,\boldsymbol{x}_k^+ + \boldsymbol{G}_{k+1}\,\boldsymbol{n}_{\boldsymbol{x};k} \tag{4.71}$$

with

$$\boldsymbol{\Phi} = e^{\Delta_t \boldsymbol{F}} = \begin{pmatrix} \boldsymbol{I}_3 & 2\Delta_t \boldsymbol{I}_3 & -\frac{1}{2}\Delta_t^2 \left[{}^W\boldsymbol{a}\right]_\times & -\frac{1}{2}\Delta_t^2 {}^W\boldsymbol{R}_I & \frac{1}{6}\Delta_t^3 \left[{}^W\boldsymbol{a}\right]_\times {}^W\boldsymbol{R}_I \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & -2\Delta_t \left[{}^W\boldsymbol{a}\right]_\times & -\Delta_t {}^W\boldsymbol{R}_I & \frac{1}{2}\Delta_t^2 \left[{}^W\boldsymbol{a}\right]_\times {}^W\boldsymbol{R}_I \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} & -\frac{1}{2}\Delta_t {}^W\boldsymbol{R}_I \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 \end{pmatrix} \tag{4.72}$$

and $\boldsymbol{G}_k \approx \Delta_t \boldsymbol{G}$.

The uncertainty of the estimated states is expressed by the covariance matrix $\boldsymbol{P}$, which is propagated according to Eq. 2.19. At the beginning this matrix is initialized as diagonal matrix, where the entries represent the estimated variances for the initial guess, $\sigma^2_{x;0}$, of the corresponding state

$$\boldsymbol{P}_0^- = \operatorname{diag}\left(\sigma^2_{\boldsymbol{t}_x;0}\,,\, \sigma^2_{\boldsymbol{t}_y;0}\,,\, \sigma^2_{\boldsymbol{t}_z;0}\,,\, \sigma^2_{\boldsymbol{v}_x;0}\,,\, \ldots\,,\, \sigma^2_{\boldsymbol{b}_{\omega;z};0}\right)\,. \tag{4.73}$$

4.4.2 EKF Propagation and Update

The IMU measurements are now used to propagate the state in time by applying the strapdown computation as described in Section 2.6. The more tricky part is how to fuse the camera measurements. In order to prevent drifting when being static our visual motion estimation should stick to a keyframe as long as there are enough features available. To deal with such relative motion measurements, we enhance the state vector by stochastic cloning of the actual state [200, 211]. Furthermore, we use a monocular camera, which does not allow to compute the proper scale from natural landmarks. Hence, we also require to estimate the proper camera scale s. This yields following augmented and extended state vector $\boldsymbol{\delta}_{\breve{\boldsymbol{x}};r,k} \in \mathbb{R}^{31}$, with

$$\boldsymbol{\delta}_{\breve{\boldsymbol{x}};r,k} = \begin{pmatrix} \boldsymbol{\delta}_{\boldsymbol{x};r} \\ \boldsymbol{\delta}_{\boldsymbol{x};k} \\ s \end{pmatrix} \tag{4.74}$$

and $\boldsymbol{\delta}_{\boldsymbol{x};r}$ denoting the estimated error state at the reference time of the keyframe acquisition and $\boldsymbol{\delta}_{\boldsymbol{x};k}$ is the currently estimated error state.

As long as we reference to a specific keyframe we assume a constant scale and, thus, $n_s = 0$. Hence, the matrices for the state propagation become

$$\breve{\boldsymbol{\Phi}}_k = \begin{pmatrix} \boldsymbol{I}_{15} & \boldsymbol{0}_{15\times15} & \boldsymbol{0}_{15\times1} \\ \boldsymbol{0}_{15\times15} & \boldsymbol{\Phi}_k & \boldsymbol{0}_{15\times1} \\ \boldsymbol{0}_{1\times15} & \boldsymbol{0}_{1\times15} & 1 \end{pmatrix}\,, \tag{4.75}$$

$$\breve{\boldsymbol{G}}_k = \begin{pmatrix} \mathbf{0}_{15\times 12} \\ \boldsymbol{G}_k \\ \mathbf{0}_{1\times 12} \end{pmatrix} \tag{4.76}$$

and

$$\breve{\boldsymbol{P}}_0^- = \begin{pmatrix} \boldsymbol{P}_0^- & \boldsymbol{P}_0^- & \mathbf{0}_{15\times 1} \\ \boldsymbol{P}_0^- & \boldsymbol{P}_0^- & \mathbf{0}_{15\times 1} \\ \mathbf{0}_{1\times 15} & \mathbf{0}_{1\times 15} & \sigma_{s;0}^2 \end{pmatrix} . \tag{4.77}$$

Every time a keyframe-switch occurs and the scale of the visual odometry accumulates an error, we model that uncertainty as additional noise on the scale, n_s, by modified model noise matrices $\breve{\boldsymbol{G}}_{s;k}$ and $\boldsymbol{Q}_{s;k}$, such that

$$\breve{\boldsymbol{G}}_{s;k} = \begin{pmatrix} \mathbf{0}_{15\times 12} & \mathbf{0}_{15\times 1} \\ \boldsymbol{G}_k & \mathbf{0}_{15\times 1} \\ \mathbf{0}_{1\times 12} & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{Q}_{s;k} = \operatorname{Diag}(\boldsymbol{Q}_{\boldsymbol{a}}, \boldsymbol{Q}_{\boldsymbol{\omega}}, \boldsymbol{Q}_{\boldsymbol{b_a}}, \boldsymbol{Q}_{\boldsymbol{b_\omega}}, q_s) \tag{4.78}$$

with $q_s = \mathrm{E}[n_s]$.

The second part of a Kalman filter consists in the update of the state by a measurement. To provide the relation between the sensor and our state representation, we have to derive the linearized mapping $\boldsymbol{H}_{r,k}$ between error state and measurement $\boldsymbol{z}_{r,k}$

$$\boldsymbol{z}_{r,k} = \boldsymbol{H}_{r,k}\, \boldsymbol{\delta}_{\breve{\boldsymbol{x}};r,k} + \boldsymbol{n}_{\boldsymbol{z};r,k} , \tag{4.79}$$

which can be computed analogue to [211].

The relation between the rotation error and the rotation measurement can be derived as follows. Let the estimated rotation ${}^{Io}\widehat{\boldsymbol{q}}_{Ir}$ denote the rotation from the IMU frame at time instance r, the keyframe acquisition, to the IMU reference frame, at time instance $o = 0$. Then the error at time instance r, $\boldsymbol{\delta}_{\boldsymbol{q};r}$, can be formulated as

$${}^{Io}\boldsymbol{q}_{Ir} = \boldsymbol{\delta}_{\boldsymbol{q};r} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir} . \tag{4.80}$$

The camera measurement relative to the keyframe is described as the real state corrupted by noise $\boldsymbol{n}_{\boldsymbol{z_q};r,k}$

$${}^{C;r}\boldsymbol{z}_{\boldsymbol{q};C;k} = {}^{C;r}\boldsymbol{q}_{C;k} + \boldsymbol{n}_{\boldsymbol{z_q};r,k} . \tag{4.81}$$

Thus, the measurement error $\boldsymbol{\delta}_{\boldsymbol{z_q};r,k}$ can be computed by

$$\begin{aligned} \boldsymbol{\delta}_{\boldsymbol{z_q};r,k} &= {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot {}^{Cr}\boldsymbol{z}_{\boldsymbol{q};Ck}^{-1} = {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot \left({}^{Cr}\boldsymbol{q}_{Ck} + \boldsymbol{n}_{\boldsymbol{z_q}}\right)^{-1} \\ &= {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot \left({}^{Ck}\boldsymbol{q}_{Cr} + \bar{\boldsymbol{n}}_{\boldsymbol{z_q}}\right) = {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot {}^{Ck}\boldsymbol{q}_{Cr} + {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot \bar{\boldsymbol{n}}_{\boldsymbol{z_q};r,k} , \end{aligned} \tag{4.82}$$

where ${}^{X}\boldsymbol{q}_{Y}^{-1} = {}^{Y}\boldsymbol{q}_{X}$ denotes the inverse rotation and with

$$\bar{\boldsymbol{n}}_{\boldsymbol{z_q};r,k} = \operatorname{diag}(-1,-1,-1,1)\, \boldsymbol{n}_{\boldsymbol{z_q};r,k} \,. \tag{4.83}$$

Assuming that the sensor suffers from zero-mean, white Gaussian noise, we can neglect Eq. 4.83 and say that $\boldsymbol{n}_{\boldsymbol{z_q};r,k} = \bar{\boldsymbol{n}}_{\boldsymbol{z_q};r,k}$.

The rotation between the IMU and the camera frame, ${}^{I}\boldsymbol{q}_{C}$, is assumed to be computed in an offline calibration step, *e.g.*, according to Section 4.2. Hence, we ignore the noise for a moment, with $\boldsymbol{\delta}'_{\boldsymbol{z_q};r,k} = \boldsymbol{\delta}_{\boldsymbol{z_q};r,k} - {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot \boldsymbol{n}_{\boldsymbol{z_q};r,k}$, and extend Eq. 4.82, such that we get a direct relation between the measurement and the estimated error:

$$\begin{aligned}
\boldsymbol{\delta}'_{\boldsymbol{z_q};r,k} &= {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot {}^{Ck}\boldsymbol{q}_{Cr} \\
&= {}^{I}\boldsymbol{q}_{C}^{-1} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir}^{-1} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ik} \odot {}^{I}\boldsymbol{q}_{C} \odot {}^{I}\boldsymbol{q}_{C}^{-1} \odot \left(\boldsymbol{\delta}_{\boldsymbol{q};k} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ik}\right)^{-1} \odot \left(\boldsymbol{\delta}_{\boldsymbol{q};r} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir}\right) \odot {}^{I}\boldsymbol{q}_{C} \\
&= {}^{I}\boldsymbol{q}_{C}^{-1} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir}^{-1} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ik} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ik}^{-1} \odot \boldsymbol{\delta}_{\boldsymbol{q};k}^{-1} \odot \boldsymbol{\delta}_{\boldsymbol{q};r} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir} \odot {}^{I}\boldsymbol{q}_{C} \\
&= {}^{I}\boldsymbol{q}_{C}^{-1} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir}^{-1} \odot \boldsymbol{\delta}_{\boldsymbol{q};k}^{-1} \odot \boldsymbol{\delta}_{\boldsymbol{q};r} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir} \odot {}^{I}\boldsymbol{q}_{C} \,.
\end{aligned} \tag{4.84}$$

Next, we make use of the fact that the rotation of small angles can be approximated by the sum of the quaternions and the approximation defined in Eq. 2.6, such that

$$\begin{aligned}
\boldsymbol{\delta}_{\boldsymbol{z_q};r,k} &= {}^{I}\boldsymbol{q}_{C}^{-1} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir}^{-1} \odot \boldsymbol{\delta}_{\boldsymbol{q};k}^{-1} + \boldsymbol{\delta}_{\boldsymbol{q};r} \odot {}^{Io}\widehat{\boldsymbol{q}}_{Ir} \odot {}^{I}\boldsymbol{q}_{C} + {}^{Cr}\widehat{\boldsymbol{q}}_{Ck} \odot \boldsymbol{n}_{\boldsymbol{z_q};r,k} \\
&= -{}^{I}\boldsymbol{R}_{C}^{T}\, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T} \left(\frac{1}{2}\boldsymbol{\sigma}_{\boldsymbol{\delta}_{\boldsymbol{q};k}}\right) + {}^{Io}\widehat{\boldsymbol{R}}_{Ir}\, {}^{I}\boldsymbol{R}_{C} \left(\frac{1}{2}\boldsymbol{\sigma}_{\boldsymbol{\delta}_{\boldsymbol{q};r}}\right) + {}^{Cr}\widehat{\boldsymbol{R}}_{Ck} \left(2\,\tilde{\boldsymbol{n}}_{\boldsymbol{z_q};r,k}\right) \\
&= {}^{Io}\widehat{\boldsymbol{R}}_{Ir}\, {}^{I}\boldsymbol{R}_{C}\, \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r} - {}^{Io}\widehat{\boldsymbol{R}}_{Ir}\, {}^{I}\boldsymbol{R}_{C}\, \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k} + 2\, {}^{Cr}\widehat{\boldsymbol{R}}_{Ck} \tilde{\boldsymbol{n}}_{\boldsymbol{z_q};r,k} \,,
\end{aligned} \tag{4.85}$$

with $\tilde{\boldsymbol{n}}_{\boldsymbol{z_q};r,k} = \boldsymbol{n}_{\boldsymbol{z_q};r,k;1:3}$.

Let us now consider the translational part of the measurement matrix. The relation between estimated translation, its error and the real translation is provided by

$${}^{Io}\boldsymbol{t}_{Io,Ik} = {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - \boldsymbol{\delta}_{\boldsymbol{t};k} \,. \tag{4.86}$$

An analog relation applies for the scale, such that

$$s = \widehat{s} - \delta_{s} \,. \tag{4.87}$$

The camera measurement for the translation is again described in terms of the real state corrupted by some noise $\boldsymbol{n}_{\boldsymbol{z_t}}$

$${}^{Cr}\boldsymbol{z}_{\boldsymbol{t};Cr,Ck} = s\, {}^{Cr}\boldsymbol{t}_{Cr,Ck} + \boldsymbol{n}_{\boldsymbol{z_t}} \,. \tag{4.88}$$

The measurement error $\boldsymbol{\delta}_{\boldsymbol{z_t}}$ can be computed by

$$\begin{aligned}
\boldsymbol{\delta}_{\boldsymbol{z_t}} &= \widehat{s}\,{}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} - {}^{Cr}\boldsymbol{z}_{\boldsymbol{t};Cr,Ck} \\
&= \widehat{s}\,{}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} - s\,{}^{Cr}\boldsymbol{t}_{Cr,Ck} - \boldsymbol{n}_{\boldsymbol{z_t}} \\
&= \widehat{s}\,{}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} - (\widehat{s} - \delta_s)\,{}^{Cr}\boldsymbol{t}_{Cr,Ck} - \boldsymbol{n}_{\boldsymbol{z_t}} \\
&= \widehat{s}\,{}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} - \widehat{s}\,{}^{Cr}\boldsymbol{t}_{Cr,Ck} + \delta_s\,{}^{Cr}\boldsymbol{t}_{Cr,Ck} - \boldsymbol{n}_{\boldsymbol{z_t}} \\
&= \widehat{s}\,\left({}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} - {}^{Cr}\boldsymbol{t}_{Cr,Ck}\right) + {}^{Cr}\boldsymbol{t}_{Cr,Ck}\,\delta_s - \boldsymbol{n}_{\boldsymbol{z_t}}\,. \qquad (4.89)
\end{aligned}$$

For the sake of readability, the terms are extended separately, resulting in

$$\begin{aligned}
{}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} &= {}^{Cr}\widehat{\boldsymbol{t}}_{Io,Ck} - {}^{Cr}\widehat{\boldsymbol{t}}_{Io,Cr} \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\left({}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\,{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} + {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\,{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} - {}^{I}\boldsymbol{t}_{I,C}\right) \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\left({}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} + {}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} - {}^{Io}\widehat{\boldsymbol{R}}_{Ir}\,{}^{I}\boldsymbol{t}_{I,C}\right)\,, \qquad (4.90)
\end{aligned}$$

whereas the translation between IMU and camera, ${}^{I}\boldsymbol{t}_{I,C}$, is assumed to be computed in an offline step, *e.g.*, according to the method presented in Section 4.2. By applying Eq. 4.80 and the small angle approximation described in Eq. 2.8, we get

$$\begin{aligned}
{}^{Cr}\boldsymbol{t}_{Cr,Ck} &= {}^{Cr}\boldsymbol{t}_{Io,Ck} - {}^{Cr}\boldsymbol{t}_{Io,Cr} \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\,{}^{Io}\boldsymbol{R}_{Ir}^{T}\left({}^{Io}\boldsymbol{t}_{Io,Ik} + {}^{Io}\boldsymbol{R}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - {}^{Io}\boldsymbol{t}_{Io,Ir} - {}^{Io}\boldsymbol{R}_{Ir}\,{}^{I}\boldsymbol{t}_{I,C}\right) \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\left(\boldsymbol{\Delta}_{\boldsymbol{R};r}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}\right)^{T}\left({}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - \boldsymbol{\delta}_{\boldsymbol{t};k} + \boldsymbol{\Delta}_{\boldsymbol{R};k}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - \right. \\
&\qquad \left. {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + \boldsymbol{\delta}_{\boldsymbol{t};r} - \boldsymbol{\Delta}_{\boldsymbol{R};r}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}\,{}^{I}\boldsymbol{t}_{I,C}\right) \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\,\boldsymbol{\Delta}_{\boldsymbol{R};r}^{T}\left({}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - \boldsymbol{\delta}_{\boldsymbol{t};k} + \boldsymbol{\Delta}_{\boldsymbol{R};k}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - \right. \\
&\qquad \left. {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + \boldsymbol{\delta}_{\boldsymbol{t};r} - \boldsymbol{\Delta}_{\boldsymbol{R};r}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}\,{}^{I}\boldsymbol{t}_{I,C}\right) \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\left(-{}^{Io}\widehat{\boldsymbol{R}}_{Ir}\,{}^{I}\boldsymbol{t}_{I,C} + \left(\boldsymbol{I}_3 - 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}\right)\right. \\
&\qquad \left.\left({}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - \boldsymbol{\delta}_{\boldsymbol{t};k} + \left(\boldsymbol{I}_3 + 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}\right]_{\times}\right){}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + \boldsymbol{\delta}_{\boldsymbol{t};r}\right)\right) \\
&= {}^{I}\boldsymbol{R}_{C}^{T}\,{}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\left(-{}^{Io}\widehat{\boldsymbol{R}}_{Ir}\,{}^{I}\boldsymbol{t}_{I,C} + {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - \boldsymbol{\delta}_{\boldsymbol{t};k} + {}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} + \right. \\
&\quad 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}\right]_{\times}{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + \boldsymbol{\delta}_{\boldsymbol{t};r} - 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} + \\
&\quad 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}\boldsymbol{\delta}_{\boldsymbol{t};k} - 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} - 4\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}\right]_{\times}{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\,{}^{I}\boldsymbol{t}_{I,C} + \\
&\quad \left. 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} - 2\left[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}\right]_{\times}\boldsymbol{\delta}_{\boldsymbol{t};r}\right)\,, \qquad (4.91)
\end{aligned}$$

with $\boldsymbol{\Delta}_{\boldsymbol{R}} = \mathrm{DCM}(\boldsymbol{\delta}_{\boldsymbol{q}})$. We make the assumption that the products of two error terms

are negligible

$$[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, \boldsymbol{\delta}_{\boldsymbol{t};r} \approx \mathbf{0} \tag{4.92}$$

$$[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, \boldsymbol{\delta}_{\boldsymbol{t};k} \approx \mathbf{0} \tag{4.93}$$

$$[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, [\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} \approx \mathbf{0} \,, \tag{4.94}$$

such that

$$\begin{aligned} {}^{Cr}\boldsymbol{t}_{Cr,Ck} \approx {}^{I}\boldsymbol{R}_C^T \, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^T \Big(& {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} + {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} - {}^{Io}\widehat{\boldsymbol{R}}_{Ir} \, {}^{I}\boldsymbol{t}_{I,C} - \\ & \boldsymbol{\delta}_{\boldsymbol{t};k} + \boldsymbol{\delta}_{\boldsymbol{t};r} + 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} - 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} - \\ & 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} + 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} \Big) \,. \end{aligned} \tag{4.95}$$

By combining Eq. 4.95 and 4.90 as well as the rule $[a]_\times \, b = -\,[b]_\times \, a$, we get

$$\begin{aligned} {}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} - {}^{Cr}\boldsymbol{t}_{Cr,Ck} &\approx {}^{I}\boldsymbol{R}_C^T \, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^T \Big(\boldsymbol{\delta}_{\boldsymbol{t};k} - \boldsymbol{\delta}_{\boldsymbol{t};r} - 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} + \\ &\qquad 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} + 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} \Big) \\ &\approx {}^{I}\boldsymbol{R}_C^T \, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^T \Big(\boldsymbol{\delta}_{\boldsymbol{t};k} - \boldsymbol{\delta}_{\boldsymbol{t};r} + 2\left[{}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} \right]_\times \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k} - \\ &\qquad 2\left[{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C} \right]_\times \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r} \Big) \end{aligned} \tag{4.96}$$

Extending the third term of Eq. 4.89

$$\begin{aligned} {}^{Cr}\boldsymbol{t}_{Cr,Ck}\,\delta_s \approx {}^{I}\boldsymbol{R}_C^T \, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^T \Big(& {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik}\,\delta_s + {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C}\,\delta_s - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir}\,\delta_s - {}^{Io}\widehat{\boldsymbol{R}}_{Ir} \, {}^{I}\boldsymbol{t}_{I,C}\,\delta_s - \\ & \boldsymbol{\delta}_{\boldsymbol{t};k}\,\delta_s + \boldsymbol{\delta}_{\boldsymbol{t};r}\,\delta_s + 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C}\,\delta_s - 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C}\,\delta_s - \\ & 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik}\,\delta_s + 2\,[\boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r}]_\times \, {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir}\,\delta_s \Big) \end{aligned} \tag{4.97}$$

and assuming again that the product of error terms is negligible we get

$$\begin{aligned} {}^{Cr}\boldsymbol{t}_{Cr,Ck}\,\delta_s &\approx {}^{I}\boldsymbol{R}_C^T \, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^T \Big({}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik}\,\delta_s + {}^{Io}\widehat{\boldsymbol{R}}_{Ik} \, {}^{I}\boldsymbol{t}_{I,C}\,\delta_s - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir}\,\delta_s - {}^{Io}\widehat{\boldsymbol{R}}_{Ir} \, {}^{I}\boldsymbol{t}_{I,C}\,\delta_s \Big) \\ &\approx {}^{I}\boldsymbol{R}_C^T \, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^T \, {}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck}\,\delta_s \,. \end{aligned} \tag{4.98}$$

Now we can insert Eq. 4.96 and 4.98 into Eq. 4.89 and yield

$$\begin{aligned}\boldsymbol{\delta}_{\boldsymbol{z_t}} \approx\ & \widehat{s}\, {}^{I}\boldsymbol{R}_{C}^{T}\, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T} \Big(\boldsymbol{\delta}_{\boldsymbol{t};k} - \boldsymbol{\delta}_{\boldsymbol{t};r} + 2 \left[{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\, {}^{I}\boldsymbol{t}_{I,C} \right]_{\times} \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k} - \\ & 2 \left[{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + {}^{Io}\widehat{\boldsymbol{R}}_{Ik}\, {}^{I}\boldsymbol{t}_{I,C} \right]_{\times} \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r} \Big) + \\ & {}^{I}\boldsymbol{R}_{C}^{T}\, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}\, {}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck}\, \delta_s - \boldsymbol{n}_{\boldsymbol{z_t}} \,. \end{aligned} \tag{4.99}$$

The measurement matrix can then be determined by combining Eq. 4.85 and 4.99:

$$\begin{aligned}\boldsymbol{z}'_{r,k} =\ & \begin{pmatrix} \boldsymbol{H}_{1,1} & \boldsymbol{0}_{3\times3} & \boldsymbol{H}_{1,3} & \boldsymbol{0}_{3\times6} & \boldsymbol{H}_{1,6} & \boldsymbol{0}_{3\times3} & \boldsymbol{H}_{1,8} & \boldsymbol{0}_{3\times6} & \boldsymbol{h}_{1,11} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{H}_{2,3} & \boldsymbol{0}_{3\times6} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{H}_{2,8} & \boldsymbol{0}_{3\times6} & \boldsymbol{0}_{3\times1} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta}_{\boldsymbol{t};r} \\ \boldsymbol{\delta}_{\boldsymbol{v};r} \\ \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};r} \\ \boldsymbol{\delta}_{\boldsymbol{b_a};r} \\ \boldsymbol{\delta}_{\boldsymbol{b_\omega};r} \\ \boldsymbol{\delta}_{\boldsymbol{t};k} \\ \boldsymbol{\delta}_{\boldsymbol{v};k} \\ \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k} \\ \boldsymbol{\delta}_{\boldsymbol{b_a};k} \\ \boldsymbol{\delta}_{\boldsymbol{b_\omega};k} \\ \boldsymbol{\delta}_{s} \end{pmatrix} \\ \overset{def}{=}\ & \boldsymbol{H}_{r,k}\, \boldsymbol{\delta}_{\breve{\boldsymbol{x}};r,k} \,, \end{aligned} \tag{4.100}$$

with

$$\begin{aligned} \boldsymbol{H}_{1,1} &= -\widehat{s}\, {}^{Cr}\widehat{\boldsymbol{R}}_{Io} \\ \boldsymbol{H}_{1,3} &= -2\, \widehat{s}\, {}^{Cr}\widehat{\boldsymbol{R}}_{Io} \left[{}^{Io}\widehat{\boldsymbol{t}}_{Io,Ik} - {}^{Io}\widehat{\boldsymbol{t}}_{Io,Ir} + {}^{Io}\widehat{\boldsymbol{R}}_{Ik}\, {}^{I}\boldsymbol{t}_{I,C} \right]_{\times} \\ \boldsymbol{H}_{1,6} &= \widehat{s}\, {}^{Cr}\widehat{\boldsymbol{R}}_{Io} \\ \boldsymbol{H}_{1,8} &= 2\, \widehat{s}\, {}^{Cr}\widehat{\boldsymbol{R}}_{Io} \left[{}^{Io}\widehat{\boldsymbol{R}}_{Ik}\, {}^{I}\boldsymbol{t}_{I,C} \right]_{\times} \\ \boldsymbol{h}_{1,11} &= {}^{Cr}\widehat{\boldsymbol{R}}_{Io}\, {}^{Cr}\widehat{\boldsymbol{t}}_{Cr,Ck} \\ \boldsymbol{H}_{2,3} &= {}^{Cr}\widehat{\boldsymbol{R}}_{Io} \\ \boldsymbol{H}_{2,8} &= -{}^{Cr}\widehat{\boldsymbol{R}}_{Io} \,, \end{aligned}$$

whereas $\boldsymbol{z}'_{r,k} = \boldsymbol{z}_{r,k} - \boldsymbol{n}_{\boldsymbol{z};r,k}$ and ${}^{Cr}\widehat{\boldsymbol{R}}_{Io} = {}^{I}\boldsymbol{R}_{C}^{T}\, {}^{Io}\widehat{\boldsymbol{R}}_{Ir}^{T}$. The corresponding measurement noise covariance matrix $\boldsymbol{L}_{r,k}$ is computed by

$$\boldsymbol{L}_{r,k} = \boldsymbol{N}_{r,k}\, \mathrm{Diag}\left(\boldsymbol{L}'_{\boldsymbol{n_{z;t}};r,k}\,,\, \boldsymbol{L}'_{\tilde{\boldsymbol{n}}_{\boldsymbol{z;q}};r,k} \right) \boldsymbol{N}_{r,k}^{T} \tag{4.101}$$

with

$$\boldsymbol{N}_{r,k} = \begin{pmatrix} -\boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & 2\,{}^{Cr}\widehat{\boldsymbol{R}}_{Ck} \end{pmatrix} \quad \text{and} \quad \boldsymbol{L}'_{x;r,k} = \mathrm{E}\left[x\right] \mid x \in \left\{\boldsymbol{n}_{\boldsymbol{z}_t}, \tilde{\boldsymbol{n}}_{\boldsymbol{z}_q}\right\} . \tag{4.102}$$

4.4.3 Pseudo-Measurement Model

Up to now we considered the translation measurements and their noise to be provided in the Cartesian coordinate frame of our system model. Thinking of a camera as a goniometer, we measure the epipol and try to keep the translational scale constant. This measurements are actually not expressed in a Cartesian frame, but in a polar coordinate frame, where the epipol determines the bearing (or azimuth) ϕ and the elevation ψ. The range r is then computed by keeping the relative scales of the landmarks constant. This yields a measurement vector of the form

$${}^{P}\boldsymbol{z} = \begin{pmatrix} {}^{P}\boldsymbol{z}_r \\ {}^{P}\boldsymbol{z}_\phi \\ {}^{P}\boldsymbol{z}_\psi \end{pmatrix} . \tag{4.103}$$

Similar to [203], we apply a pseudo-measurement model which allows to represent the measurements and the noise in the *line-of-sight* (LOS) coordinate system. The x-axis of this coordinate frame is always along the direction of translation. Thus, any translation measurement in LOS coordinates is of the form $\mathring{\boldsymbol{z}} = \begin{pmatrix} {}^{P}\boldsymbol{z}_r & 0 & 0 \end{pmatrix}^T$. The mapping between Cartesian and LOS coordinates is defined by

$$\mathring{\boldsymbol{z}} = \begin{pmatrix} {}^{P}\boldsymbol{z}_r \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\left({}^{P}\boldsymbol{z}_\phi\right)\cos\left({}^{P}\boldsymbol{z}_\psi\right) & \sin\left({}^{P}\boldsymbol{z}_\phi\right)\cos\left({}^{P}\boldsymbol{z}_\psi\right) & \sin\left({}^{P}\boldsymbol{z}_\psi\right) \\ -\sin\left({}^{P}\boldsymbol{z}_\phi\right) & \cos\left({}^{P}\boldsymbol{z}_\phi\right) & 0 \\ -\cos\left({}^{P}\boldsymbol{z}_\phi\right)\sin\left({}^{P}\boldsymbol{z}_\psi\right) & -\sin\left({}^{P}\boldsymbol{z}_\phi\right)\sin\left({}^{P}\boldsymbol{z}_\psi\right) & \cos\left({}^{P}\boldsymbol{z}_\psi\right) \end{pmatrix} \boldsymbol{z} \overset{def}{=} \boldsymbol{M}_{\boldsymbol{z}}\,\boldsymbol{z} \tag{4.104}$$

and

$$\boldsymbol{z} = \boldsymbol{M}_{\boldsymbol{z}}^{-1}\,\mathring{\boldsymbol{z}} = \boldsymbol{M}_{\boldsymbol{z}}^{T}\,\mathring{\boldsymbol{z}} . \tag{4.105}$$

Recapitulating the Kalman state transition, we formulate the new representation as

$$\begin{aligned} \boldsymbol{x}_{k+1}^{+} &= \boldsymbol{\Phi}_{k+1}\boldsymbol{x}_k^{+} + \boldsymbol{K}_{k+1}\boldsymbol{M}_{\boldsymbol{z}_{k+1}}^{T}\left(\mathring{\boldsymbol{z}}_{k+1} - \boldsymbol{M}_{\boldsymbol{z}_{k+1}}\boldsymbol{H}_{k+1}\boldsymbol{x}_{k+1}^{-}\right) \\ &= \boldsymbol{\Phi}_{k+1}\boldsymbol{x}_k^{+} + \mathring{\boldsymbol{K}}_{k+1}\left(\mathring{\boldsymbol{z}}_{k+1} - \mathring{\boldsymbol{H}}_{k+1}\boldsymbol{x}_{k+1}^{-}\right) . \end{aligned} \tag{4.106}$$

This results in the following error state transition

$$\delta_{\boldsymbol{x};k+1}^{+} = \delta_{\boldsymbol{x};k+1}^{-} + \mathring{\boldsymbol{K}}_{k+1}\,\mathring{\boldsymbol{r}}_{k+1} . \tag{4.107}$$

The measurement covariance matrix $\mathring{\boldsymbol{L}}$ can then be computed, analogue to Eq. 4.101, by

$$\mathring{\boldsymbol{L}}_{r,k} = \boldsymbol{N}_{r,k}\,\mathrm{Diag}\left(\mathring{\boldsymbol{L}}'_{\boldsymbol{n}_{\boldsymbol{z};\boldsymbol{t};r,k}}, \boldsymbol{L}'_{\tilde{\boldsymbol{n}}_{\boldsymbol{z};\boldsymbol{q};r,k}}\right)\boldsymbol{N}_{r,k}^{T}\,, \tag{4.108}$$

with

$$\mathring{\boldsymbol{L}}'_{\boldsymbol{n}_{\boldsymbol{z};\boldsymbol{t};r,k}} = \begin{pmatrix} {}^{P}\boldsymbol{n}_{\boldsymbol{z}_r}\cos\left({}^{P}\boldsymbol{n}_{\boldsymbol{z}_\phi}\right)\cos\left({}^{P}\boldsymbol{n}_{\boldsymbol{z}_\psi}\right) \\ \sqrt{x_{\boldsymbol{z}}^2+y_{\boldsymbol{z}}^2}\,\sin\left({}^{P}\boldsymbol{n}_{\boldsymbol{z}_\phi}\right) \\ \sqrt{x_{\boldsymbol{z}}^2+y_{\boldsymbol{z}}^2+z_{\boldsymbol{z}}^2}\,\cos\left({}^{P}\boldsymbol{n}_{\boldsymbol{z}_\phi}\right)\sin\left({}^{P}\boldsymbol{n}_{\boldsymbol{z}_\psi}\right) \end{pmatrix}. \tag{4.109}$$

and ${}^{P}\boldsymbol{n}_{\boldsymbol{z}} = \begin{pmatrix} {}^{P}\boldsymbol{n}_{\boldsymbol{z}_r} & {}^{P}\boldsymbol{n}_{\boldsymbol{z}_\phi} & {}^{P}\boldsymbol{n}_{\boldsymbol{z}_\psi} \end{pmatrix}^{T}$.

For a derivation please refer to [203].

4.4.4 The Filter-Steps in a Nutshell

In the following we summarize our Kalman fusion, which is in general described in Section 2.7:

1. compute the LOS-mapping $\boldsymbol{M}_{\boldsymbol{z}_{k+1}}$ according to Eq. 4.104
2. compute the residual $\mathring{\boldsymbol{r}}_{k+1} = \mathring{\boldsymbol{z}}_{k+1} - \mathring{\boldsymbol{H}}_{k+1}\boldsymbol{x}_{k+1}^{-}$
3. compute the error state covariance prior $\boldsymbol{P}_{k+1}^{-}$ according to Eq. 2.19
4. compute the innovation $\mathring{\boldsymbol{S}}_{k+1} = \mathring{\boldsymbol{H}}_{k+1}\boldsymbol{P}_{k+1}^{-}\mathring{\boldsymbol{H}}_{k+1}^{T} + \mathring{\boldsymbol{L}}_{r,k}$
5. compute the Kalman gain $\mathring{\boldsymbol{K}}_{k+1} = \boldsymbol{P}_{k+1}^{-}\mathring{\boldsymbol{H}}_{k+1}^{T}\mathring{\boldsymbol{S}}_{k+1}^{-1}$
6. compute the error state covariance posterior $\boldsymbol{P}_{k+1}^{+}$ according to Eq. 2.23
7. compute the correction $\boldsymbol{\delta}_{\boldsymbol{x};k+1}^{+}$ according to Eq. 4.107
8. update the state by the following equations and reset the error to zero

$$\begin{aligned} \boldsymbol{t}_{k+1} &= \boldsymbol{t}_k - \boldsymbol{\delta}_{\boldsymbol{t};k+1} \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k - \boldsymbol{\delta}_{\boldsymbol{v};k+1} \\ \boldsymbol{q}_{k+1} &= \begin{pmatrix} \boldsymbol{\delta}_{\tilde{\boldsymbol{q}};k+1} \\ 1 \end{pmatrix} \odot \boldsymbol{q}_k \\ \boldsymbol{b}_{\boldsymbol{a};k+1} &= \boldsymbol{b}_{\boldsymbol{a};k} - \boldsymbol{\delta}_{\boldsymbol{b}_{\boldsymbol{a}};k+1} \\ \boldsymbol{b}_{\boldsymbol{\omega};k+1} &= \boldsymbol{b}_{\boldsymbol{\omega};k} - \boldsymbol{\delta}_{\boldsymbol{b}_{\boldsymbol{\omega}};k+1} \end{aligned}$$

In Section 8.5.4 we will show some first experiments on synthetic data as proof of concept.

Chapter 5

Image-Based Pose Estimation

In this section, we will introduce two image-based pose estimation techniques. The Z_∞-algorithm is a biological motivated method which estimates the pose based on optical flow. We mentioned in Section 1.1 that insects are observed to behaviorally separate rotation and translation, stabilizing their head during motion so that they only measure translational optical flow. It is difficult to realize such a stabilization on technical systems, but, nonetheless, we use features with certain projection properties to compensate for the rotational motion and, thus, allow for a pure translation estimation. The other approach which we present is a keyframe-based VSLAM technique. It estimates the pose based on the 3D structure of a landmark set, which has to be initialized.

Both approaches work for arbitrary goniometers – the methods are only based on the angular direction of landmarks. However, in our work we focus on inertial and visual sensors and, hence, we will discuss the algorithms in the context of cameras.

5.1 Z_∞-Algorithm – Efficient Monocular Motion Estimation

One problem in estimating an arbitrary motion in 3D from noisy sensor data is to quantify the error and to suppress wrong measurements that deteriorate the result. Most known approaches, like the eight-point algorithm introduced in Section 3.3.1, compute all degrees of freedom simultaneously. An error in one dimension can be compensated by the other dimensions, which makes it difficult to weight or isolate bad measurements from the data set. Erroneous data can be detected and rejected more effectively if less parameters are estimated at once. Thus, only the parameters which belong together should be estimated simultaneously. The effects of a combined and a separated estimation of translation and rotation are depicted in Section 8.2.1.

The Z_∞-algorithm uses the usually undesirable quantization effects of the camera projection to separate translation-invariant from translation-dependent landmarks in the image. In fact, we determine the rotational component of the present motion without ever considering the translational one. Fast closed form solutions exist, which provide an efficient and globally optimal rotation and translation computation from optical flow without any a-priori information. Hence, the separation of the rotation estimation from the translation simplifies the computation and the suppression of error prone data drastically, which are the major advantages of the Z_∞-algorithm.

The key problem is to identify translation-invariant landmarks for rotation estimation. The remaining features can then be used to compute the translation.

5.1.1 Z_∞-Principle

Talking about the projective transformation and the pixel discretization of digital cameras, these characteristics are usually considered to be the accuracy barriers for image based motion estimation. However, splitting the motion in a rotational and translational part is based on just these effects - Z_∞uses the projective transformation and pixel discretization to split the tracked features into a translation-invariant and a translation-dependent set.

We see from the basic camera projection equation (Eq. 2.9) that the X- and Y-axis offsets of the 3D world coordinates, $x_{\boldsymbol{P}}$ and $y_{\boldsymbol{P}}$, are both divided by the landmark's distance $z_{\boldsymbol{P}}$. Hence, according to the motion formula (Eq. 2.3), the landmarks mapped onto the image plane move as described in the following.

Each image feature of a calibrated camera can be interpreted as a ray from the optical center to the 3D landmark intersecting the image plane.

$$\boldsymbol{P}_i \;=\; \lambda_i \bar{\boldsymbol{r}}_i \quad \text{with} \quad \bar{\boldsymbol{r}} \;=\; \frac{\boldsymbol{r}}{\|\boldsymbol{r}\|} \;. \tag{5.1}$$

The length λ_i of the normalized vector $\bar{\boldsymbol{r}}_i$ to the i-th landmark $\boldsymbol{P}_i$ is unknown by the camera projection. While the rotation of a feature is not affected by the distance of the respective landmark, this is not the case for translations. A translational motion of the camera results in a different motion in the image for each landmark depending on its distance to the camera. The motion transformation of a landmark can be described by

$$\lambda_i' \, \bar{\boldsymbol{r}}_i' \;=\; \boldsymbol{R}^T \, (\lambda_i \bar{\boldsymbol{r}}_i - \boldsymbol{t}) \;. \tag{5.2}$$

Due to the pixel discretization, the translation cannot be measured anymore if a landmark exceeds a certain distance to the camera. This distance can be computed as

Table 5.1: Concrete examples illustrating the relation between the distance threshold z_∞ and the camera and motion parameters. The upper values denote the focal length in pixel size with 900 px ($\sim 37°$ aperture angle) being a quite conventional industrial camera (*e.g.*, with $f = 8.6\,\text{mm}$ and $s = 9.6\,\mu\text{m}$) going down to 300 px ($\sim 90°$ aperture angle) representing a wide angle camera and 100 px ($\sim 143°$ aperture angle) a fisheye lense. The left values represent the measurable motion per frame, where r is the frame-rate of the camera. Please note again that the motion in the direction of the landmark does not affect z_∞. A measured motion of $0.5\,\frac{\text{m}}{\text{fr}}$ is equal to a translational motion perpendicular to the feature ray of $27\,\frac{\text{km}}{\text{h}}$ at a frame-rate of only 15 Hz, while $0.005\,\frac{\text{m}}{\text{fr}}$ would still correspond to a perpendicular motion component of $2.16\,\frac{\text{km}}{\text{h}}$ at 120 Hz.

$\frac{f}{s}$ [px] / $\max(t_m) \cdot r \left[\frac{\text{m}}{\text{fr}}\right]$	900	600	300	100
0.5	450.0 m	300.0 m	150.0 m	50.0 m
0.1	90.0 m	60.0 m	30.0 m	10.0 m
0.01	9.0 m	6.0 m	3.0 m	1.0 m
0.005	4.5 m	3.0 m	1.5 m	0.5 m

follows

$$z_\infty = \frac{f}{s_m}\, t_m \quad \text{with} \quad m \in \{x, y\} \ . \tag{5.3}$$

Hence, z_∞ depends on the ratio between the focal length f and the pixel size s_x and s_y and the translational components parallel to the camera plane, t_x and t_y, respectively. A small lateral translational component, a large aperture angle, low resolution, or a high frame-rate reduce the distance threshold for translation invariant landmarks. Table 5.1 and Fig. 5.1 illustrate the relation of these factors and the translation invariant distance by concrete examples. If the camera is not mounted parallel to the main direction of motion, the measurable component of translational motion becomes significantly smaller then the values assumed in the examples. As illustrated in Fig. 5.2, the observed translational component is shortened by two angles

$$\begin{aligned} \Delta \boldsymbol{t}_{s_m} &= \cos(\psi_m)\, \Delta \boldsymbol{t} \\ \Delta \boldsymbol{t}_m &= \frac{\boldsymbol{t}_{s_m}}{\cos(\varphi_m)} = \frac{\cos(\psi_m)}{\cos(\varphi_m)}\, \Delta \boldsymbol{t} \quad \text{with} \quad m \in \{x, y\} \ , \end{aligned} \tag{5.4}$$

whereas the angle ψ is between the direction of translation and the plane perpendicular to the projection ray and the angle φ is between the optical axis and the projection ray. Hence, all the points which lie exactly in the direction of translation become translation-invariant independent of their distance. The more off the points are from this axis, the more translational sensitive they get.

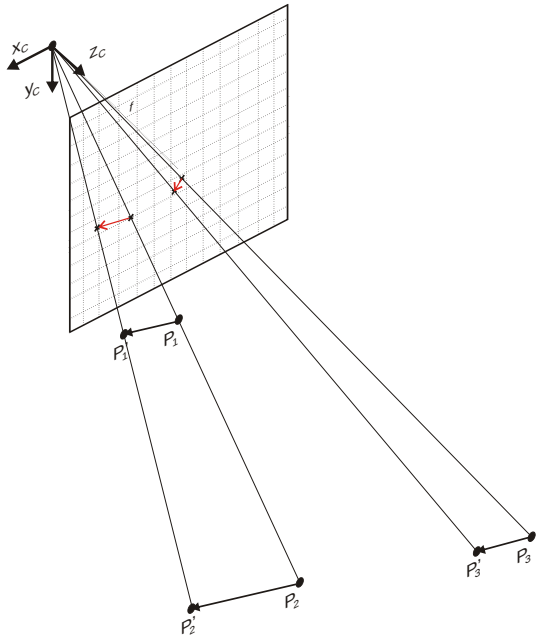


Figure 5.1: Optical flow vector 1 ($\boldsymbol{P}_1, \boldsymbol{P}_1'$) is shorter than vector 2 but it is closer to the optical center. Therefore, both vectors result in the same projection. Vector 3 has the same length as vector 1, but it is farther from the image plane, hence, the projection is much shorter. It is so small, that the projection of $\boldsymbol{P}_3$ and $\boldsymbol{P}_3'$ lies within the same pixel. If such a projection results only from a translation then this feature would be translation-invariant, because the translation does not yield a measurable motion.

We have shown that image features can be split into translation-invariant and translation-dependent features, depending on their distance to the camera. Indoor scenes do, in general, not provide enough depth range to allow for translation invariant landmarks. Only if really slow translations are applied to the camera a short distance to the landmarks is sufficient. Experiments have shown, that in outdoor pictures the contrast of the sky to the ground at the horizon generates many good features to track. Indeed, an average percentage of 60 % of the selected features lies at the horizon. But how can we identify which feature corresponds to which set? We have no information about the rotation, the translation or the distance of the landmarks visible in the image. The solution is provided by the algorithm described in the next section.

5.1.2 RANSAC Revisited

The general approach of the *Random Sampling Consensus* (RANSAC) algorithm is described in Section 2.5. In our special case, the estimated model is a rotation matrix for which the entries have to be calculated. Therefore, we take a set of three random

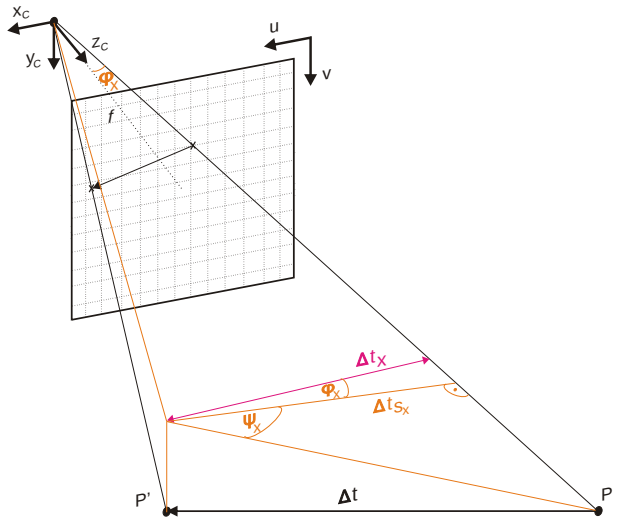


Figure 5.2: This drawing visualizes how the measurable translational component $\Delta \boldsymbol{t}_x$ can be calculated. In general the optical flow vectors become longer the more parallel the translation to the image plane is and the closer the features are to the image border (due to a higher angular resolution). The orange auxiliary line illustrates the projections onto the X-Z plane.

optical flow vectors $\mathcal{R}_{\text{rand}}$ and we estimate the rotation matrix based on them $\boldsymbol{R} = \text{R}(\mathcal{R}_{\text{rand}})$, as it will be explained in Section 5.1.3. The estimated rotation is applied to the remaining data set. In case that the initial three correspondences were from points in the world at a distance further than z_∞ and so represent only the rotational part, the true rotation is estimated. Otherwise, the influence of the translational component of the optical flow vectors results in a wrong rotation matrix. Applying the computed rotation $\boldsymbol{R}$ on the remaining data set shows whether a minimum percentage $\Xi_{\#\boldsymbol{R}}$ of the optical flow vectors, representing the residual translation-invariant features, fit the estimate up to a specified threshold $\Xi_{\boldsymbol{R}}$. This proofs that the initial points were truly translational invariant:

$$\mathcal{R}_{\text{inl}} = \left\{ \left(\boldsymbol{r}_i, \boldsymbol{r}'_i \right) : \left\| \bar{\boldsymbol{r}}_i - \boldsymbol{R} \bar{\boldsymbol{r}}'_i \right\| < \Xi_{\boldsymbol{R}} \right\} . \tag{5.5}$$

Based on whether the number of elements in $\mathcal{R}_{\text{inl}}$ is sufficient

$$\frac{|\mathcal{R}_{\text{inl}}|}{N} \geqslant \Xi_{\#\boldsymbol{R}} , \tag{5.6}$$

one has to choose a new landmark set for another rotation estimation iteration or a more accurate rotation can be computed using all identified translation-invariant inliers $\boldsymbol{R} = \text{R}(\mathcal{R}_{\text{inl}})$.

Once the rotation has been successfully calculated, the back-rotated outlier set consists of the translation vectors $\mathcal{T}_{\text{inl}}$ and the measurement outliers $\mathcal{S}_{\text{outl}}$

$$\mathcal{R}_{\text{outl}} = \{\mathcal{T}_{\text{inl}}, \mathcal{S}_{\text{outl}}\} \ . \tag{5.7}$$

Again we use RANSAC to robustly estimate the translation $\mathcal{T}_{\text{rand}}$ as described in Section 5.1.4 and to identify wrong measurements. The deviation of the vectors from the estimated epipolar line and the percentage of found inliers provides an accuracy measure and allows to compare the RANSAC iterations.

The probability to find an initial set of three translation-invariant optical flow elements depends on the percentage of such vectors in the data set. As described above, an evaluation of real outdoor images has shown, that an average of 60 % of the features are at the horizon and, thus, far enough to be translation-invariant. The lower 5 % quantile was about 30 %. This results in approximately 37 iterations necessary to assure that in 95 % of the cases RANSAC can determine the rotation and divide the data set. Usually, several rotation inliers can also be tracked in the next image. Such features are then preferred for rotation estimation, because it can be assumed that the corresponding landmarks are still further away than the z_∞ threshold. Thus, the number of RANSAC iterations for rotation estimation is reduced to one - in practice few iterations are required to ensure a robust result.

At this point the trade-off one has to find applying this concept should be mentioned. On one hand we are looking for a camera setup which reduces the measurable translational component of landmarks to allow for translation-invariant points, but on the other hand, if we do not measure the translation, we can not or only poorly estimate it. Furthermore, an erroneous rotation estimation due to translation influence yields a bad translation estimation. Hence, the crucial part of this algorithm is a proper and reliable identification of translation-invariant landmarks.

5.1.3 Rotation Estimation

The rotation of a point cloud can be estimated in closed form based on the direction vectors to the different landmarks, *e.g.*, based on quaternions [212] or discrete cosine matrices using an eigenvalue decomposition [157] or a singular value decomposition [213, 214] (see also Section 3.3.2). We use the so called Umeyama algorithm described in [214], which is an extension of the Arun's algorithm [213], fixing a possible ambiguity in the result.

The corresponding points used for rotation estimation belong all to translation-invariant points. Hence, Eq. 5.2 can be abbreviated to

$$\boldsymbol{P}_i' \;=\; \lambda_i' \, \bar{\boldsymbol{r}}_i' \;=\; \boldsymbol{R}^T \boldsymbol{P}_i \;=\; \boldsymbol{R}^T \, \lambda_i \, \bar{\boldsymbol{r}}_i \;=\; \lambda_i \, \boldsymbol{R}^T \, \bar{\boldsymbol{r}}_i \qquad \Rightarrow \qquad \bar{\boldsymbol{r}}_i' \;=\; \boldsymbol{R}^T \, \bar{\boldsymbol{r}}_i \; . \tag{5.8}$$

Thus, we need to solve the following least squares problem

$$\mathrm{R}(\mathcal{S}) \;:\quad \boldsymbol{R} \;=\; \arg\min_{\boldsymbol{R}} \left(\sum_{(\bar{\boldsymbol{r}}_i, \bar{\boldsymbol{r}}_i') \in \mathcal{S}} \left\| \boldsymbol{R}^T \, \bar{\boldsymbol{r}}_i - \bar{\boldsymbol{r}}_i' \right\| \right) . \tag{5.9}$$

In the following, we assume that we have a set of corresponding rays $\{(\boldsymbol{r}_i, \boldsymbol{r}_i')\}$, which are only rotated and whose rotation we want to estimate. First, the origin of the coordinate frame has to be moved to the center of the point cloud:

$$\begin{aligned} \boldsymbol{c} \;&=\; \frac{1}{|\mathcal{S}|} \sum_{\bar{\boldsymbol{r}}_i \in \mathcal{S}} \bar{\boldsymbol{r}}_i & \qquad \boldsymbol{c}' \;&=\; \frac{1}{|\mathcal{S}|} \sum_{\bar{\boldsymbol{r}}_i' \in \mathcal{S}} \bar{\boldsymbol{r}}_i' \\ \boldsymbol{r}_i^* \;&=\; \bar{\boldsymbol{r}}_i - \boldsymbol{c} & \qquad \boldsymbol{r}_i'^* \;&=\; \bar{\boldsymbol{r}}_i' - \boldsymbol{c}' \end{aligned} \tag{5.10}$$

Now, the non scaled sample cross-covariance matrix for these point clouds is calculated from the set $\mathcal{S}^* = \{(\boldsymbol{r}_i^*, \boldsymbol{r}_i'^*)\}$, such that

$$\boldsymbol{M} = \sum_{(\boldsymbol{r}_i^*, \boldsymbol{r}_i'^*) \in \mathcal{S}^*} \boldsymbol{r}_i'^* \, \boldsymbol{r}_i^{*T} \; . \tag{5.11}$$

Therefore, $\frac{1}{|\mathcal{S}|}\boldsymbol{M}$ is the sample cross-covariance matrix between $\{\bar{\boldsymbol{r}}_i\}$ and $\{\bar{\boldsymbol{r}}_i'\}$. It can be shown that the rotation matrix which minimizes Eq. 5.9 also fulfills

$$\boldsymbol{R} \;=\; \arg\max_{\boldsymbol{R}} \left(\mathrm{tr}(\boldsymbol{R}\boldsymbol{M}) \right) \; . \tag{5.12}$$

Consequently, $\boldsymbol{R}$ can be calculated as

$$\boldsymbol{R} \;=\; \boldsymbol{V}_{\boldsymbol{M}} \boldsymbol{W} \boldsymbol{U}_{\boldsymbol{M}}^T \quad \text{with} \quad \mathrm{SVD}\left(\boldsymbol{M}\right) \;=\; \boldsymbol{U}_{\boldsymbol{M}} \boldsymbol{\Sigma}_{\boldsymbol{M}} \boldsymbol{V}_{\boldsymbol{M}}^T \tag{5.13}$$

with

$$\boldsymbol{W} \;=\; \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det\left(\boldsymbol{V}_{\boldsymbol{M}} \boldsymbol{U}_{\boldsymbol{M}}^T\right) \end{pmatrix} \; . \tag{5.14}$$

The uncertainty measure of the rotation estimation consists of the average rotation error and the percentage of rotation inliers in the data set. We propose the following

formula to compute the quality measure $q_{\boldsymbol{R}}$:

$$q_{\boldsymbol{R}} = \frac{\sum_{\bar{\boldsymbol{r}}_i \in \mathcal{R}_{\text{inl}}} \left\| \boldsymbol{R}^T \bar{\boldsymbol{r}}_i - \bar{\boldsymbol{r}}'_i \right\|}{|\mathcal{R}_{\text{inl}}|^2} \,. \tag{5.15}$$

This measure results from the mean feature error weighted by the inverse of the inlier percentage

$$\frac{\sum_{\bar{\boldsymbol{r}}_i \in \mathcal{R}_{\text{inl}}} \left\| \boldsymbol{R}^T \bar{\boldsymbol{r}}_i - \bar{\boldsymbol{r}}'_i \right\|}{|\mathcal{R}_{\text{inl}}|} \frac{N}{|\mathcal{R}_{\text{inl}}|}$$

with N being the overall number of feature correspondences, but which is constant and, thus, can be neglected for the quality measure. The weighting term is steep for small inlier percentages and flattens out for large ones. Hence, the number of inliers is crucial if only a small percentage of inliers could be found and becomes insignificant for large inlier sets.

5.1.4 Translation Estimation

All the features in $\mathcal{R}_{\text{outl}}$ are rotated back by $\boldsymbol{R}$ and consist afterwards only of the translational part of the motion

$$\mathring{\boldsymbol{r}}' = \boldsymbol{R}\,\boldsymbol{r}' = \boldsymbol{R}\,\boldsymbol{R}^T(\boldsymbol{r} - \boldsymbol{t}) = \boldsymbol{r} - \boldsymbol{t} \tag{5.16}$$

with $\boldsymbol{t} \overset{!}{\neq} \boldsymbol{0}$ because only translation-dependent features are used. Projecting the rays on the unit focal image plane we get

$$\check{\boldsymbol{r}}' = \frac{\mathring{\boldsymbol{r}}'}{z_{\mathring{\boldsymbol{r}}'}} \,. \tag{5.17}$$

The back-rotated optical flow vectors $\{(\boldsymbol{r}_i \check{\boldsymbol{r}}'_i)\}$ should all meet in the epipole, which means that the epipole should be the intersection of all the lines defined by the translational flow vectors. This constrained defines a linear system of equations. Let

$$\boldsymbol{n}_i = \begin{pmatrix} y_{\check{\boldsymbol{r}}'_i} - y_{\boldsymbol{r}_i} \\ x_{\boldsymbol{r}_i} - x_{\check{\boldsymbol{r}}'_i} \end{pmatrix} \tag{5.18}$$

be the normal of the translational flow vector, then the scalar product d_i of $\boldsymbol{n}_i$ and an arbitrary point on the line needs to be the same, where d_i can be computed as follows:

$$d_i = \boldsymbol{n}_i^T \boldsymbol{r}_{i;1:2} \,. \tag{5.19}$$

Thus, the linear system of equations for the epipole $\boldsymbol{e}$ looks like

$$\boldsymbol{A}\,\boldsymbol{e} = \begin{pmatrix} \boldsymbol{n}_1^T \\ \boldsymbol{n}_2^T \\ \vdots \\ \boldsymbol{n}_{|\mathcal{R}_{\text{out}}|}^T \end{pmatrix} \begin{pmatrix} x_{\boldsymbol{e}} \\ y_{\boldsymbol{e}} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{|\mathcal{R}_{\text{out}}|} \end{pmatrix} \stackrel{def}{=} \boldsymbol{d}\,. \tag{5.20}$$

This linear system could then be solved by using the Moore-Pennrose pseudoinverse $\boldsymbol{A}^+$ of $\boldsymbol{A}$, which leads to poor results, if $\boldsymbol{A}$ is close to be singular in case of a motion parallel to the image plane. Therefore, we use the SVD to compute the solution. Let the linear system be

$$\boldsymbol{A}\,\boldsymbol{e} = \boldsymbol{d} \quad \Rightarrow \quad \boldsymbol{A}\,\boldsymbol{e} - \boldsymbol{d} = \boldsymbol{0} \quad \Rightarrow \quad \begin{pmatrix} \boldsymbol{A} & \boldsymbol{d} \end{pmatrix} \begin{pmatrix} \boldsymbol{e} \\ -1 \end{pmatrix} = \boldsymbol{0} \stackrel{def}{=} \boldsymbol{B}\,\boldsymbol{g}\,. \tag{5.21}$$

Thus, the direction of translation up to sign $\tilde{\boldsymbol{t}}$ can be computed by

$$\tilde{\boldsymbol{t}} = \frac{\tilde{\boldsymbol{t}}'}{\left\|\tilde{\boldsymbol{t}}'\right\|} \quad \text{with} \quad \tilde{\boldsymbol{t}}' = \begin{pmatrix} \boldsymbol{e} \\ 1 \end{pmatrix} \quad , \quad \boldsymbol{e} = -\frac{\boldsymbol{V}_{\boldsymbol{B};1:2,3}}{\boldsymbol{V}_{\boldsymbol{B};3,3}} \quad \text{and} \quad \text{SVD}\left(\boldsymbol{B}\right) = \boldsymbol{U}_{\boldsymbol{B}}\boldsymbol{\Sigma}_{\boldsymbol{B}}\boldsymbol{V}_{\boldsymbol{B}}^T\,. \tag{5.22}$$

Analogue to Eq. 3.29 the ambiguity of sign of the translation vector can be resolved by following formula:

$$\boldsymbol{t} = \begin{cases} -\tilde{\boldsymbol{t}} & \text{if } \sum_i \left(\boldsymbol{r}_i \times \check{\boldsymbol{r}}_i'\right)^T \left(\tilde{\boldsymbol{t}} \times \boldsymbol{r}_i\right) < 0 \\ \tilde{\boldsymbol{t}} & \text{else} \end{cases}\,. \tag{5.23}$$

As uncertainty measure for the translation, we use the Euclidean distance between the calculated epipole and the optical flow vectors $\|\boldsymbol{A}\,\boldsymbol{e} - \boldsymbol{d}\|$ together with the percentage of translation inliers, which are defined by the threshold $\Xi_{\boldsymbol{t}}$ and

$$\mathcal{T}_{\text{inl}} = \left\{\left(\boldsymbol{r}_i, \boldsymbol{r}_i'\right) : \left\|\boldsymbol{n}_i^T\boldsymbol{e} - d_i\right\| < \Xi_{\boldsymbol{t}}\right\}\,. \tag{5.24}$$

Similar to the quality measure for the rotation estimation, as denoted in Eq. 5.15, we propose following quality measure $q_{\boldsymbol{t}}$ for the translation estimation

$$q_{\boldsymbol{t}} = \frac{\sum_{\boldsymbol{r}_i \in \mathcal{T}_{\text{inl}}} \left\|\boldsymbol{n}_i^T\boldsymbol{e} - d_i\right\|}{|\mathcal{T}_{\text{inl}}|^2}\,. \tag{5.25}$$

5.2 Keyframe Based VSLAM

Thinking of applications where not only efficiency is required but also high accuracy, we need to estimate and model also the depth of the features which we use for pose estimation. In that way, we can improve the accuracy of image-based pose estimation further, but it requires an initialization phase. Other keyframe based algorithms, *e.g.*, PTAM, require an initialization step with a long translational motion in front of a planar surface to initialize the detected features by planar homography decomposition [108]. In the following we introduce a keyframe based VSLAM approach based on a modified version of the VGPS algorithm as introduced in Section 3.3.2.

In this algorithm we treat the feature initialization sequentially. The algorithm tracks the features in one camera, but uses a second one to initialize the features of every new keyframe by sub-pixel stereo triangulation as described in Section 7.1.1. This initialization step runs in parallel and yields highly accurate feature-depths. Hence, we do not require an initialization phase when setting up the system, nor a bundle adjustment to preserve the congruency of the keyframes' scales.

Independent of the tracker, its result can contain bad features due to occlusions, repetitive patterns, virtual features and reflections. Moreover, the stereo initialization can provide false matches resulting in an incorrect depth of the respective points. Occlusions occur when background objects are not present in the image anymore, because they are overlaid by an object in the foreground. In the opposed case, such background objects can also suddenly appear in the image at the edge of a foreground object and avoid proper tracking. Another problem are virtual features. These do not correspond to real points in the world, but appear at the projection edges of a foreground and a background object. Such features do not fit the rigid body motion corresponding to a certain 3D point, but follow an arbitrary, misleading motion model. The same applies to reflections. Almost none surface has only lambertian reflection, but also a specular part. Such reflections have a large intensity gradient and are, therefore, likely selected for tracking. However, the location of the reflection depends not only on the position of the object, but also on the location of the light source. This leads to a moving feature with misleading motion model.

Within the tracking layer such problematic cases can not be detected. Therefore, the localization algorithm has to be able to detect bad features and reject them from processing and future tracking.

5.2.1 Robustified VGPS

We rely on a highly accurate feature initialization and do not want to modify the estimated depths by some bundle adjustment approach as seen in Section 3.3.3. Hence, we use VGPS (see Section 3.3.2) for motion estimation, which does not change the depth estimates. Due to its iterative character, *maximum likelihood-type* estimators (M-estimators) suggest themselves to be used for outlier rejection. The M-estimators are a generalization of maximum likelihood (ML) methods. They are more robust to outliers than ML-methods due to a special weighting. In most cases, an iteratively re-weighted least squares fitting method is used for the M-estimation. The advantage by using M-estimators and not a simple outlier rejection method like RANSAC is that each measurement can be weighted according to its accuracy contribution.

The redescending M-estimator is applied on the residual Euclidean distances between matched and reprojected points. This is done to disregard gross outliers without compromising the estimation convergence because of the naturally noisy nature of the problem. Such outliers may correspond to either a faulty internal 3D model, false matching correspondences, virtual features (*e.g.* features from occlusions), or, more infrequently, to wrong tentative ranges and are fatal to unbiased pose estimation. Therefore, in case of outliers, the robustified VGPS not only disregards this data but also sends a signal to the features' database in order to remove those landmarks. In particular, we use the biweight function of Tukey because of its continuous derivatives and its handy weights [215]. Both, the initial estimation and the chosen scale, are much more influential parameters to global fast convergence than the nature of the employed function. Thus, the modification consists in weighting the contribution of each point to the inertia matrix of the matched set of points with the weight

$$w_i = \begin{cases} \left(1 - \dfrac{\boldsymbol{e}_i^T \boldsymbol{e}_i}{s^2}\right)^2 & \text{if } \|\boldsymbol{e}_i\| \leq s \\ 0 & \text{else} \end{cases}, \tag{5.26}$$

where $\boldsymbol{e}_i = \boldsymbol{P}_i - \boldsymbol{R}\widehat{P}'_i - \boldsymbol{t}$ is the estimated normalized matching residual for object point i and s is the scale of the inlier noise. This yields the following modification of Eq. 3.31

$$\boldsymbol{R} = \arg\max_{\boldsymbol{R}} \left(\mathrm{tr}\left(\boldsymbol{R}^T \boldsymbol{M}\right)\right) \quad \text{with} \quad \boldsymbol{M} = \sum_{i=1}^{N} w_i \, \widehat{\boldsymbol{P}}_i'^{*} \, \boldsymbol{P}_i^{*}\,. \tag{5.27}$$

Finally, we use an efficient termination policy determined by a threshold on the absolute orientation correction over the course of the iterations.

Experiments in Section 8.2.3 compare RVGPS to the conventional VGPS algorithm and illustrate the improvements by using M-estimators.

Chapter 6

Uncertainty Prediction for Image-Based Motion Estimation

To allow for a proper fusion of different sensors, it is crucial to know the respective accuracy of each measurement. By using optical flow based motion estimation techniques the accuracy of the estimate depends on the tracked feature set and the applied algorithm. While there is no error propagation for the Z_∞-algorithm known in the literature, it has already been derived for the regular, original eight-point algorithm [150]. Applying those formulas one obtains the uncertainties for the rotation quaternion and the translation estimation. However, it does not cover the modifications introduced in Section 3.3.1, which make the algorithm more stable. Furthermore, the error propagation for the rotation matrix estimation, which is more robust than the quaternion computation, has also not been discussed in the literature. In this chapter, we will tackle this missing points and derive a first order error propagation for all the discussed eight-point algorithm variants and also the Z_∞-algorithm, as described in Section 5.1. With these equations we can approximate the expected error in the computed translation vector and rotation matrix, given a set of feature correspondences and the provided tracking accuracy.

6.1 Error Propagation of the Eight-Point Algorithm

In the following, we will present a first order perturbation analysis of the eight-point algorithm, adapted from [150]. First, we will introduce the notation of errors. Let us assume a measurement matrix $\tilde{\boldsymbol{A}}$, consisting of the matrix $\boldsymbol{A}$ denoting the real values

and the perturbation matrix $\boldsymbol{\Delta_A}$, such that

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{\Delta_A} . \tag{6.1}$$

The following theorem will be used in several steps of the derivation.

Theorem: Let $\boldsymbol{A}$ be an $n \times n$ symmetric matrix with the eigenvalues $\lambda_1, \ldots, \lambda_n$ and $\boldsymbol{H}$ be an orthonormal matrix consisting of the eigenvectors $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n$, such that

$$\boldsymbol{A} = \boldsymbol{H} \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \, \boldsymbol{H}^T . \tag{6.2}$$

$\boldsymbol{x}$ is then a vector in the span of an eigenvector $\boldsymbol{h}_j$ with scaling factor $k \in \mathbb{R}$, resulting in

$$\boldsymbol{x} = k\boldsymbol{h}_j . \tag{6.3}$$

The perturbation of $\boldsymbol{x}$ can then be described up to first order by the perturbation in $\boldsymbol{A}$, denoted as $\boldsymbol{\Delta_A}$, according to

$$\delta_{\boldsymbol{x}} \cong \boldsymbol{H}\boldsymbol{D}_i\boldsymbol{H}^T\boldsymbol{\Delta_A}\boldsymbol{x} , \tag{6.4}$$

with

$$\boldsymbol{D}_i = \operatorname{diag}(l_1, \ldots, l_n) \quad \text{and} \quad l_j = \begin{cases} 0 & \text{if } i = j \\ \dfrac{1}{\lambda_i - \lambda_j} & \text{else} \end{cases} . \tag{6.5}$$

Proof: See [150].

The starting point of the derivation is the error $\boldsymbol{\delta_A}$ of the matrix $\boldsymbol{A}$ of Eq. 3.5, which can be linearly approximated by

$$\boldsymbol{\delta}_{\boldsymbol{A}^T} = \left[\boldsymbol{\Delta}_{\boldsymbol{A}}^T\right]_{\text{col}} \text{ with } \boldsymbol{\Delta}_{\boldsymbol{A}}^T \cong \begin{pmatrix} \delta_{u_1}u'_1 + \delta_{u'_1}u_1 & \delta_{u_2}u'_2 + \delta_{u'_2}u_2 & \ldots & \delta_{u_N}u'_N + \delta_{u'_N}u_N \\ \delta_{u_1}v'_1 + \delta_{v'_1}u_1 & \delta_{u_2}v'_2 + \delta_{v'_2}u_2 & \ldots & \delta_{u_N}v'_N + \delta_{v'_N}u_N \\ \delta_{u_1} & \delta_{u_2} & \ldots & \delta_{u_N} \\ \delta_{v_1}u'_1 + \delta_{u'_1}v_1 & \delta_{v_2}u'_2 + \delta_{u'_2}v_2 & \ldots & \delta_{v_N}u'_N + \delta_{u'_N}v_N \\ \delta_{v_1}v'_1 + \delta_{v'_1}v_1 & \delta_{v_2}v'_2 + \delta_{v'_2}v_2 & \ldots & \delta_{v_N}v'_N + \delta_{v'_N}v_N \\ \delta_{v_1} & \delta_{v_2} & \ldots & \delta_{v_N} \\ \delta_{u'_1} & \delta_{u'_2} & \ldots & \delta_{u'_N} \\ \delta_{v'_1} & \delta_{v'_2} & \ldots & \delta_{v'_N} \\ 0 & 0 & \ldots & 0 \end{pmatrix} \tag{6.6}$$

where $u_i = u_{\boldsymbol{r}_i}$, $u'_i = u_{\boldsymbol{r}'_i}$ and $v_i = v_{\boldsymbol{r}_i}$, $v'_i = v_{\boldsymbol{r}'_i}$ for the sake of clarity. We use the transposed matrix $\boldsymbol{A}^T$ because it eases the computation of the covariance matrix and the understanding of its shape.

Our aim is to compute the variance of the estimated $\boldsymbol{t}$ and $\boldsymbol{R}$. Hence, we have to compute the linear transitions of each step of the eight-point algorithm, as explained in Section 3.3.1.

6.1.1 Error Propagation for the Essential Matrix

According to Eq. 3.8 the entries of the Essential matrix correspond to the ninth eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$, which agrees with the smallest eigenvalue. Thus, we first have to compute the error $\boldsymbol{\delta}_{\boldsymbol{A}^T\boldsymbol{A}}$ from $\boldsymbol{\delta}_{\boldsymbol{A}^T}$. Using first order approximation we get

$$\begin{aligned} \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{\Delta}_{\boldsymbol{A}^T\boldsymbol{A}} &= (\boldsymbol{A} + \boldsymbol{\Delta}_{\boldsymbol{A}})^T (\boldsymbol{A} + \boldsymbol{\Delta}_{\boldsymbol{A}}) = \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{\Delta}_{\boldsymbol{A}} + \boldsymbol{\Delta}_{\boldsymbol{A}}^T\boldsymbol{A} + \boldsymbol{\Delta}_{\boldsymbol{A}}^T\boldsymbol{\Delta}_{\boldsymbol{A}} \\ \Rightarrow \boldsymbol{\Delta}_{\boldsymbol{A}^T\boldsymbol{A}} &\cong \boldsymbol{A}^T\boldsymbol{\Delta}_{\boldsymbol{A}} + \boldsymbol{\Delta}_{\boldsymbol{A}}^T\boldsymbol{A} \end{aligned} \tag{6.7}$$

which corresponds to

$$\boldsymbol{\delta}_{\boldsymbol{A}^T\boldsymbol{A}} \cong \left(\boldsymbol{A}^T \otimes \boldsymbol{I}_9 + \text{row}\left[\boldsymbol{I}_9 \otimes (\boldsymbol{A}_{i,-})^T\right]_{i=1}^{N}\right) \boldsymbol{\delta}_{\boldsymbol{A}^T} \overset{def}{=} \boldsymbol{G}_{\boldsymbol{A}^T\boldsymbol{A}}\, \boldsymbol{\delta}_{\boldsymbol{A}^T} \,. \tag{6.8}$$

The pertubation of the ninth eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$ can then be computed by the theorem introduced at the beginning of this chapter, such that

$$\begin{aligned} \boldsymbol{\delta}_{\hat{\boldsymbol{e}}} &\cong \boldsymbol{V}_{\boldsymbol{A}}\boldsymbol{D}_{\boldsymbol{A}^T\boldsymbol{A};9}\boldsymbol{V}_{\boldsymbol{A}}^T\boldsymbol{\Delta}_{\boldsymbol{A}^T\boldsymbol{A}}\,\hat{\boldsymbol{e}} \\ &= \boldsymbol{V}_{\boldsymbol{A}}\boldsymbol{D}_{\boldsymbol{A}^T\boldsymbol{A};9}\boldsymbol{V}_{\boldsymbol{A}}^T\left(\hat{\boldsymbol{e}}^T \otimes \boldsymbol{I}_9\right)\boldsymbol{\delta}_{\boldsymbol{A}^T\boldsymbol{A}} \\ &\overset{def}{=} \boldsymbol{G}_{\hat{\boldsymbol{E}}}\,\boldsymbol{\delta}_{\boldsymbol{A}^T\boldsymbol{A}} \end{aligned} \tag{6.9}$$

with $\text{SVD}(\boldsymbol{A}) = \boldsymbol{U}_{\boldsymbol{A}}\boldsymbol{\Sigma}_{\boldsymbol{A}}\boldsymbol{V}_{\boldsymbol{A}}^T$ because $\boldsymbol{V}_{\boldsymbol{A}}$ corresponds to the eigenvector matrix of $\boldsymbol{A}^T\boldsymbol{A}$ and the singular values of $\boldsymbol{A}$ are the square roots of the eigenvalues of $\boldsymbol{A}^T\boldsymbol{A}$: $\lambda_{\boldsymbol{A}^T\boldsymbol{A}} = \sigma_{\boldsymbol{A}}^2$.

As noted in the last equation, depending on the variant we do not get $\boldsymbol{e}$ but $\hat{\boldsymbol{e}}$ and, thus, we have to recover the Essential matrix according to Eq. 3.12 by

$$\begin{aligned} \boldsymbol{E} = \boldsymbol{S}'^T\left(\hat{\boldsymbol{E}} + \boldsymbol{\Delta}_{\hat{\boldsymbol{E}}}\right)\boldsymbol{S} \quad &\Rightarrow \boldsymbol{\Delta}_{\boldsymbol{E}} = \boldsymbol{S}'^T\boldsymbol{\Delta}_{\hat{\boldsymbol{E}}}\boldsymbol{S} \\ &\Rightarrow \boldsymbol{\delta}_{\boldsymbol{e}} = \left(S^T \otimes \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}\right) \circ \left(\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \otimes S'^T\right) \boldsymbol{\delta}_{\hat{\boldsymbol{e}}} \\ &\overset{def}{=} \boldsymbol{G}_{\boldsymbol{E}}\,\boldsymbol{\delta}_{\hat{\boldsymbol{e}}} \,. \end{aligned} \tag{6.10}$$

Now, that we know the propagation of the error from the input matrix $\boldsymbol{A}$ to the

Essential matrix, we can estimate the covariance matrix of $\boldsymbol{E}$ by following formula:

$$\boldsymbol{\Gamma}_{\boldsymbol{E}} = \boldsymbol{H}_{\boldsymbol{E}} \boldsymbol{\Gamma}_{\boldsymbol{A}^T} \boldsymbol{H}_{\boldsymbol{E}}^T \quad \text{with} \quad \boldsymbol{H}_{\boldsymbol{E}} = \boldsymbol{G}_{\boldsymbol{E}} \boldsymbol{G}_{\hat{\boldsymbol{E}}} \boldsymbol{G}_{\boldsymbol{A}^T \boldsymbol{A}} .$$

Assuming that the errors of different points and different components are uncorrelated and equally of magnitude σ, $\boldsymbol{\delta}_{\tilde{\boldsymbol{r}}} = \boldsymbol{\delta}_{\tilde{\boldsymbol{r}}'} = \begin{pmatrix} \sigma & \sigma & 0 \end{pmatrix}^T$, the covariance matrix $\boldsymbol{\Gamma}_{\tilde{\boldsymbol{A}}^T}$ can be computed based on $\boldsymbol{\Gamma}_x = \mathrm{E}\left(\delta_x \delta_x^T\right)$ from $\boldsymbol{\delta}_{\tilde{\boldsymbol{A}}^T}$. Hence, for the basic eight-point algorithm it has the form

$$\boldsymbol{\Gamma}_{\tilde{\boldsymbol{A}}^T} = \mathrm{Diag}\left(\tilde{\boldsymbol{P}}_1, \tilde{\boldsymbol{P}}_2, \ldots, \tilde{\boldsymbol{P}}_N\right)$$

with

$$\tilde{\boldsymbol{P}}_i = \left(\tilde{\boldsymbol{r}} \tilde{\boldsymbol{r}}^T\right) \otimes \boldsymbol{\Gamma}_{\tilde{\boldsymbol{r}}'} + \boldsymbol{\Gamma}_{\tilde{\boldsymbol{r}}} \otimes \left(\tilde{\boldsymbol{r}}' \tilde{\boldsymbol{r}}'^T\right) \quad \text{and} \quad \boldsymbol{\Gamma}_{\tilde{\boldsymbol{r}}} = \boldsymbol{\Gamma}_{\tilde{\boldsymbol{r}}'} = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} , \tag{6.11}$$

where σ denotes the standard variation of the feature detector and with $\tilde{\boldsymbol{r}}$ as introduced in Eq. 2.9. However, because of robustness issues we do not use $\tilde{\boldsymbol{r}}$ or $\boldsymbol{r}$ but $\hat{\boldsymbol{r}}$ which is a modification of $\boldsymbol{r}$ according to Eq. 3.11. Hence, we need to adapt the measurement variance for each point by

$$\begin{aligned} \hat{\boldsymbol{r}} &= \boldsymbol{S}\boldsymbol{r} \;\Rightarrow\; \boldsymbol{\delta}_{\hat{\boldsymbol{r}}} = \boldsymbol{S}\boldsymbol{\delta}_{\boldsymbol{r}} & \hat{\boldsymbol{r}}' &= \boldsymbol{S}'\boldsymbol{r}' \;\Rightarrow\; \boldsymbol{\delta}_{\hat{\boldsymbol{r}}'} = \boldsymbol{S}'\boldsymbol{\delta}_{\boldsymbol{r}'} \\ \boldsymbol{\Gamma}_{\hat{\boldsymbol{r}}} &= \left(\mathrm{diag}\left(\frac{1}{f}\boldsymbol{S}\boldsymbol{\delta}_{\tilde{\boldsymbol{r}}}\right)\right)^2 & \boldsymbol{\Gamma}_{\hat{\boldsymbol{r}}'} &= \left(\mathrm{diag}\left(\frac{1}{f}\boldsymbol{S}'\boldsymbol{\delta}_{\tilde{\boldsymbol{r}}'}\right)\right)^2 , \end{aligned} \tag{6.12}$$

which results in

$$\boldsymbol{\Gamma}_{\hat{\boldsymbol{A}}^T} = \mathrm{Diag}\left(\hat{\boldsymbol{P}}_1, \hat{\boldsymbol{P}}_2, \ldots, \hat{\boldsymbol{P}}_N\right) \quad \text{with} \quad \hat{\boldsymbol{P}}_i = \left(\hat{\boldsymbol{r}} \hat{\boldsymbol{r}}^T\right) \otimes \boldsymbol{\Gamma}_{\hat{\boldsymbol{r}}'} + \boldsymbol{\Gamma}_{\hat{\boldsymbol{r}}} \otimes \left(\hat{\boldsymbol{r}}' \hat{\boldsymbol{r}}'^T\right) . \tag{6.13}$$

Please note that if the robustification transformations $\boldsymbol{S}$ and $\boldsymbol{S}'$ are used and $\boldsymbol{E}$ is computed according to Eq. 3.12, the Frobenius norm of the Essential matrix is in general not two anymore, $\|\boldsymbol{E}\|_F \neq 2$. In the computation of $\bar{\boldsymbol{E}}$ the norm is fixed by Eq. 3.16. This step has also to be considered in the accuracy estimation. It is approximated by adjusting the Frobenius norm of the covariance matrix to the Frobenius norm of $\bar{\boldsymbol{E}}$ which yields

$$\boldsymbol{\Gamma}_{\bar{\boldsymbol{E}}} = \boldsymbol{H}_{\boldsymbol{E}} \boldsymbol{\Gamma}_{\hat{\boldsymbol{A}}^T} \boldsymbol{H}_{\boldsymbol{E}}^T \frac{2}{\sum_{i=1}^{3} \sigma_{\hat{\boldsymbol{A}};i}^2} \quad \text{with} \quad \boldsymbol{U}_{\hat{\boldsymbol{A}}} \, \mathrm{diag}\left(\sigma_{\hat{\boldsymbol{A}};1}, \sigma_{\hat{\boldsymbol{A}};2}, \sigma_{\hat{\boldsymbol{A}};3}\right) \boldsymbol{V}_{\hat{\boldsymbol{A}}}^T = \mathrm{SVD}\left(\hat{\boldsymbol{A}}\right) . \tag{6.14}$$

The square root for the factor $\frac{2}{\sum_{i=1}^{3} \sigma_{\hat{\boldsymbol{A}};i}^2}$ vanishes because it has to be squared for the

covariance matrix. Please note that this is only an approximation and does actually not correspond to the counterpart of Eq. 3.16, because it is not assured that the singular values of $\bar{\boldsymbol{E}}$ are $\mathrm{diag}(1,1,0)$ but only that the Frobenius norm is 2.

6.1.2 Error Propagation for the Rotation Matrix

According to Eq. 3.23, the rotation matrix is estimated by $\boldsymbol{R} = \boldsymbol{U_E Z W V_E}^T$. Consequently the error of $\boldsymbol{R}$ propagates as follows

$$\begin{aligned} \boldsymbol{R} + \boldsymbol{\Delta_R} &= \left(\boldsymbol{U_E} + \boldsymbol{\Delta_{U_E}}\right) \boldsymbol{ZW} \left(\boldsymbol{V_E} + \Delta_{\boldsymbol{V_E}}\right)^T \Rightarrow \\ \boldsymbol{\Delta_R} &\cong \boldsymbol{\Delta_{U_E}} \boldsymbol{ZW V_E}^T + \boldsymbol{U_E ZW} \Delta_{\boldsymbol{V_E}}^T . \end{aligned} \tag{6.15}$$

Hence, we need to propagate the error for $\boldsymbol{\Delta_{U_E}}$ and $\boldsymbol{\Delta_{V_E}}$. The vectors of $\boldsymbol{U_E}$ and $\boldsymbol{V_E}$ are eigenvectors of $\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T$ and $\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}$ respectively. Thus, we first need to compute the transistion matrix $\boldsymbol{G}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}}$ by

$$\boldsymbol{\delta}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}} \cong \left(\mathrm{col}\left[\boldsymbol{I}_3 \otimes \left(\bar{\boldsymbol{E}}_{-,i}\right)^T\right]_{i=1}^3 + \boldsymbol{I}_3 \otimes \bar{\boldsymbol{E}}^T\right) \boldsymbol{\delta_e} \overset{def}{=} \boldsymbol{G}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}}\, \boldsymbol{\delta_e} . \tag{6.16}$$

Now, the error in $\boldsymbol{U_E}$ and $\boldsymbol{V_E}$ can be computed by using again the aforementioned theorem

$$\begin{aligned} \boldsymbol{\delta}_{\boldsymbol{U}_{\bar{\boldsymbol{E}}}} &\cong \mathrm{col}\left[\boldsymbol{U_E}\, \boldsymbol{D}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T;i}\, \boldsymbol{U}_{\boldsymbol{E}}^T \left(\left(\boldsymbol{U}_{\boldsymbol{E};-,i}\right)^T \otimes \boldsymbol{I}_3\right)\right]_{i=1}^3 \boldsymbol{\delta}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T} \\ &\overset{def}{=} \boldsymbol{G}_{\boldsymbol{U}_{\bar{\boldsymbol{E}}}}\, \boldsymbol{\delta}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T} \quad \text{and} \end{aligned} \tag{6.17}$$

$$\begin{aligned} \boldsymbol{\delta}_{\boldsymbol{V}_{\bar{\boldsymbol{E}}}} &\cong \mathrm{col}\left[\boldsymbol{V_E}\, \boldsymbol{D}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}};i}\, \boldsymbol{V}_{\boldsymbol{E}}^T \left(\left(\boldsymbol{V}_{\boldsymbol{E};-,i}\right)^T \otimes \boldsymbol{I}_3\right)\right]_{i=1}^3 \boldsymbol{\delta}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}} \\ &\overset{def}{=} \boldsymbol{G}_{\boldsymbol{V}_{\bar{\boldsymbol{E}}}}\, \boldsymbol{\delta}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}} . \end{aligned} \tag{6.18}$$

where we choose

$$\boldsymbol{D}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T;1} = \boldsymbol{D}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T;2} = \boldsymbol{D}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}};1} = \boldsymbol{D}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}};2} = \mathrm{diag}(0,0,1) \tag{6.19}$$

as proposed by [108], to avoid a division by zero.

Finally, the perturbation in the rotation matrix can be computed by

$$\boldsymbol{\delta_R} = \begin{pmatrix} \boldsymbol{G_{RU}} & \boldsymbol{G_{RV}} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta}_{\boldsymbol{U}_{\bar{\boldsymbol{E}}}} \\ \boldsymbol{\delta}_{\boldsymbol{V}_{\bar{\boldsymbol{E}}}} \end{pmatrix} . \tag{6.20}$$

with

$$\boldsymbol{G_{RU}} = \operatorname{col}\left[\left(\boldsymbol{Z}_{1,2}\boldsymbol{V}_{\boldsymbol{E};i,2} \quad \boldsymbol{Z}_{2,1}\boldsymbol{V}_{\boldsymbol{E};i,1} \quad \det\left(\boldsymbol{U}_{\boldsymbol{E}}\boldsymbol{V}_{\boldsymbol{E}}^{T}\right)\boldsymbol{V}_{\boldsymbol{E};i,3}\right) \otimes \boldsymbol{I}_3\right]_{i=1}^{3} \quad \text{and} \tag{6.21}$$

$$\boldsymbol{G_{RV}} = \left(\boldsymbol{I}_3 \otimes \boldsymbol{Z}_{2,1}\boldsymbol{U}_{\boldsymbol{E};i,2} \quad \boldsymbol{I}_3 \otimes \boldsymbol{Z}_{1,2}\boldsymbol{U}_{\boldsymbol{E};i,1} \quad \boldsymbol{I}_3 \otimes \det\left(\boldsymbol{U}_{\boldsymbol{E}}\boldsymbol{V}_{\boldsymbol{E}}^{T}\right)\boldsymbol{U}_{\boldsymbol{E};i,3}\right) . \tag{6.22}$$

This yields the following covariance matrix for the rotation matrix

$$\boldsymbol{\Gamma_R} = \left(\boldsymbol{G_{RU}}\,\boldsymbol{G_{U_{\bar{E}}}} \quad \boldsymbol{G_{RV}}\,\boldsymbol{G_{V_{\bar{E}}}}\right)\begin{pmatrix}\boldsymbol{G_{\bar{E}\bar{E}^T}}\\ \boldsymbol{G_{\bar{E}^T\bar{E}}}\end{pmatrix}\boldsymbol{\Gamma}_{\bar{\boldsymbol{E}}}\left(\boldsymbol{G}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T}^{T} \quad \boldsymbol{G}_{\bar{\boldsymbol{E}}^T\bar{\boldsymbol{E}}}^{T}\right)\begin{pmatrix}\boldsymbol{G}_{\boldsymbol{U}_{\bar{E}}}^{T}\,\boldsymbol{G}_{\boldsymbol{RU}}^{T}\\ \boldsymbol{G}_{\boldsymbol{V}_{\bar{E}}}^{T}\,\boldsymbol{G}_{\boldsymbol{RV}}^{T}\end{pmatrix} . \tag{6.23}$$

Actually, the error of a rotation matrix has to meet some requirements in order to fulfill the orthogonality constraint of a DCM [216]:

$$\begin{aligned}\boldsymbol{I} &= \left(\boldsymbol{R}+\boldsymbol{\Delta_R}\right)\left(\boldsymbol{R}+\boldsymbol{\Delta_R}\right)^T\\ &= \boldsymbol{R}\boldsymbol{R}^T + \boldsymbol{\Delta_R}\boldsymbol{R} + \boldsymbol{R}\boldsymbol{\Delta}_{\boldsymbol{R}}^{T} + \mathcal{O}\left(\boldsymbol{\Delta_R}\boldsymbol{\Delta}_{\boldsymbol{R}}^{T}\right)\\ &\cong \boldsymbol{I} + \boldsymbol{\Delta_R}\boldsymbol{R} + \boldsymbol{R}\boldsymbol{\Delta}_{\boldsymbol{R}}^{T}\\ &\Rightarrow \boldsymbol{\Delta_R}\boldsymbol{R}^T = -\left(\boldsymbol{\Delta_R}\boldsymbol{R}^T\right)^T \overset{def}{=} \boldsymbol{\Delta'_R} = \left[\boldsymbol{d'_R}\right]_\times .\end{aligned} \tag{6.24}$$

where the operator $[\cdot]_\times$ denotes the cross-product skew matrix. It follows that

$$\boldsymbol{R}+\Delta_{\boldsymbol{R}} = \left(\boldsymbol{I}+\Delta_{\boldsymbol{R}}\boldsymbol{R}^T\right)\boldsymbol{R} = \left(\boldsymbol{I}+\Delta'_{\boldsymbol{R}}\right)\boldsymbol{R} \overset{def}{=} \boldsymbol{R}_{\Delta'_{\boldsymbol{R}}}\boldsymbol{R} . \tag{6.25}$$

Thus, geometric insights of the rotation error can be gained computing the error rotation axis $\boldsymbol{d'_R}$. It can be derived by finding the kernel of $\Delta'_{\boldsymbol{R}}$, which is the eigenvector of $\boldsymbol{\Delta'_R}$ corresponding to its smallest eigenvalue. However, we do not know the error-free $\boldsymbol{R}$ and, thus, we cannot perform the eigenvalue decomposition directly. By multiplying the error with its transposed we can eleminate $\boldsymbol{R}$, such that

$$\boldsymbol{Q} = \boldsymbol{\Delta'_R}\boldsymbol{\Delta'}_{\boldsymbol{R}}^{T} = \boldsymbol{\Delta_R}\boldsymbol{\Delta}_{\boldsymbol{R}}^{T} = -\boldsymbol{\Delta_R}\boldsymbol{\Delta_R} . \tag{6.26}$$

The power of a matrix does not change its eigenvectors but only apply to its eigenvalues. Hence, we can compute the kernel of matrix $\boldsymbol{\Delta'_R}$ by the left singular vector of $\boldsymbol{Q}$ corresponding to the smallest singular value. The magnitude of the error rotation $\boldsymbol{d'_R}$ corresponds to the two largest singular values of $\boldsymbol{\Delta'_R}$. Actually, the singular values of the skew-symmetric matrix $\boldsymbol{\Delta'_R}$ should look like $\{\|\boldsymbol{d'_R}\|, \|\boldsymbol{d'_R}\|, 0\}$. We compute the magnitude of the error rotation by the mean of the square roots of the two largest

singular values, such that

$$\boldsymbol{d}'_{\boldsymbol{R}} = \sigma_m \boldsymbol{U}_{\boldsymbol{Q};-,3} \quad \text{with} \quad \sigma_m = 0.5\left(\sqrt{\sigma_{\boldsymbol{Q};1}} + \sqrt{\sigma_{\boldsymbol{Q};2}}\right) \tag{6.27}$$

and $\boldsymbol{U}_{\boldsymbol{Q}}\,\mathrm{diag}(\sigma_{\boldsymbol{Q};1}, \sigma_{\boldsymbol{Q};2}, \sigma_{\boldsymbol{Q};3})\,\boldsymbol{V}_{\boldsymbol{Q}}^T = \mathrm{SVD}(\boldsymbol{Q})$. The matrix $\boldsymbol{Q}$ can be composed by the diagonal elements of $\boldsymbol{\Gamma}_{\boldsymbol{R}}$ as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{\Gamma}_{\boldsymbol{R};1,1} + \boldsymbol{\Gamma}_{\boldsymbol{R};4,4} + \boldsymbol{\Gamma}_{\boldsymbol{R};7,7} & \boldsymbol{\Gamma}_{\boldsymbol{R};1,2} + \boldsymbol{\Gamma}_{\boldsymbol{R};4,5} + \boldsymbol{\Gamma}_{\boldsymbol{R};7,8} & \boldsymbol{\Gamma}_{\boldsymbol{R};1,3} + \boldsymbol{\Gamma}_{\boldsymbol{R};4,6} + \boldsymbol{\Gamma}_{\boldsymbol{R};7,9} \\ \boldsymbol{\Gamma}_{\boldsymbol{R};2,1} + \boldsymbol{\Gamma}_{\boldsymbol{R};5,4} + \boldsymbol{\Gamma}_{\boldsymbol{R};8,7} & \boldsymbol{\Gamma}_{\boldsymbol{R};2,2} + \boldsymbol{\Gamma}_{\boldsymbol{R};5,5} + \boldsymbol{\Gamma}_{\boldsymbol{R};8,8} & \boldsymbol{\Gamma}_{\boldsymbol{R};2,3} + \boldsymbol{\Gamma}_{\boldsymbol{R};5,6} + \boldsymbol{\Gamma}_{\boldsymbol{R};8,9} \\ \boldsymbol{\Gamma}_{\boldsymbol{R};3,1} + \boldsymbol{\Gamma}_{\boldsymbol{R};6,4} + \boldsymbol{\Gamma}_{\boldsymbol{R};9,7} & \boldsymbol{\Gamma}_{\boldsymbol{R};3,2} + \boldsymbol{\Gamma}_{\boldsymbol{R};6,5} + \boldsymbol{\Gamma}_{\boldsymbol{R};9,8} & \boldsymbol{\Gamma}_{\boldsymbol{R};3,3} + \boldsymbol{\Gamma}_{\boldsymbol{R};6,6} + \boldsymbol{\Gamma}_{\boldsymbol{R};9,9} \end{pmatrix}. \tag{6.28}$$

6.1.3 Error Propagation for the Translation Vector

We know from Eq. 3.28 that the translation vector equals up to sign to the eigenvector corresponding to the smallest eigenvalue of $\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T$. Hence, we first need to estimate the error transition similar to Eq. 6.8

$$\boldsymbol{\delta}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T} \cong \left(\mathrm{row}\left[\boldsymbol{I}_3 \otimes \bar{\boldsymbol{E}}_{-,i}\right]_{i=1}^3 + \bar{\boldsymbol{E}} \otimes \boldsymbol{I}_3\right)\boldsymbol{\delta}_{\boldsymbol{e}} \overset{def}{=} \boldsymbol{G}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T}\,\boldsymbol{\delta}_{\boldsymbol{e}}\,. \tag{6.29}$$

Analog to Eq. 6.9, the first order perturbation propagation of $\tilde{\boldsymbol{t}}$ can then be computed using the above theorem, by

$$\begin{aligned} \boldsymbol{\delta}_{\tilde{\boldsymbol{t}}} &\cong \boldsymbol{U}_{\boldsymbol{E}}\boldsymbol{D}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T;3}\boldsymbol{U}_{\boldsymbol{E}}^T\left(\tilde{\boldsymbol{t}}^T \otimes \boldsymbol{I}_3\right)\boldsymbol{\delta}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T} \\ &\overset{def}{=} \boldsymbol{G}_{\tilde{\boldsymbol{t}}}\,\boldsymbol{\delta}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T} \end{aligned} \tag{6.30}$$

with $\mathrm{SVD}(\bar{\boldsymbol{E}}) = \boldsymbol{U}_{\boldsymbol{E}}\boldsymbol{\Sigma}_{\bar{\boldsymbol{E}}}\boldsymbol{V}_{\boldsymbol{E}}^T$.

This yields the following covariance matrix, which is equal to the one for $\boldsymbol{t}$, due to the nature of a covariance matrix (the transition matrix $\boldsymbol{G}_{\tilde{\boldsymbol{t}}}$ is applied twice and becomes symmetric)

$$\boldsymbol{\Gamma}_{\boldsymbol{t}} = \boldsymbol{\Gamma}_{\tilde{\boldsymbol{t}}} = \boldsymbol{G}_{\tilde{\boldsymbol{t}}}\boldsymbol{G}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T}\boldsymbol{\Gamma}_{\boldsymbol{E}}\boldsymbol{G}_{\bar{\boldsymbol{E}}\bar{\boldsymbol{E}}^T}^T\boldsymbol{G}_{\tilde{\boldsymbol{t}}}^T\,. \tag{6.31}$$

In Section 8.4.1, we will evaluate the quality of the first order error propagation using synthetic data.

6.2 Error Propagation of the Z_∞-Algorithm

The motion estimation of the Z_∞-algorithm is split into two parts, the rotation and the translation estimation. We are going to analyze the first order error propagation without considering the correlation of these two components, because an exhaustive analysis is tedious and beyond the scope of this work. However, the mutual influence of rotation and translation will be discussed in the experiment Section 8.4.2.

6.2.1 Error Propagation for the Rotation Matrix

The pure rotation estimation of a point cloud is part of the so called orthogonal Procrustes problem. Its first order accuracy analysis has been presented and discussed in [216]. We adapt this approach to provide a first order error propagation for the Z_∞ rotation estimation.

Using the same rotation error definition as in Eq. 6.25, the column vector $\boldsymbol{\delta_R}$ of the matrix $\boldsymbol{\Delta_R}$ can be computed from $[\boldsymbol{d'_R}]_\times$ by

$$\boldsymbol{\delta_R} = \operatorname{col}\left[[-\boldsymbol{R}_{-,i}]_\times\right]_{i=1}^{3} \boldsymbol{d'_R} \overset{def}{=} \boldsymbol{G_R}\, \boldsymbol{d'_R}\,. \tag{6.32}$$

According to the error propagation of the Procrustes problem in [216], the covariance matrix for $\boldsymbol{d'_R}$ can be estimated by

$$\boldsymbol{\Gamma}_{\boldsymbol{d'_R}} = -\boldsymbol{R}\boldsymbol{H}\boldsymbol{R}^T \left(\sum_{i=1}^{N} \left([\boldsymbol{R}\boldsymbol{r}_i^*]_\times \boldsymbol{\Gamma}_{\boldsymbol{r}_i'^*} [\boldsymbol{R}\boldsymbol{r}_i^*]_\times \right) + \sum_{i=1}^{N} \left([\boldsymbol{r}_i'^*]_\times \boldsymbol{R}\boldsymbol{\Gamma}_{\boldsymbol{r}_i^*}\boldsymbol{R}^T [\boldsymbol{r}_i'^*]_\times \right) \right) \boldsymbol{R}\boldsymbol{H}\boldsymbol{R}^T \tag{6.33}$$

with N being the number of features and

$$\boldsymbol{H} = (\operatorname{tr}(\boldsymbol{S})\,\boldsymbol{I} - \boldsymbol{S})^{-1} \quad \text{where} \quad \operatorname{EVD}(\boldsymbol{S}) = \boldsymbol{V_M}\, \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)\, \boldsymbol{V_M}^T \tag{6.34}$$

and $\operatorname{SVD}(\boldsymbol{M}) = \boldsymbol{U_M}\, \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)\, \boldsymbol{V_M}^T$ according to Eq. 5.11. The vectors $\boldsymbol{r}_i^*$ and $\boldsymbol{r}_i'^*$ were introduced in Eq. 5.10 and the singular values σ_1, σ_2 and σ_3 of $\boldsymbol{M}$ are in decreasing order. Please note that $\boldsymbol{H}$ might also be written as

$$\boldsymbol{H} = \boldsymbol{V_M}\, \operatorname{diag}\left(\frac{1}{\sigma_2+\sigma_3}, \frac{1}{\sigma_1+\sigma_3}, \frac{1}{\sigma_1+\sigma_2} \right) \boldsymbol{V_M}^T\,. \tag{6.35}$$

Hence, the weighting matrix $\boldsymbol{H}$ has a similar ordering as the inertial shape of the cloud $\boldsymbol{S}$ (which is the Hermetian component of the polar decomposition) with the

terms muting their ratios, which reminds to the weighting term of the first order error propagation of an eigenvector in Eq. 6.4.

It remains to compute the covariance matrices $\mathbf{\Gamma}_{\boldsymbol{r}_i^*}$ and $\mathbf{\Gamma}_{\boldsymbol{r}_i'^*}$ from the feature detector accuracy σ^2. As in Eq. 6.11 we start from

$$\mathbf{\Gamma}_{\tilde{\boldsymbol{r}}} = \mathbf{\Gamma}_{\tilde{\boldsymbol{r}}'} = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} ,$$

but for our rotation estimation we are using unit length rays. Hence, according to the theorem of intersecting lines, the deviation changes, such that

$$\mathbf{\Gamma}_{\boldsymbol{r}_i^*} = \left(\frac{z_{\boldsymbol{r}_i}}{f}\right)^2 \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{\Gamma}_{\boldsymbol{r}_i'^*} = \left(\frac{z_{\boldsymbol{r}_i'}}{f}\right)^2 \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix} , \tag{6.36}$$

where $z_{\boldsymbol{r}_i}$ is the z-coordinate of the unit-ray $\boldsymbol{r}_i$ and f is the focal length as defined in Eq. 2.9. The factor $\frac{z_{\boldsymbol{r}_i'}}{f}$ simply corresponds to the reciprocal of the vector length $\frac{1}{\|\tilde{\boldsymbol{r}}\|}$. The shift of the origin to the centroid does not affect the variance.

Finally, putting together Eq. 6.33 and 6.32 we get

$$\mathbf{\Gamma}_{\boldsymbol{R}} = \boldsymbol{G}_{\boldsymbol{R}} \mathbf{\Gamma}_{\boldsymbol{d}'_{\boldsymbol{R}}} \boldsymbol{G}_{\boldsymbol{R}}^T . \tag{6.37}$$

6.2.2 Error Propagation for the Translation Vector

The error in the translation estimation has to be propagated through Eq. 5.21. The initial error $\boldsymbol{\Delta}_{\boldsymbol{B}}$ of $\boldsymbol{B}$ can be estimated according to

$$\boldsymbol{\Delta}_{\boldsymbol{B}} = \begin{pmatrix} \boldsymbol{\Delta}_{\boldsymbol{A}} & \boldsymbol{\delta}_{\boldsymbol{d}} \end{pmatrix} \tag{6.38}$$

with

$$\begin{aligned} \boldsymbol{n} + \boldsymbol{\delta}_{\boldsymbol{n}} &= \begin{pmatrix} \left(y_{\tilde{\boldsymbol{r}}_i'} + \delta_{y_{\tilde{\boldsymbol{r}}_i'}}\right) - \left(y_{\boldsymbol{r}_i} + \delta_{y_{\boldsymbol{r}_i}}\right) \\ \left(x_{\boldsymbol{r}_i} + \delta_{x_{\boldsymbol{r}_i}}\right) - \left(x_{\tilde{\boldsymbol{r}}_i'} + \delta_{x_{\tilde{\boldsymbol{r}}_i'}}\right) \end{pmatrix} \\ \Rightarrow \boldsymbol{\delta}_{\boldsymbol{n}} &= \begin{pmatrix} \delta_{y_{\tilde{\boldsymbol{r}}_i'}} - \delta_{y_{\boldsymbol{r}_i}} \\ \delta_{x_{\boldsymbol{r}_i}} - \delta_{x_{\tilde{\boldsymbol{r}}_i'}} \end{pmatrix} \\ \Rightarrow \boldsymbol{\Delta}_{\boldsymbol{A}} &= \begin{pmatrix} \boldsymbol{\delta}_{\boldsymbol{n}_1} & \boldsymbol{\delta}_{\boldsymbol{n}_2} & \dots & \boldsymbol{\delta}_{\boldsymbol{n}_{|\mathcal{R}_{\text{out}}|}} \end{pmatrix}^T \end{aligned} \tag{6.39}$$

and

$$\begin{aligned} d + \delta_d &= (\boldsymbol{n} + \boldsymbol{\delta_n})^T \begin{pmatrix} x_{\boldsymbol{r}} + \delta_{x_{\boldsymbol{r}}} \\ y_{\boldsymbol{r}} + \delta_{y_{\boldsymbol{r}}} \end{pmatrix} \\ &= (x_{\boldsymbol{n}} + x_{\boldsymbol{\delta_n}})\,(x_{\boldsymbol{r}} + \delta_{x_{\boldsymbol{r}}}) + (y_{\boldsymbol{n}} + y_{\boldsymbol{\delta_n}})\,(y_{\boldsymbol{r}} + \delta_{y_{\boldsymbol{r}}}) \\ &\Rightarrow \delta_d \cong x_{\boldsymbol{n}}\delta_{x_{\boldsymbol{r}}} + x_{\boldsymbol{\delta_n}} x_{\boldsymbol{r}} + y_{\boldsymbol{n}}\delta_{y_{\boldsymbol{r}}} + y_{\boldsymbol{\delta_n}} y_{\boldsymbol{r}} \\ &\Rightarrow \boldsymbol{\delta_d} \cong \begin{pmatrix} \delta_{d_1} & \delta_{d_2} & \dots & \delta_{d_{|\mathcal{R}_{\text{out}}|}} \end{pmatrix}^T . \end{aligned} \tag{6.40}$$

In the following, we will use the transposed error matrix $\boldsymbol{\delta_{B^T}}$ instead of $\boldsymbol{\delta_B}$ because it will ease the computation of the covariance matrix later on.

The error propagation from the feature-error to $\boldsymbol{\delta_{B^T}}$ can be computed by following transition matrices:

$$\boldsymbol{\delta_{r,n}} = \begin{pmatrix} \boldsymbol{\delta_{r_1}} \\ \boldsymbol{\delta_{n_1}} \\ \boldsymbol{\delta_{r_2}} \\ \boldsymbol{\delta_{n_2}} \\ \vdots \\ \boldsymbol{\delta_{r_{|\mathcal{R}_{\text{out}}|}}} \\ \boldsymbol{\delta_{n_{|\mathcal{R}_{\text{out}}|}}} \end{pmatrix} = \left(\boldsymbol{I}_{|\mathcal{R}_{\text{out}}|} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \end{pmatrix} \right) \begin{pmatrix} \boldsymbol{\delta_{r_1}} \\ \boldsymbol{\delta_{\tilde{r}'_1}} \\ \boldsymbol{\delta_{r_2}} \\ \boldsymbol{\delta_{\tilde{r}'_2}} \\ \vdots \\ \boldsymbol{\delta_{r_{|\mathcal{R}_{\text{out}}|}}} \\ \boldsymbol{\delta_{\tilde{r}'_{|\mathcal{R}_{\text{out}}|}}} \end{pmatrix} \overset{def}{=} \boldsymbol{G_{r,n}}\,\boldsymbol{\delta_{r,\tilde{r}'}} \tag{6.41}$$

and

$$\boldsymbol{\delta_{B^T}} \cong \text{Diag}\big(\boldsymbol{P}_1, \boldsymbol{P}_2, \dots, \boldsymbol{P}_{|\mathcal{R}_{\text{out}}|}\big)\ \boldsymbol{\delta_{r,n}} \overset{def}{=} \boldsymbol{G_{B^T}}\,\boldsymbol{\delta_{r,n}} \tag{6.42}$$

with

$$\boldsymbol{P}_i = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ x_{\boldsymbol{n}_i} & y_{\boldsymbol{n}_i} & x_{\tilde{\boldsymbol{r}}_i} & y_{\tilde{\boldsymbol{r}}_i} \end{pmatrix} . \tag{6.43}$$

The error of $\hat{\boldsymbol{t}}$, which is derived from the least-squares solution of Eq. 5.21 according to Eq. 5.22, can be propagated using the span-of-eigenvector error propagation theorem introduced at the beginning of Section 6.1 and as specified by Eq. 6.4. Thus, we first need to compute the transition matrix from the error vector $\boldsymbol{\delta_{B^T}}$ to $\boldsymbol{\delta_{B^TB}}$ similar to Eq. 6.8

$$\boldsymbol{\delta_{B^TB}} \cong \left(\text{row}\left[\boldsymbol{I}_3 \otimes \boldsymbol{B}^T_{i,-} \right]_{i=1}^{|\mathcal{R}_{\text{out}}|} + \boldsymbol{B}^T \otimes \boldsymbol{I}_3 \right) \boldsymbol{\delta_B} \overset{def}{=} \boldsymbol{G_{B^TB}}\,\boldsymbol{\delta_{B^T}} . \tag{6.44}$$

Now we can propagate the perturbation of the eigenvector by

$$\begin{aligned}\boldsymbol{\delta}_{\bar{\boldsymbol{e}}} &\cong \boldsymbol{V}_{\boldsymbol{B}}\boldsymbol{D}_{\boldsymbol{B}^T\boldsymbol{B};3}\boldsymbol{V}_{\boldsymbol{B}}^T\boldsymbol{\Delta}_{\boldsymbol{B}^T\boldsymbol{B}}\begin{pmatrix}\boldsymbol{e}\\-1\end{pmatrix}\\ &= \boldsymbol{V}_{\boldsymbol{B}}\boldsymbol{D}_{\boldsymbol{B}^T\boldsymbol{B};3}\boldsymbol{V}_{\boldsymbol{B}}^T\left(\begin{pmatrix}\boldsymbol{e}\\-1\end{pmatrix}^T\otimes\boldsymbol{I}_3\right)\boldsymbol{\delta}_{\boldsymbol{B}^T\boldsymbol{B}}\\ &\overset{def}{=} \boldsymbol{G}_{\bar{\boldsymbol{e}}}\,\boldsymbol{\delta}_{\boldsymbol{B}^T\boldsymbol{B}}\,,\end{aligned} \tag{6.45}$$

where $\boldsymbol{e}$ has been introduced in Eq. 5.22. According to those formulas, the error of $\boldsymbol{\delta}_{\tilde{\boldsymbol{t}}}$ of the translation direction (up to sign) $\tilde{\boldsymbol{t}}$ can be estimated from $\boldsymbol{\delta}_{\bar{\boldsymbol{e}}}$ according to

$$\boldsymbol{\delta}_{\tilde{\boldsymbol{t}}} = \begin{pmatrix}\frac{\boldsymbol{\delta}_{\bar{\boldsymbol{e}}}}{\|\tilde{\boldsymbol{t}}'\|}\\0\end{pmatrix} \quad \text{and, thus,} \quad \boldsymbol{\delta}_{\tilde{\boldsymbol{t}}} = \boldsymbol{G}_{\boldsymbol{t}}\,\boldsymbol{\delta}_{\bar{\boldsymbol{e}}} \quad \text{with} \quad \boldsymbol{G}_{\boldsymbol{t}} = \begin{pmatrix}z_{\tilde{\boldsymbol{t}}} & 0 & 0\\0 & z_{\tilde{\boldsymbol{t}}} & 0\\0 & 0 & 0\end{pmatrix} \tag{6.46}$$

and $z_{\tilde{\boldsymbol{t}}}$ denoting the third component of $\tilde{\boldsymbol{t}}$. The error for $\boldsymbol{t}$ and $\tilde{\boldsymbol{t}}$ varies only in the sign and, hence, results in the same covariance propagation matrix.

It remains to compute the initial covariance matrix $\boldsymbol{\Gamma}_{\boldsymbol{r},\check{\boldsymbol{r}}'}$. Assuming again that the errors of different points and different components are uncorrelated and equally $\boldsymbol{\delta}_{\boldsymbol{r}_i,\check{\boldsymbol{r}}_i'} = \frac{\sigma}{f}\,\boldsymbol{I}_4$, the covariance matrix $\boldsymbol{\Gamma}_{\boldsymbol{r},\check{\boldsymbol{r}}'}$ can be computed based on $\boldsymbol{\Gamma}_{\boldsymbol{r},\check{\boldsymbol{r}}'} = \mathrm{E}\left(\boldsymbol{\delta}_{\boldsymbol{r},\check{\boldsymbol{r}}'}\boldsymbol{\delta}_{\boldsymbol{r},\check{\boldsymbol{r}}'}^T\right)$. As result we get

$$\boldsymbol{\Gamma}_{\boldsymbol{r},\check{\boldsymbol{r}}'} = \left(\frac{\sigma}{f}\right)^2\boldsymbol{I}_{4|\mathcal{R}_{\text{out}}|}\,. \tag{6.47}$$

Putting together the pieces we can compute the covariance matrix of the direction of translation vector $\boldsymbol{\Gamma}_{\hat{\boldsymbol{t}}}$, such that

$$\boldsymbol{\Gamma}_{\boldsymbol{t}} \cong \boldsymbol{H}_{\boldsymbol{t}}\,\boldsymbol{\Gamma}_{\boldsymbol{r},\check{\boldsymbol{r}}'}\,\boldsymbol{H}_{\boldsymbol{t}}^T \quad \text{with} \quad \boldsymbol{H}_{\boldsymbol{t}} = \boldsymbol{G}_{\boldsymbol{t}}\,\boldsymbol{G}_{\bar{\boldsymbol{e}}}\,\boldsymbol{G}_{\boldsymbol{B}^T\boldsymbol{B}}\,\boldsymbol{G}_{\boldsymbol{B}^T}\,\boldsymbol{G}_{\boldsymbol{r},\boldsymbol{n}}\,. \tag{6.48}$$

We will evaluate the quality of the presented first order error propagation using synthetic data in Section 8.4.2.

Chapter 7

Feature Tracking

The accuracy in estimating the mean value of a random variable increases with the number of its occurrences considered by the computation. Hence, it is possible to tune a tracking algorithm in two ways to increase the motion estimation accuracy: either the tracking accuracy is improved, lowering the variance of the random variable, or the algorithm is tuned to be more efficient and, thus, it allows to use more features[1].

In this chapter, we describe two tracking techniques. The first one aims for high tracking accuracy and is based on the Kanade-Lucas-Tomasi (KLT) feature tracker, whereas we enhanced it to increase efficiency and allow for sub-pixel accurate stereo-triangulation. The second one is a tracking-by-matching approach based on AGAST-features. It does not require any image pre-processing and is, thus, highly efficient.

7.1 Extended KLT Tracking

One of the most popular and efficient local tracking methods currently is KLT (see Section 3.2). In the following, we will denote a few bottlenecks and drawbacks of the algorithm and how we bypassed them in our implementation.

KLT tracks the features in the gradient image. For that in conventional implementations first off the X- and Y-axis gradients are computed for the whole image. Furthermore, if larger pixel displacements have to be detected, image pyramids are used which require a low-pass filtering of the image to prevent aliasing. This is done by convolving the image with a Gaussian smoothing filter. In Appendix A.1, we describe how the appropriate smoothing kernel for a specific image pyramid can be computed. For each pyramid level and each axis one convolution for smoothing and one for the computation of the gradient is required. For the sake of efficient one should try to

[1] Some experimental results of this effect are shown in Fig. 8.17.

avoid the computation and the processing of too many image pyramid layers, not only because of the large processing costs, but also because image pyramids may lead to wrong matches. If in an upper level, where less detail about the feature is available, a wrong correspondence (*e.g.* of a repetitive pattern) is found, there is no possibility to fix that in the lower levels. Furthermore, considering the large number of pixels per image one should try to avoid full image processing steps and reduce the search area as much as possible, of course by still allowing for the necessary dynamics.

KLT generally uses the Shi-Tomasi features (see Section 3.1) with a specified non-maximum distance threshold. However, this does not prevent that only features in a small, texture-rich region of the image are found. Such a local limitation of the features leads to a ill-posed motion estimation, independent of the algorithm.

Moreover, we can assume that we are going to track the same features for quite a while. This leads to an oblique problem. Let us briefly recapitulate the functional principle of KLT. First, good features to track have to be chosen. The selection criterion therefore are the horizontal and vertical gradients in a surrounding region. However, the tracking itself is based on a simple iterative comparing routine of the gradient images. The position is estimated with sub-pixel precision up to a specified accuracy - a residual error will always remain due to the linearization of the tracking step. Thus, in each step the new feature location is estimated and defines the new template for the next tracking step. This image-by-image tracking is highly efficient but results in an accumulation of the tracking error. There are two ways to overcome this problem. Either, we always use the keyframe patches as reference, which requires a processing expensive patch normal estimation and warping, or we try to avoid the tracking bias by binding the feature location to a drift-free point.

To overcome the stated problems we implemented following modifications of the conventional KLT:

- KLT is a local tracker, which assumes that the features do not move too much. Hence, the KLT preprocessing (gradient computation and smoothing) is reduced to small patches around the tracked points. In this way, not the entire images but only the small feature patches have to be processed and stored. Thus, also the tracking on high resolution images becomes feasible.

- A linear motion model allows not only for larger feature displacements between the images, but also for smaller search areas. The results are fewer tracking iterations and a reduced size of the image-patches. The motion is modeled based on the 2D feature displacements in the image and provides a strict separation of

the tracking and the pose estimation routines which prevents a building up of the error.

- It is often the case that a scene captured by a camera is built from different structured regions. Patterns with a rich contrast are preferred by feature detectors because they allow a good discrimination during tracking. Splitting the image into commensurate sub-images for feature selection, leads to a better landmark dispersion. A feature set, which covers a wider cone of view allows a better conditioned pose estimation.

- To prevent feature drifts, we bind the feature to the response of its detector. KLT uses the Shi and Thomasi detector which detects pixels with the largest second eigenvalue of the moment matrix of the surrounding patch and, thus, not necessarily a real corner, but in general a pixel close to it. Hence, we assume that the feature detector would always extract the same interesting point in the close pixel neighborhood of the tracked feature. Let $\boldsymbol{v}_d$ denote the vector from the sub-pixel accurate tracking result $\boldsymbol{p}_t$ to the pixel accurate feature detector response. Then the feature gets attracted by a weight resulting from the ratio between the detector response value of the tracked location $s_{\boldsymbol{p}_t}$ and the maximum one $s_{\boldsymbol{v}_d}$. Because the tracked location is not a single pixel its value has to be interpolated by the surrounding pixels. At the end we get a novel sub-pixel accurate and drift-compensated feature location $\boldsymbol{p}'$:

$$\boldsymbol{p}' = \boldsymbol{p}_t + \left(1 - \frac{s_{\boldsymbol{p}_t}}{s_{\boldsymbol{v}_d}}\right) \boldsymbol{v}_d \,. \tag{7.1}$$

  This yields a mixture between gradient-descent tracking and tracking by matching as it is used in Section 7.2.3.

Some experimental results are shown in Section 8.1.1.

The computational costs can be significantly reduced by tracking the features only in one instead of both cameras. However, to increase the motion estimation accuracy the features 3D structure has to be initialized. There are three different ways to initialize the scale from images: by using the dimensions of some known objects (*structure from reference*), by moving a camera (*structure from motion*) or by using a stereo camera system with stereo-triangulation (*structure from stereo*). In the next section, we describe a method which allows to find sub-pixel accurate feature correspondences for an accurate stereo-triangulation.

7.1.1 KLT-Based Stereo Initialization

The default way to find stereo correspondences is to search for a corresponding patch on the epipolar line [108]. For that the neighborhood of each pixel on this line is compared to the pixel neighborhood of the respective feature. This results in a pixel-accurate stereo matching. Even if the epipolar line is restricted to a small area due to the distance limitation of the environment, all the pixels along that line have to be checked for matching. In practice, that line even grows to a small band due to calibration inaccuracies and if one wants to achieve sub-pixel accuracy, all possible sub-pixel locations in this band have to be interpolated. The fact, that the current problem is principally the same as in tracking, namely to find pixel-correspondences, leads to following algorithm.

If the cameras are mounted parallel on the stereo rig and the baseline is short, the affine component of the stereo-transformation can be neglected. Hence, we assume that the same features are found by a feature detector. We extract features from both images, the reference as well as the supporting image which is only used for stereo initialization, whereas for the second one we use a slightly lower threshold for feature detection, which yields more responses. Now, for each feature in the reference image we check all detected features on the epipolar line for correspondence. The features are selected according to following constraints: The coordinate of the horizontal axis (in general the X-axis) of a possible correspondence has to be smaller or larger (denoted by $\lessgtr$) than the one of the reference camera, depending on whether the left or the right camera has been chosen as reference camera. Furthermore, the distance from the epipolar line cannot exceed a predefined threshold Ξ_d, which has to be chosen according to the feature detector and the accuracy of the camera calibration. The distance from the epipolar line is computed as follows.

The relation between Fundamental matrix, the image point in the reference camera $\boldsymbol{p}_r$ and the epipolar line in the image coordinate frame of the second camera $\boldsymbol{l}$ is

$$\boldsymbol{F}\boldsymbol{p}_r = \boldsymbol{l}\,. \tag{7.2}$$

We compute the projection factor f_p of the epipolar line on the image plane by

$$f'_p = f\,\bar{\boldsymbol{l}}^T\,\bar{\boldsymbol{l}}_p \quad \text{with} \quad \bar{\boldsymbol{l}}_p = \frac{1}{\sqrt{x_{\bar{\boldsymbol{l}}}^2 + y_{\bar{\boldsymbol{l}}}^2}} \begin{pmatrix} x_{\bar{\boldsymbol{l}}} \\ y_{\bar{\boldsymbol{l}}} \\ 0 \end{pmatrix} = \frac{1}{1 - z_{\bar{\boldsymbol{l}}}^2} \begin{pmatrix} x_{\bar{\boldsymbol{l}}} \\ y_{\bar{\boldsymbol{l}}} \\ 0 \end{pmatrix} \quad , \quad \bar{\boldsymbol{l}} = \frac{\boldsymbol{l}}{\|\boldsymbol{l}\|} \tag{7.3}$$

and f being the focal length in pixel size, which leads to

$$f_p \;=\; f\,\frac{x_{\boldsymbol{l}}^2 + y_{\boldsymbol{l}}^2}{\|\boldsymbol{l}\|\left(\|\boldsymbol{l}\|^2 - z_{\boldsymbol{l}}^2\right)} \tag{7.4}$$

and, finally, get the pixel distance $d_{\boldsymbol{p}_t}$ of the feature to test $\boldsymbol{p}_t$ from the epipolar line by

$$d_{\boldsymbol{p}_t} \;=\; f_p'\,\bar{\boldsymbol{l}}^T\boldsymbol{p}_t \;=\; f_p\,\boldsymbol{l}^T\boldsymbol{p}_t\,. \tag{7.5}$$

This yields the, usually relative small, set of possible correspondences $\mathcal{C}_{\boldsymbol{p}_r}$ from the set of detected features in the second image $\mathcal{T}$, whereas

$$\mathcal{C}_{\boldsymbol{p}_r} = \{\boldsymbol{p}_t \in \mathcal{T} \,|\, x_{\boldsymbol{p}_t} \lesseqgtr x_{\boldsymbol{p}_r} \;\wedge\; |d_{\boldsymbol{p}_t}| < \Xi_d\}\;. \tag{7.6}$$

We apply the KLT Tracker on each of these matches to find the sub-pixel accurate correspondence. Hence, not every feature or even pixel, but only some interesting points within the epipolar-band are used as starting point for the KLT tracker. In case of a match, a sub-pixel accurate feature correspondence is found. If the tracker finds more than one match, we use the one with the smallest pixel difference of the gradient patches.

By limiting the search range for stereo matches to the displacement corresponding to an object's distance, only features in the specified range are found. With this restriction, we can localize in respect to an object even if it is moved, because we do not refer to landmarks out of the specified range, as, *e.g.*, on the scene background. This allows tracking of a moving object and in the 3D modeling scenario the reconstruction of dynamic subjects (see Section 9.1).

Some results of this initialization technique are presented in Section 8.1.1.

7.2 AGAST-Based Tracking by Matching

In the following, we present an approach which is less accurate but much more efficient. Due to a highly efficient corner detection we can look for good features to track in each image and match them. Thus, first we will introduce the *adaptive and generic accelerated segment test* (AGAST) and then we describe which matching policy for feature tracking we use in combination with this feature detector and when it is applicable.

7.2.1 Adaptive and Generic Accelerated Segment Test

AGAST is a corner detector which, analog to FAST, is also based on the AST (see Section 3.1.1), but which is more efficient, while being more generic too. We introduce the reader step-wise to the different concepts underlying the algorithm: the binary search tree, the optimal decision tree, and the automatic scene adaption.

Configuration Space for a Binary Search Tree

Instead of only considering a restricted configuration space, as in FAST, we propose to use a more detailed configuration space in order to provide a more efficient solution. Therefore, we do not consider two questions of the AST per time as in FAST, but we evaluate only one. The idea is as follows: choose one of the pixels to test and *one* question to pose. The question is then evaluated for this given pixel, and the response is used to decide the following pixel and question to query. Searching for a corner, hence, reduces to traversing a *binary* decision tree. Since, it is required to specify which pixel to query and the type of question to use. Consequently, the configuration space increases by the addition of two more states: "not brighter" ($\overline{b}$) and "not darker" ($\overline{d}$). Using a similar notion as in [128], the state of a pixel relative to the nucleus n, denoted by $n \to x$, is assigned as follows:

$$S_{n\to x} = \begin{cases} d, & I_{n\to x} < I_n - t & & & \text{(darker)} \\ \overline{d}, & I_{n\to x} \not< I_n - t & \bigwedge & S'_{n\to x} = u & \text{(not darker)} \\ s, & I_{n\to x} \not< I_n - t & \bigwedge & S'_{n\to x} = \overline{b} & \text{(similar)} \\ s, & I_{n\to x} \not> I_n + t & \bigwedge & S'_{n\to x} = \overline{d} & \text{(similar)} \\ \overline{b}, & I_{n\to x} \not> I_n + t & \bigwedge & S'_{n\to x} = u & \text{(not brighter)} \\ b, & I_{n\to x} > I_n + t & & & \text{(brighter)} \end{cases}, \tag{7.7}$$

where $S'_{n\to x}$ is the preceding state, I is the brightness of a pixel and u means that the state is still unknown. This results in a binary tree representation, as opposed to a ternary tree, allowing a single evaluation at each node. Please note that this increases the configuration space size to 6^N, which yields $6^{16} \approx 2 \cdot 10^{12}$ possible nodes for $N = 16$.

Associated with each branch of our tree is a processing cost, which represents the computational cost on the target machine. These costs vary due to different memory access times. We specify these as follows,

- c_R: register access cost (second comparison of the last tested pixel),
- c_C: cache access cost (testing of a pixel in the same row)

- c_M: memory access cost (testing of any other pixel).

Furthermore, for each of these, an additional cost equivalent to evaluating a greater-than operation, is required.

Building the Optimal Decision Tree

It is well known that a greedy algorithm, such as ID3, may perform rather poorly when finding the optimal decision tree [217]. However, the issue of finding such a tree is a well-studied problem, where it has been shown that finding the global optimum is NP-complete [218]. There are several solutions towards finding the optimal tree [219, 220, 221], but they are either approximations to the global optimum or are restricted to special cases, making them ill-suited for this application.

In order to find the optimal decision tree, an algorithm which is similar to the backward induction method, as described in [219], has been implemented. We explore the whole configuration space starting at the root of the decision tree, where none of the pixels is known. Nodes of the tree are formed by recursively evaluating a possible question at a given pixel. We explore the configuration space (using Depth First Search) until a leaf is found, whereas a leaf is defined as the first node on the path which fulfills or cannot fulfill anymore the AST corner criteria. The cost at a given leaf is zero, while the cost at any given internal node, c_P, is determined by picking the minimum cost computed for each child pair C_+ and C_-, representing the positive and negative results of a test, by

$$c_P = \min_{\{(C_+,C_-)\}} c_{C_+} + p_{C_+} c_T + c_{C_-} + p_{C_-} c_T = c_{C_+} + c_{C_-} + p_P c_T \tag{7.8}$$

where c_T represents the cost of the pixel evaluation with $c_T \in \{c_R, c_C, c_M\}$ and the p_P, p_{C_+} and p_{C_-} are the probabilities of the pixel configurations at the parent and child nodes respectively. Using this dynamic programming technique allows us to find the decision tree for an *optimal AST* (OAST) efficiently. The resulting decision tree can, hence, be optimized for different c_R, c_C and c_M, but also for arbitrary probabilities for each pixel configuration, which is necessary for our approach described in the following section.

The binary configuration space allows for decision trees which reduce the entropy more quickly than a ternary tree, as questions which contain little information gain are deferred to later stages of the decision process. Please note that the additional cost of re-evaluating the same pixel at a subsequent point in time is taken into account when computing the optimal tree.

Adaptive Tree Switching

Every image has, independent of the scene, homogeneous and (or) cluttered areas representing uniform surfaces or structured regions with texture. Hence, instead of learning the distribution of the pixel configurations from training images, like FAST, a first generalization would be to learn the probability of structured and homogeneous regions and optimize the decision tree according to this distribution. The resulting tree is complete and optimized for the trained scene, while being invariant to any motion. The probability of an image to be uniform can be modeled by the probability of a pixel state to be similar to the nucleus (p_s). The "brighter" and "darker" states are mirrored states, which means that, *e.g.*, a brighter pixel on the test pattern will evaluate the current nucleus pixel as darker as soon as it becomes the center pixel. Due to this mirroring the states "brighter" and "darker" are assumed to have the same probability (p_{bd}), which is chosen to sum up to one with p_s ($p_s + 2p_{bd} = 1$). Thus, the probability of a pixel configuration p_X can be computed as follows:

$$p_X = \prod_{i=1}^{N} p_i \quad \text{with } p_i = \begin{cases} 1 & \text{for } S_{n\to i} = u \\ p_s & \text{for } S_{n\to i} = s \\ p_{bd} & \text{for } S_{n\to i} = d \bigvee S_{n\to i} = b \\ p_{bd} + p_s & \text{for } S_{n\to i} = \overline{d} \bigvee S_{n\to i} = \overline{b} \end{cases} \tag{7.9}$$

The probability distribution of the pixel configuration is, therefore, a trinomial distribution with the probabilities p_s and twice p_{bd}. Please note that the states $\overline{d}$, $\overline{b}$ and u are not single samples of this distribution but represent a set of two and three, respectively. While this approach provides a good solution for the trained environment, it is not generic and, like FAST, it has to be learned for each specific scene where it is applied.

A more efficient and generic solution is achieved, if the algorithm automatically adapts to the area which is currently processed, *i.e.* it switches between decision trees which are optimized for the specific area. The idea is to build two or more trees and specialize them for a certain structuredness - *e.g.*, in case of two trees one would be ideal for homogeneous and one for structured regions, realized by a small and a large value for p_s. At the end of each decision path, where the corner criterion is met or cannot be fulfilled anymore, a jump to the appropriate specialized tree is performed based on the pixel configuration of this leaf (see Fig. 7.1). This switch between the specialized decision trees comes with *no* additional costs, because the evaluation of the leaf node is done offline when generating the specialized tree. In this way, the AST

is adapted to each image section dynamically and its performance is *increased*, for an *arbitrary* scene. Any learning becomes needless.

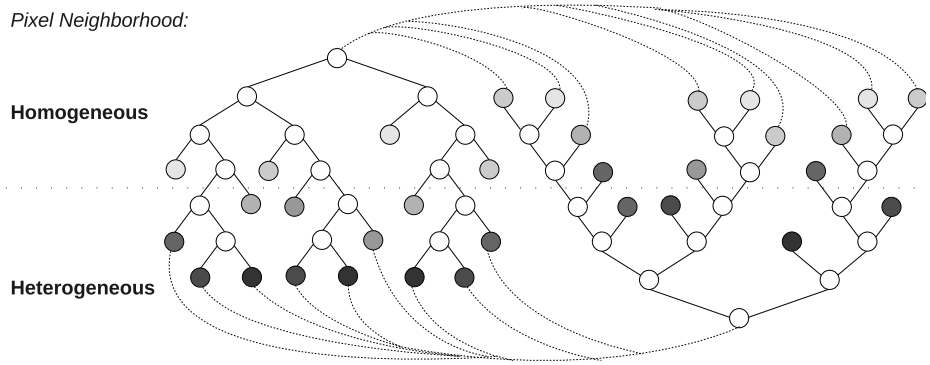


Figure 7.1: Principle of the adaptive and generic accelerated segment test. The AGAST switches between two (or more) specialized trees as soon as the pixel neighborhood changes. The lighter the gray of a leaf the more equal pixels are in the configuration. The left tree achieves less pixel evaluations (shorter decision paths) in a homogeneous pixel neighborhood, while the right one is optimized for textured regions.

Because a switch between the trees at no costs can only be performed at a leaf, the adaption is delayed by one test. Therefore, the only case were the adaptive tree switching of AGAST would worsen its performance is, if the environment switches from homogeneous to structured and vice versa at consecutive pixels. This is practically not possible, due to the mirroring effect of dissimilar pixels as described earlier. However, natural images usually do not have a random brightness distribution, but they are rather split into cluttered and uniform regions. If the decision trees can be strongly balanced by varying p_s, also more than two different weighted trees can be used.

At this point it is also worth to mention that the proper model of the probabilistic distribution would be a more complex, because the configuration states are not fully independent, due to the aforementioned mirroring effect. It can be exemplified by a single outlier in a homogeneous region, which is measured N times as unequal pixel on the test pattern, but for once it is the center pixel, too. This means that the probability of $P(0)$ is the same as $P(N-1)$ in cases where the rest of the pixel is either darker or brighter. However, the probability of $P(1)$ is not equal to $P(N-2)$, because otherwise the two outlier would have to be within the AST threshold. Hence, a complete model would have to take care of the likeliness of outliers to be similar. Experiments have shown, that only the spike of size $P(N-1)$ (for the configurations where all the pixels are brighter or all are darker) yields measurable results when overlapping the trinomial distribution. Anyway, the results in Section 8.1.2 will show, that this change influences

the corner response only slightly. This is due to the fact that balancing of the tree is rather limited, as the experiments in Section 8.1.2 will show.

7.2.2 Minimal Rotation-Invariant Descriptor

For an efficient tracking-by-matching approach, there is not only a fast feature detector needed, but also an efficient feature descriptor which allows for quick matching - but what are our constraints for such a descriptor? First, we expect to have a rather narrow search area (due to a high frame-rate to motion ratio as motivated in Section 1.1) and, thus, only a few AGAST responses have to be considered for matching. In most cases none or one feature will be detected within the locality radius which delimits possible correspondences. However, descriptors like SSD as in PTAM are not rotation invariant and a slight rotation about the optical axis may lead the matching to fail. However, as also mentioned in [136] rotation-invariance comes with the price of an increased percentage of false-positive matches. In the case at hand, we disregard this issue, which is of great relevance for global feature tracker, where all responses of a detector have to be considered, but not for local ones. Hence, a *minimal rotation-invariant descriptor* (MRID) for AGAST features is required.

It is preferred to spend as less processing time as possible for the descriptor computation. Hence, we should make use of already computed characteristics of AGAST features, which results in the following three measures for MRID:

- **Non-Maximum Suppression Measure (NMSM)**: For the non-maximum suppression of AST features one computes the maximum threshold applicable to still yield a positive response. This measure reflects a rotation-invariant property of the feature neighborhood.

- **Nucleus-Value**: The nucleus-pixel itself is best be represented by its brightness-value which is also rotation-invariant.

- **Feature Distance**: The distance between the expected and the detected location of a feature is also a good plausible measure for a match.

7.2.3 Tracking By Matching

The absolute difference of the NMSM values s_r and s_t and the nucleus-values n_r and n_t of the reference and the test feature respectively have to be below a specified threshold Ξ_s and Ξ_n. Features which are not within a certain distance Ξ_d of the propagated feature location are not even considered for matching. If these three constraints are

met, a weighted sum of the measures is used to find the most plausible match m_r for a reference feature f_r:

$$\begin{aligned} m_r &= \arg\min_{f_t \in \mathcal{F}'} \left(w_s |s_t - s_r| + w_n |n_t - n_r| + w_d \left\| p_t - p_r \right\| \right) \quad \text{with} \\ \mathcal{F}' &= \{ f_t \in \mathcal{F} \quad | \quad |s_t - s_r| < \Xi_s \ \wedge \ |n_t - n_r| < \Xi_n \ \wedge \ \left\| p_t - p_r \right\| < \Xi_d \} \ , \end{aligned} \tag{7.10}$$

where $\mathcal{F}$ is the set of all features, w_s, w_n and w_d denote the weights for the measures and p_r and p_t are the image locations.

As for the extended KLT tracker (see Section 7.1), we suggest to linearly propagate the features in 2D to allow for higher dynamic motions. Furthermore, one should aim to keep as long as possible the same reference corners to reduce drifting. A bias is only accumulated if the reference changes and, hence, sticking to the same reference set allows for a drift free stabilization in 3D space. Compared to, *e.g.*, gradient-descent methods where features get adapted from image to image to compensate for the affine transformation, tracking-by-matching methods stick in general to the same physical point. Hence, as long as valid corners are detected, their tracking is drift-free.

Chapter 8

Experiments

In this section, we will provide extensive experiments to assess and validate the algorithms discussed in the Chapters 4, 5, 6 and 7. We start with the feature detection and tracking and continue with the evaluation and comparison of the image-based pose estimation methods. As next we validate the error propagation from the detection accuracy to the pose estimation and verify the temporal and spatial alignment of an IMU-camera setup. Finally, we run some experiments on the fusion of these two sensors.

8.1 Evaluation of the Feature Tracking Methods

To increase the accuracy of motion estimation methods one can aim to improve the tracking accuracy or tracking efficiency. In Sections 7.1 and 7.2, we presented two tracking approaches where each focuses on one of these two properties. We will first evaluate the extensions made to KLT and after that we will assess AGAST-based tracking.

8.1.1 Evaluation of the Extended KLT

As proposed in Section 7.1, one way to reduce the processing load of the KLT tracking is to restrict the computation of the gradient and the image pyramids to the areas surrounding the features to track. In the following, we provide an experimental evaluation of two different approaches. The first extracts the image patches around the features and treats them as separate images. The second one processes only the regions of interest (ROI) around the feature locations but maintains the image as it is. While the patch-based approach suffer from the overhead of copying the regions, the second one requires more memory jumps for processing. Fig. 8.1 shows some experimental results

for the original, the patch-based KLT implementation and a method which uses ROIs. The experiments show that the latter approach is always less efficient than the whole-image- or patch-based variant and, thus, it can be neglected in further discussions. If there are too many features to track, the overlapping areas of the patches result in even more pixels to be processed than the image actually has, making the patch based variant less efficient. Which alternative to prefer depends, therefore, on the feature search range and the number of features. However, for a reasonable number of features the patch based alternative allows for a significant speed up.

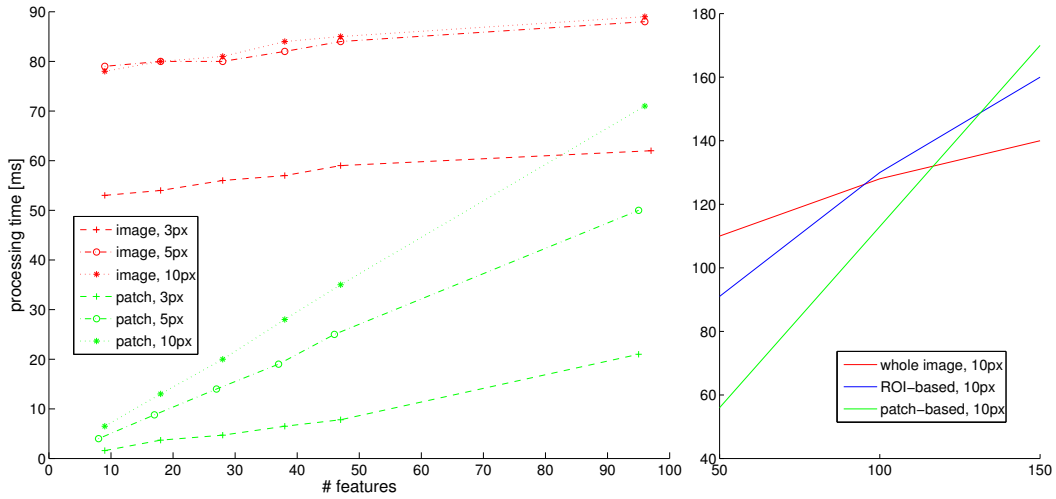
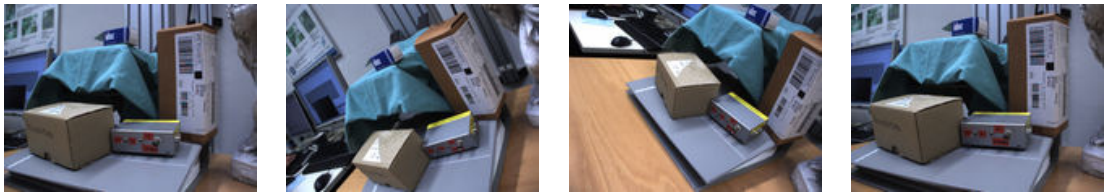


Figure 8.1: These figures compare the processing times of the KLT-variants for different number of features to track. The red values represent the whole-image-based implementation, the blue values the ROI-based version and green are the times for the patch-based variant. On the left, different sizes for the search window are chosen. While the processing time for the whole image almost only depends on the number of pyramid layers (3 px search range results in one, 5 px and 10 px in two pyramid layers), the patch-based approach is also related to the search range and the number of features. The right image shows the intersection of the lines for a 10 px search range.

In Section 7.1, an extension of the KLT tracking routine to bind a feature to a corner has been explained. This modification prevents an error accumulation while tracking. Figure 8.2 illustrates the improvements of the tracking accuracy provided by this new functionality. Fig. 8.2 makes apparent that within an image sequence the KLT-patch may change its appearance significantly. Due to the step-wise proceeding the affine transformation is ignored and becomes adapted from image to image. Especially if some occlusions or disocclusions occur, as in Fig. 8.2(e), the tracking patch is massively changed. Due to the slow alteration, the modification is below the difference-threshold and is not detected by KLT. The patch becomes a virtual feature and follows the



(a) images 1, 620, 720 and 900

(b) feature-patch 15 with conventional KLT tracking

(c) feature-patch 15 with corner binding

(d) feature-patch 49 with conventional KLT tracking

(e) feature-patch 49 with corner binding

Figure 8.2: Two different patches are tracked in an image sequence with and without corner binding. Fig. 8.2(a) shows the images where the features are extracted. While patch 15 differs only by one pixel from the original location after 900 tracking steps, patch 49 has been drifted much further due to partially occlusion. For these two patch-sequences, the corner-binding variant nullifies the accumulated tracking residues. The green square illustrates the integral-window and the green point its center and, thus, the feature location.

occlusion. This effect can be suppressed by the corner-bind method, which ties a feature to an interesting point and, thus, suppresses drifting.

Stereo Initialization Analysis

The results of the fast and sub-pixel-precise stereo initialization method, as described in Section 7.1.1, are depicted in Fig. 8.3. The stereo pairs of two different scenes with the initialization result are illustrated. The red dots in both shots are the extracted points of interest which are used as matching candidates. The green lines link up corresponding features of the left camera (upper image) and the right one (lower image). The green dots at the end of the lines in the lower image are the sub-pixel accurate correspondences. Blue lines are matches, where the KLT tracker could not find a minimum due to the limited iteration number. Nevertheless, experiments have shown, that these correspondences are mostly also as accurate as regular matches (as it can be seen in the small figures). The lower small images show enlargements of the putto-scene (left) scene which depict in detail two corresponding parts of the large pictures. The lines illustrate the correspondences, which proves that the proper matches were found.

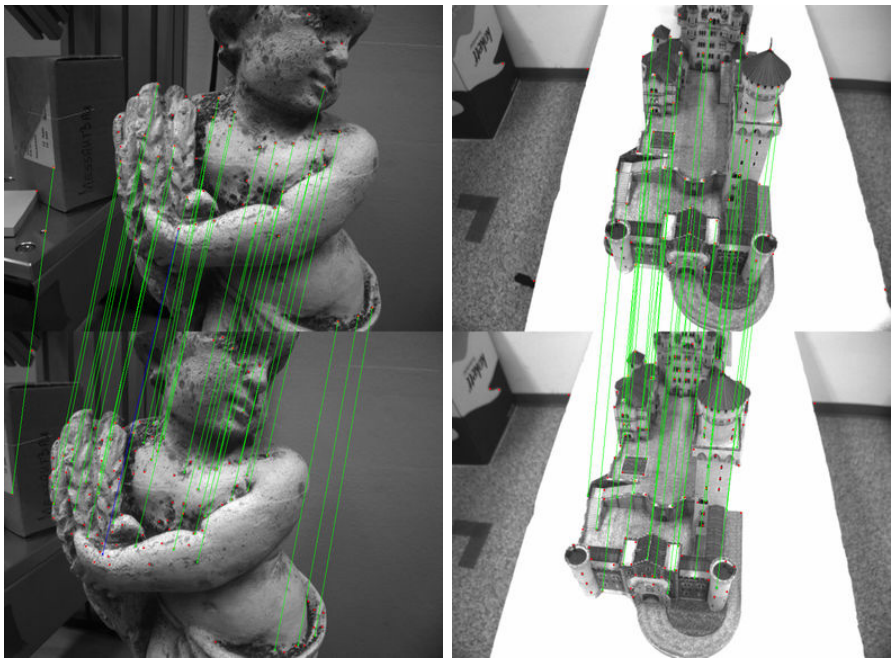
8.1.2 Efficiency Evaluation of AGAST

The speed and the repeatability of FAST have already been compared to state of the art corner detection algorithms in [127]. In those experiments FAST-9 has demonstrated better performance than, *e.g.*, Harris, DoG, or SUSAN. Thus, we renounce to compare our AST variation only with FAST-9. Please note that our approach is also based on the AST and, therefore, it detects the same corners and, thus, provides the same repeatability as FAST.

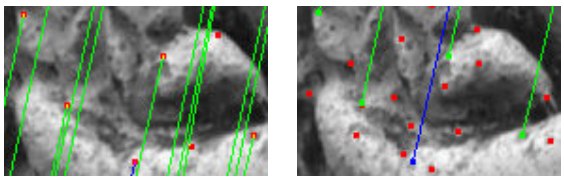
First, we show and discuss an experiment where we compare the performance of different AST masks on noisy and blurry images. Then we evaluate the corner response of different balanced decision trees and, finally, we compare the performance of FAST with our approach.

Evaluation of the Various AST Patterns

As already mentioned in Section 3.1, SUSAN as well as FAST use a circle radius of 3.4 pixels. In [125] it is noted that the mask size does not influence the feature detection as long as there is no more than one feature within the mask. The effect of the mask size of an image operator is well studied for filters with dense masks. Their size affects the smoothing behavior so that larger filters are more robust against noise. The corresponding effect for the AST pattern size has so far not been discussed in



(a) KLT based stereo initialization



(b) details of the putto-scene above

Figure 8.3: The KLT based stereo initialization allows fast sub-pixel-accurate stereo matching. The lower small images show corresponding parts of the putto-scene and show some details of the upper image (left camera) and the lower image (right camera). Blue lines denote matches which are of less confidence due to a failed tracking-convergence.

the literature. While for the dense mask of SUSAN, the same smoothing criteria as mentioned above apply, it is not obvious that large circles have a similar smoothing effect for AST. Therefore, we use eight checkerboard pictures acquired from different viewpoints (see Fig. 8.5) to evaluate the corner response of the AST patterns shown in Fig. 8.4. A checkerboard provides many bright and dark corners of different sizes if viewed from different angles. Further, we add Gaussian blur and noise to determine the performance of these pattern on images of poor quality. For all these tests the same threshold for the brighter- and darker-questions is applied.

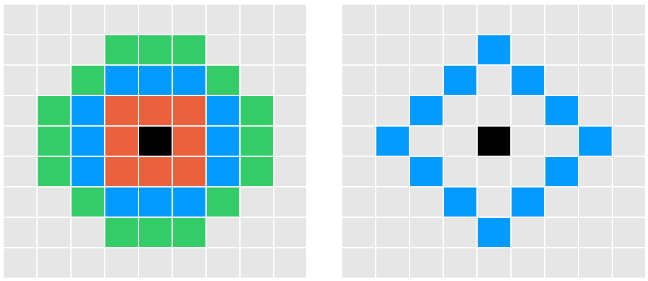


Figure 8.4: Different mask sizes for the AST: a 4 pixels mask (red), a squared and diamond shaped 12 pixels mask (blue, left and right figure) and a 16 pixels mask (green). The black pixel represents the nucleus.

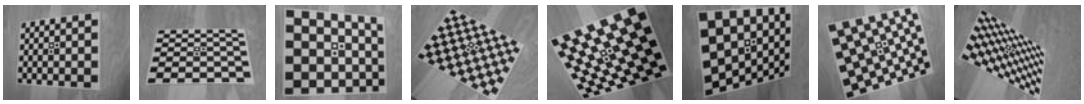
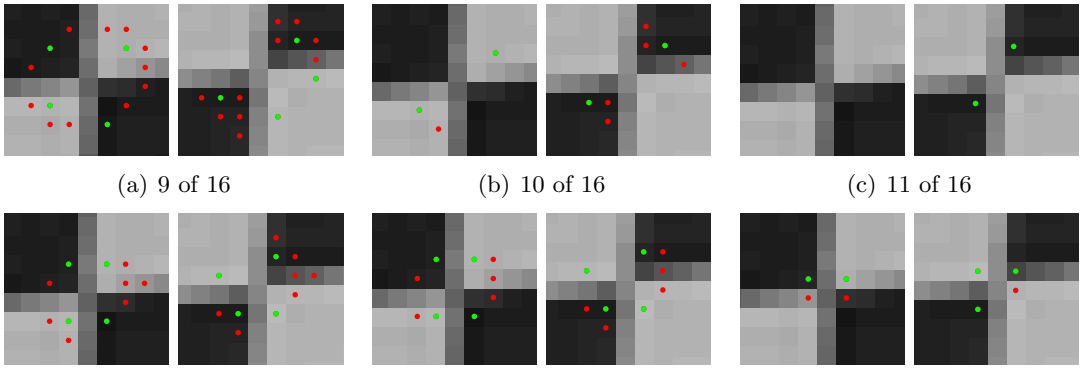


Figure 8.5: Checkerboard data set.

For pattern sizes up to 12 pixels it is possible to compute the optimal path by exploring the six state configuration space as described in Section 7.2.1. The computational resources of conventional computers are not sufficient to find the optimal tree for a 16 pixel pattern within the extended configuration space in reasonable time. Thus, for this size we compute the optimal tree based on the four state space, yielding a ternary decision tree. Before generating the machine code, the tree is splatted as described in [127] to cut off equal branches.

Fig. 8.6 shows the corner response of a 16 pixel pattern with arc lengths of 9, 10 and 11 (arc length 12 is omitted because it does not find any features at these corners), the 12 pixel pattern with a square and diamond shape as well as the 8 pixel pattern. The larger the mask and the larger the arc threshold S, the more features are found.



(a) 9 of 16 (b) 10 of 16 (c) 11 of 16

(d) 7 of 12 diamond (e) 7 of 12 square (f) 5 of 8

Figure 8.6: The corner response for different AST pattern. Detected features are colored in red. The corners after non-maximum suppression are green.

A small arc is more discriminating and yields features only close to the real corner location, which is apparent in Fig. 8.6(c). Large patterns result in multiple responses around a corner location, but they may lie at a distance of about the radius of the mask from the real corner (see Fig. 8.6(a)). Thus, they do not preserve the corner location. They are, hence, slower for two reasons: 1) the processing of a large pattern is of course computationally more expensive, and 2) they need to evaluate many features for non-maximum suppression. Smaller patterns preserve better the locality constraint of a corner. However, in the case of the pattern of size 8, the features are too close, so that a part of them gets lost after non-maximum suppression. Thus, for this size such a post processing is not necessary and should even be avoided to prevent the loss of features, because only single responses are observed at a corner.

For the next experiment the original checkerboard images were modified by adding Gaussian noise (5% and 10%) and Gaussian blur (σ = 0.5, 1.0, 1.5). Fig. 8.7 shows the performance of the different pattern on these images. Here the advantage of the 16 pixel mask with arc length 9, 9(16), becomes apparent. It is more robust against noise and blur. However, the same mask sizes but with larger arcs show a similar drop-off on blurry images as the smaller pattern with similar segment angle. The 16 pixel mask with arc length 12, 12(16), has a similar arc angle as 5(8), while 7(12) has a similar angle as 10(16).

The size of the arc angle controls the repeatability, as shown in [127], and the robustness against blur. The arc length and, thus, the radius of the mask influences the robustness against noise.

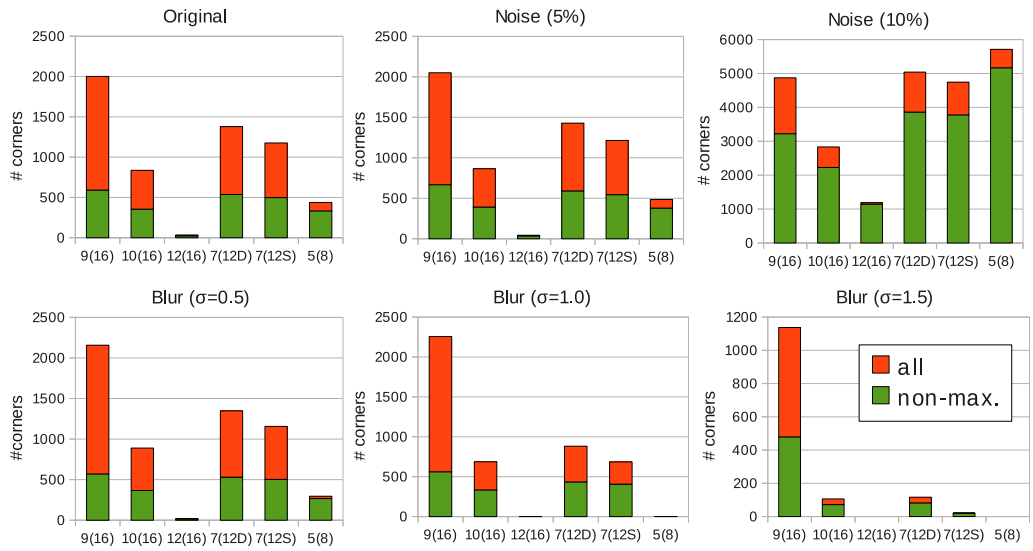


Figure 8.7: These charts compare the corner response of different patterns for blurry and noisy images. To preserve the comparability we use the arc length S and in brackets the mask size for labeling. The red bars show the total amount of features found, while the green bars represent the number of corners after non-maximum suppression. Note that the scales of the charts are not the same.

Corner Response Time

The corner response time of a certain decision tree is evaluated by computing the number of tests (greater-than or less-than evaluations) for all the possible pixel configurations of a mask. To compare the weighting effects of different probabilities of a pixel to be similar (p_s), as described in Section 7.2.1, the pixel configurations are divided into classes representing the number of similar pixels. Fig. 8.8 shows the deviation of the mean and the standard deviation of the corner response time from the minimum of all tests on a class. The trees are built for 12 pixel masks exploring the six state configuration space. For zero or one similar pixels the trees with weight $p_s = 0.1$ and $p_s = 0.01$ perform fewer tests as trees with larger values for p_s. Also the standard deviation of the classes is smaller for these trees. It is apparent that the decision trees can not be balanced significantly due to the strong symmetry of this special constrained twenty questions problem. Besides, the balance of the classes with a large number of similar pixels cannot be modified notably, because the amount of possible configurations decreases drastically for them. Nevertheless, the performance of the adaptive tree is better than if only one tree is used, as we will see in the next section.

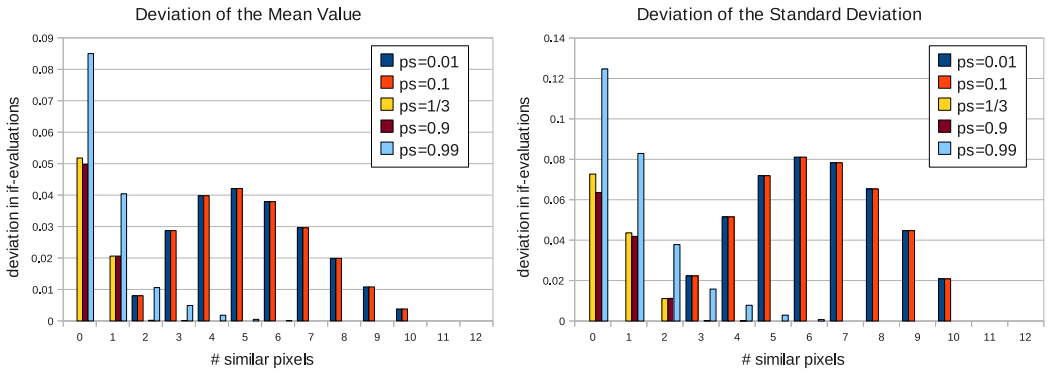


Figure 8.8: This chart illustrates the performance of various decision trees, based on different probabilities for a pixel to be similar (p_s). Each decision tree was tested with all possible pixel combination for the mask.

No performance increase can be achieved for different p_s by exploring only the restricted configuration space, due to the reduced degrees of freedom compared to the full six state configuration space. The limitations of the latter space are also apparent in the performance of the tree M12 (4st), which shows a significantly higher average of tests as M12 (6st) in Table 8.1. This table compares the corner response time for trees of various mask sizes which were built using different methods. The second data row M12 (6st) shows the minimum time of the trees compared in Fig. 8.8 which were

Table 8.1: This table compares the average tests performed for each class of configuration using various mask sizes and different methods to find the best decision tree. From left to the right: mask size 8 exploring the six states configuration space, mask size 12 exploring the six states space, mask size 12 exploring the four states space, mask size 16 exploring the four states space and the ID3-learned decision tree trained on all possible configurations. The probability of all configurations was assumed to be equal for all trees beside for M12 (6st), which represents the minimum tests for all decision trees of Fig. 8.8. These trees were built by exploring the six state configuration space for a mask size of 12 pixels using different weights.

n_s	M8 (6st)	M12 (6st)	M12 (4st)	M16 (4st)	M16 (ID3)
0	5.54	6.53	7.80	8.1528	8.3651
1	5.32	6.17	7.27	7.6485	7.8073
2	5.07	5.82	6.77	7.1948	7.3094
3	4.81	5.48	6.33	6.7893	6.8692
4	4.59	5.19	5.93	6.4277	6.4812
5	4.41	4.94	5.59	6.1044	6.1388
6	4.26	4.74	5.28	5.8144	5.8354
7	4.13	4.56	5.01	5.5529	5.5649
8	4.00	4.41	4.77	5.3160	5.3223
9	-	4.29	4.55	5.1003	5.1033
10	-	4.18	4.35	4.9031	4.9043
11	-	4.08	4.17	4.7221	4.7225
12	-	4.00	4.00	4.5554	4.5555
13	-	-	-	4.4013	4.4013
14	-	-	-	4.2583	4.2583
15	-	-	-	4.1250	4.1250
16	-	-	-	4.0000	4.0000

specialized for different p_s. Thus, these values are achieved using the AGAST, switching between two trees which are optimized for $p_s = 0.1$ and $p_s = 1/3$.

Experiments have shown, that by learning a decision tree based on 120 outdoor images as proposed in [128], only about 87000 pixel configurations out of over 43 million possible ones could be found. Any learned decision tree should, therefore, be enhanced by the missing configurations to prevent false positive and false negative responses. The ID3 based decision tree, learned from all possible configurations with equal weights, has shown to achieve the best corner response of all trees witch were optimized using ID3 and various p_s. Indeed, it yields the identical corner response as the code provided in the FAST sources.[1]

[1] http://svr-www.eng.cam.ac.uk/~er258/work/fast.html

Performance Experiments

All the timing experiments are run on one core of an Intel Core2 Duo (P8600) processor at 2.40 GHz. We are using five images from different scenes[1], shown in Fig. 8.9.

(a) Lab (b) Outdoor (c) Indoor (d) Aerial (e) Medical

Figure 8.9: The scenes used for the performance test are a lab scene (768x288), an outdoor image (640x480), an indoor environment (780x580), an aerial photo (780x582) and an image from a medical application (370x370).

Table 8.2 shows the performance of various AST-decision trees with different mask sizes and built by different methods. Please note that the achieved speed-ups do not only affect the corner detection step, but also the computation of the pixel-score for the non-maximum suppression.

Table 8.2: This table shows the computational time of various AST-decision trees. The value in parentheses, close to the tree names, stands for the mask size which the tree is based on. The specified speedup is relative to the FAST performance. The first value represents the mean speedup for all five images while the value in parentheses shows the maximum speedup measured.

Image	Lab	Outdoor	Indoor	Aerial	Medical	Speed-Up [%]
FAST-9 (16)	1.8867	2.4242	1.8516	2.2798	1.1106	-
OAST-9 (16)	1.5384	2.2970	1.6197	1.9225	0.9413	13.4 (18.5)
AGAST-7 (12)	1.2686	1.9416	1.4405	1.8865	0.8574	23.0 (32.8)
AGAST-5 (8)	0.9670	1.4582	1.3330	1.8742	0.7727	33.0 (48.7)

To compare the performance of our decision trees with the conventional FAST-9 algorithm, we use the code from the FAST sources mentioned above. The FAST and optimal AST (OAST) trees are built based on a uniform probability distribution, which means that the probability for any pixel configuration is the same. This probability distribution yields the trees with the best overall corner response and, thus, the best performance.

[1]The lab scene is provided in the FAST Matlab package at http://svr-www.eng.cam.ac.uk/~er258/work/fast-matlab-src-2.0.zip.

As mentioned in Section 7.2, it is not possible to search for the optimal decision tree for a 16 pixel mask within the complete configuration space in reasonable time on conventional computer. Therefore, the tree is optimized in the four state configuration space and achieves an average speed-up of about 13% regarding FAST-9. For the 12 pixels mask the ideal tree can be found in the six state space and by combining the trees specialized for $p_s = 1/3$ and $p_s = 0.1$ a mean speed-up of about 23% and up to more than 30% can be gained. Using the AGAST-5 decision tree on the 8 pixels mask results in a performance increase of up to almost 50%. Of course, with the drawback of its sensitivity regarding noise and blur as already discussed before in this section.

The C-sources for OAST-9, AGAST-7 and AGAST-5 are available for download at http://www6.cs.tum.edu/Main/ResearchAgast. The trees have been optimized according to standard ratios of memory access times.

8.2 Evaluation of the Pose Estimation Algorithms

We have introduced the Z_∞-algorithm in Section 5.1 and a keyframe-based SLAM method in Section 5.2. The following experiments will analyze and compare these algorithms.

8.2.1 Evaluation of Closed-Form Pose Estimation Algorithms

Before we run experiments on real data we first compare the various variants of the eight-point algorithm as introduced in Section 3.3.1. We also evaluate the motion estimation accuracy for different camera setups, motion directions and rotation angles. We want to get some insights in which algorithm should be preferred for a specific motion, which aperture angle should be used and how the cameras shall be mounted on a robot to allow for the most accurate and robust motion estimation. A more extensive evaluation regarding noise-sensitivity, number of landmarks and aperture angle is provided together with the validation of the error propagation in Section 8.4.

The simulated camera parameters are: 20 features, noise in both images (if not mentioned differently) due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.

Evaluation of Eight-Point Variants

In the following plots (Fig. 8.10 to Fig. 8.13), we compare the different eight-point variants as introduced in Section 3.3.1. Each of these figures evaluates a certain camera aperture angle ($\sim 143°$, $\sim 100°$, $\sim 53°$, $\sim 37°$) and consists of following plots: the first four plots show the effect of the direction of translation on the translation and rotation estimation (only translational motion is simulated). The next two plots show how the rotation estimation error depends on the direction of rotation axis at a constant absolute rotation of 10° (only rotational motion is simulated). For that, the direction of translation and the rotation axis direction varies by 10°-steps between the optical-axis (Z-axis) and the X-axis. The rotation estimation error is specified as absolute angle error which can be computed from the error DCM, $\boldsymbol{R}_\Delta = \boldsymbol{R}_S^T \boldsymbol{R}$, by

$$\theta = \operatorname{acos}\left(\frac{\boldsymbol{R}_{1,1} + \boldsymbol{R}_{2,2} + \boldsymbol{R}_{3,3} - 1}{2}\right) . \tag{8.1}$$

The left column of these six plots is generated by simulating only noise in the second image (as it is the case in feature-tracking) and the right column assumes noise in both images (as in tracking-by-matching approaches).

The last two plots illustrate the median condition number and stability number of the $\boldsymbol{A}$-matrix of the eight-point algorithm (see 3.5) for the direction of translation simulation. The condition number is the quotient of the largest and the second smallest singular value $\frac{\sigma_{A;1}}{\sigma_{A;8}}$ and represents the influence distribution between the strongest and weakest vector of the nullspace which generates the solution - the second smallest and not the smallest is used because we have a homogeneous system of equations. Hence, a small condition number implies a well-conditioned problem, where all vectors in the nullspace have approximately equal influence on the solution. The stability number is the ratio of the second smallest and the smallest singular value and describes how stable the solution is. A small value means that the residuum vector and the smallest nullspace-vector have similar weights and, hence, the solution can be regarded as strongly affected by the error. For the last two plots there is no significant difference whether we simulate noise in one or in both images, thus, we only show one plot for each measure. Further, the Mühlich TLS-FC variant fixes the rank of $\boldsymbol{A}$ to be eight, which means that the condition number is infinitely large and, therefore, it is not plotted for this variant.

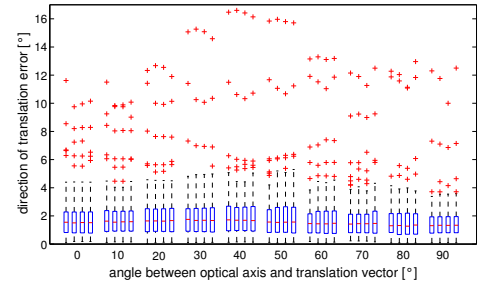
Evaluating the simulation plots 8.10 to 8.13 we can establish following statements for the eight-point algorithm:

- A significant difference between the four eight-point variants becomes only for small aperture angles notable, whereas the Mühlich variants slightly outperform the Hartley variant. The un-normalized, original algorithm proofs to have more difficulties in translation estimation but performs much better in estimating the rotation if no translation is present.

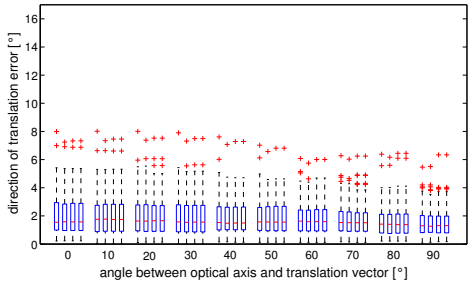
- The advantage of the Mühlich TLS-FC variant becomes apparent for the simulations with noise in only one image (see Fig. 8.13(e)). While the translation estimation is not affected, the rotation estimation is significantly improved for small aperture angles. Small focal lengths, on the contrary, seem to have a slightly reduced accuracy compared to the simpler Mühlich variant.

- In the case of noise in both images, *e.g.* if the correspondences are found by a tracking-by-matching algorithm, the TLS-FC step in the Mühlich approach can be neglected.

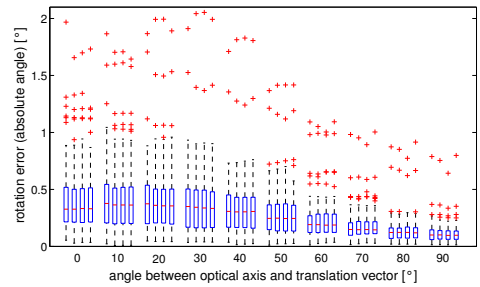
- The normalization of the Hartley and Mühlich variants provides a much better-conditioned motion estimation, especially for large aperture angles - as intended by the inventors.



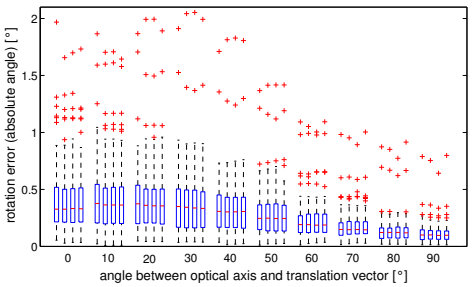
(a) trans. estimation error (trans., 1 img. noise)



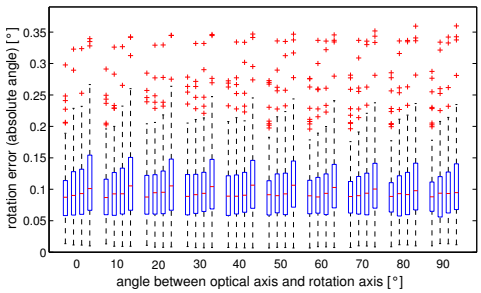
(b) trans. estimation error (trans., 2 img. noise)



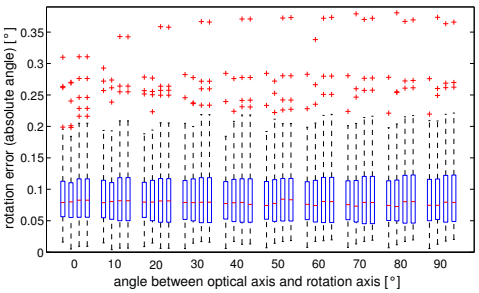
(c) rot. estimation error (trans., 1 img. noise)



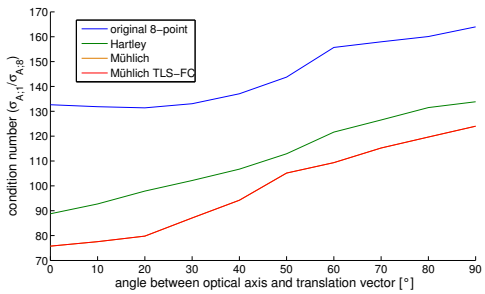
(d) rot. estimation error (trans., 2 img. noise)



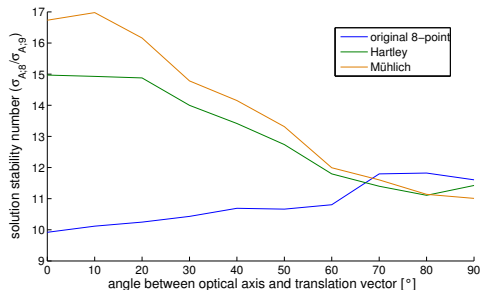
(e) rot. estimation error (rot., 1 img. noise)



(f) rot. estimation error (rot., 2 img. noise)

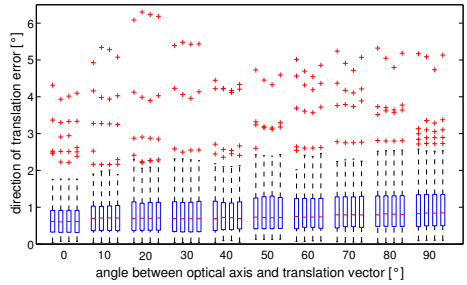


(g) condition number

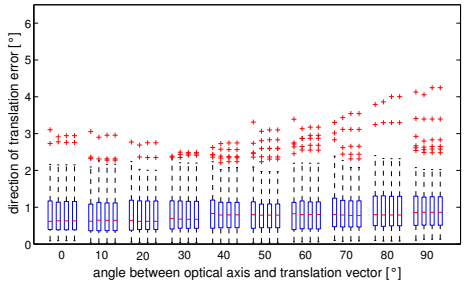


(h) solution stability number

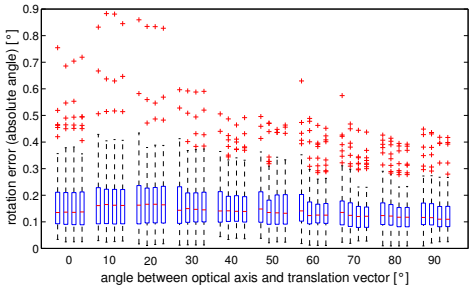
Figure 8.10: This graphic shows the error of translation and rotation estimation of the eight-point algorithm for different directions of translation and of the rotation axis simulating a focal length of 100 px, which is equivalent to an aperture angle of $\sim 143°$. The lower two plots illustrate the condition and stability number as described in the text. The simulation parameters used are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.



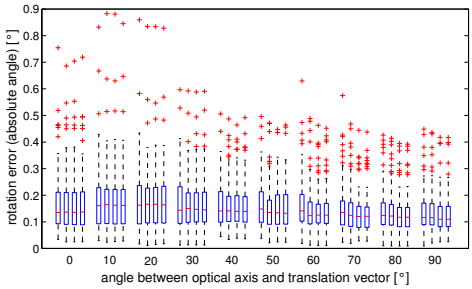
(a) trans. estimation error (trans., 1 img. noise)



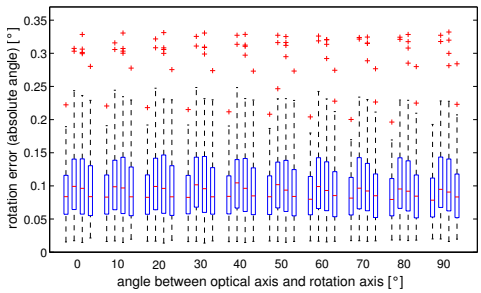
(b) trans. estimation error (trans., 2 img. noise)



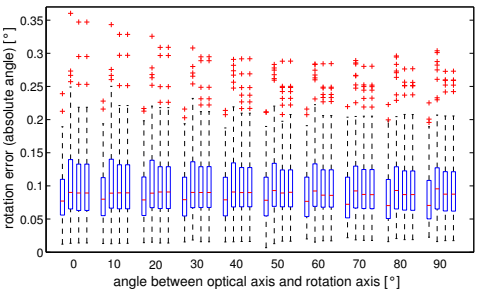
(c) rot. estimation error (trans., 1 img. noise)



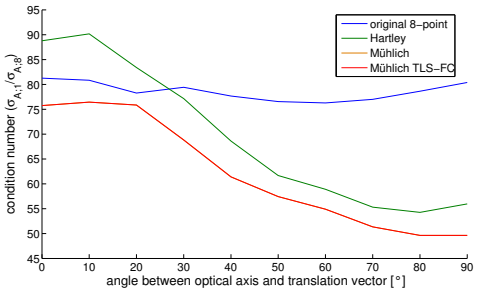
(d) rot. estimation error (trans., 2 img. noise)



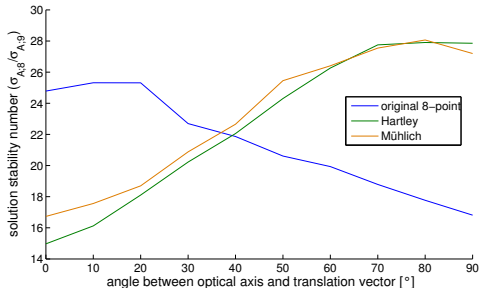
(e) rot. estimation error (rot., 1 img. noise)



(f) rot. estimation error (rot., 2 img. noise)

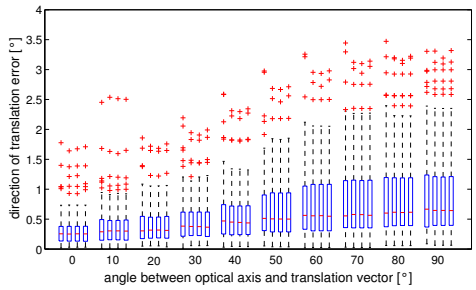


(g) condition number

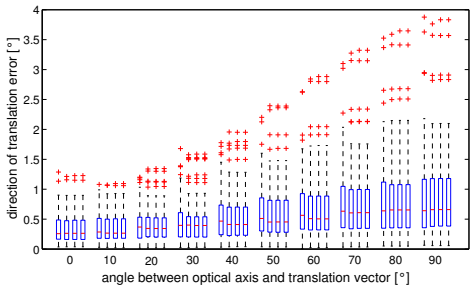


(h) solution stability number

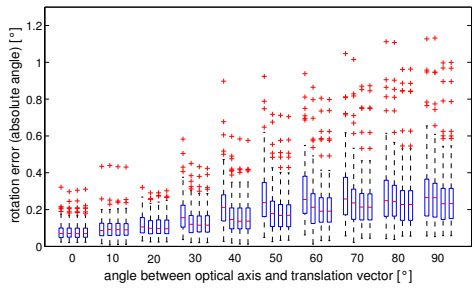
Figure 8.11: This graphic shows the error of translation and rotation estimation of the eight-point algorithm for different directions of translation and of the rotation axis simulating a focal length of 250 px, which is equivalent to an aperture angle of $\sim 100°$. The lower two plots illustrate the condition and stability number as described in the text. The simulation parameters used are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.



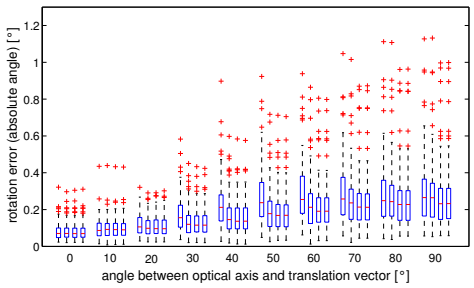
(a) trans. estimation error (trans., 1 img. noise)



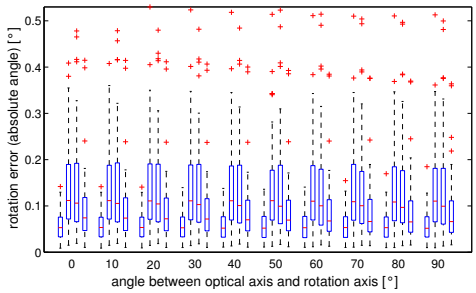
(b) trans. estimation error (trans., 2 img. noise)



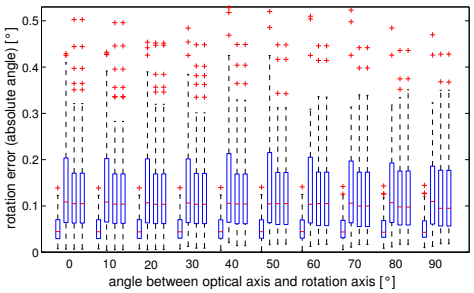
(c) rot. estimation error (trans., 1 img. noise)



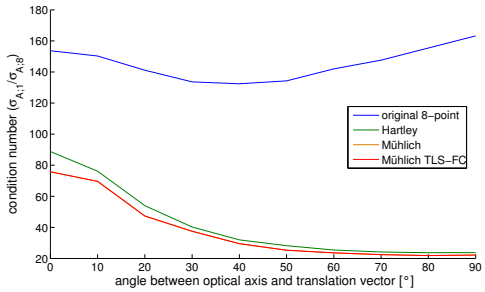
(d) rot. estimation error (trans., 2 img. noise)



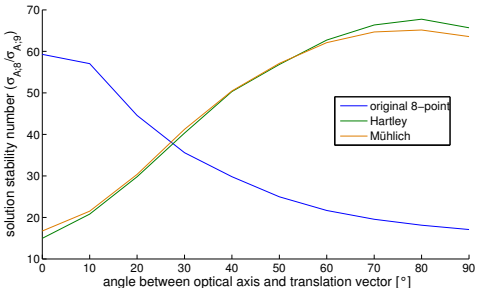
(e) rot. estimation error (rot., 1 img. noise)



(f) rot. estimation error (rot., 2 img. noise)

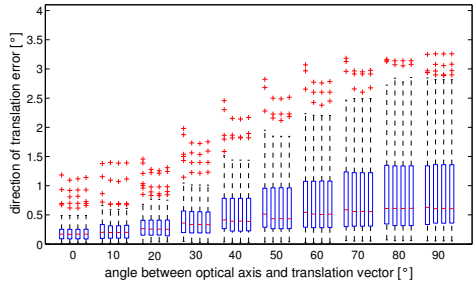


(g) condition number

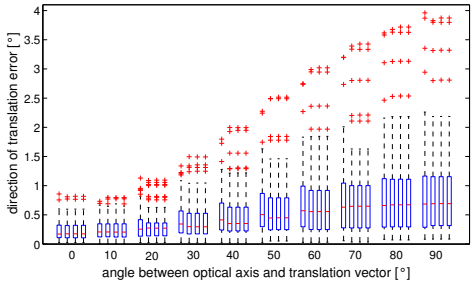


(h) solution stability number

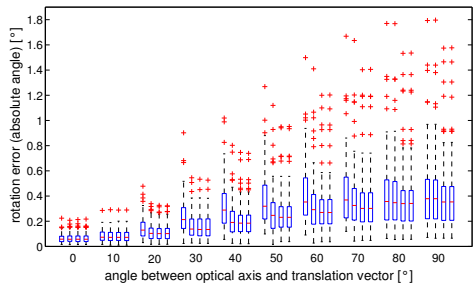
Figure 8.12: This graphic shows the error of translation and rotation estimation of the eight-point algorithm for different directions of translation and of the rotation axis simulating a focal length of 600 px, which is equivalent to an aperture angle of $\sim 53°$. The lower two plots illustrate the condition and stability number as described in the text. The simulation parameters used are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.



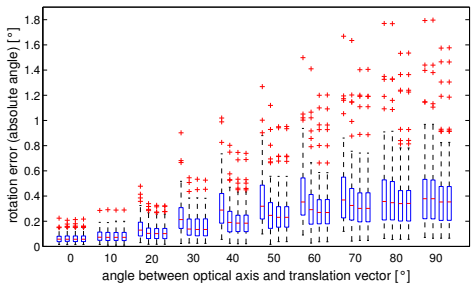
(a) trans. estimation error (trans., 1 img. noise)



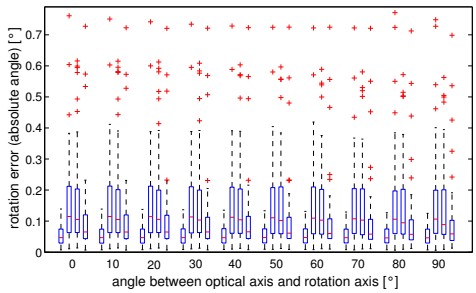
(b) trans. estimation error (trans., 2 img. noise)



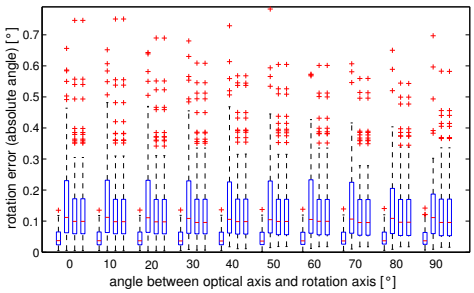
(c) rot. estimation error (trans., 1 img. noise)



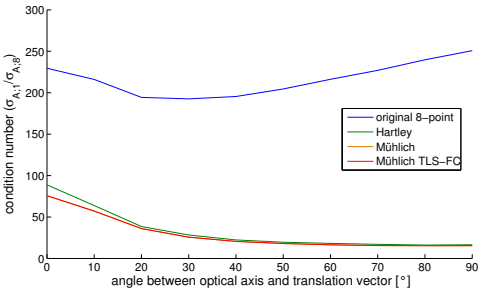
(d) rot. estimation error (trans., 2 img. noise)



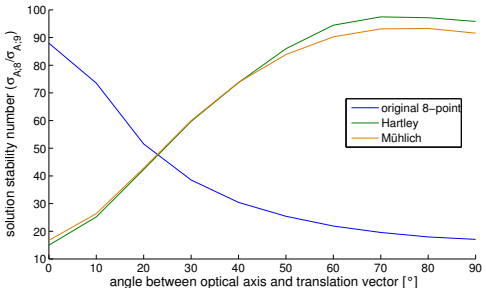
(e) rot. estimation error (rot., 1 img. noise)



(f) rot. estimation error (rot., 2 img. noise)



(g) condition number



(h) solution stability number

Figure 8.13: This graphic shows the error of translation and rotation estimation of the eight-point algorithm for different directions of translation and of the rotation axis simulating a focal length of 900 px, which is equivalent to an aperture angle of $\sim 37°$. The lower two plots illustrate the condition and stability number as described in the text. The simulation parameters used are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.

- The normalization switches the stability of estimating translations. The unnormalized variant has a stable estimation of translations in the direction of the optical axis for short focal lengths, while the normalized ones have a more stable estimation for translations parallel to the image plane. This effect flips for large aperture angles (similar to the conditioning).

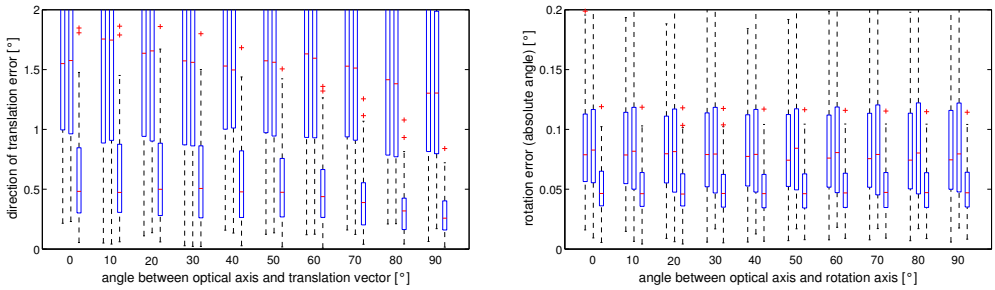
- The feature normalization, as provided by all the eight-point derivatives, is crucial for large focal lengths, but becomes unnecessary for short ones.

Comparison of the Eight-Point and the Z_∞-Algorithm

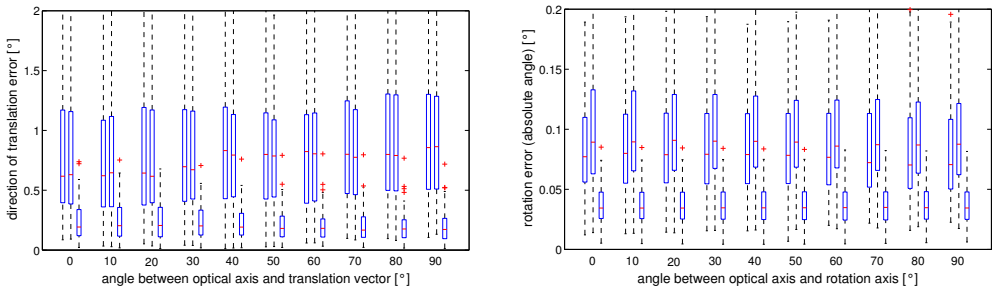
We compare the eight-point algorithm with the Z_∞-algorithm in Fig. 8.14. For that we chose two variants, the original and the Mühlich variant, and run 100 random runs per test-case. The simulated translational motion is 5 m or 10° absolute rotation per frame and the simulated landmarks have a distance of 1 to 50 m.

There are common effects notable which vary depending on the optical system (these effects will also be apparent in Fig. 8.31 and Fig. 8.35): a large aperture angle lessens the accuracy of translation estimation due to a reduced angular resolution. The rotation estimation has a flat optimum around 90° and performs less at small or large focal lengths. Furthermore, the direction of translation becomes more significant for small aperture angles, which provide less accuracy in estimating translations perpendicular to the optical axis. The estimation error becomes also visible in the rotation estimation - apparently, a translational vector field which is parallel to the image plane is much easier (mis-)interpreted as rotation. For large aperture angles the opposite trend is notable. On the other hand, the direction of the rotation axis does not seem to influence significantly the rotation estimation.

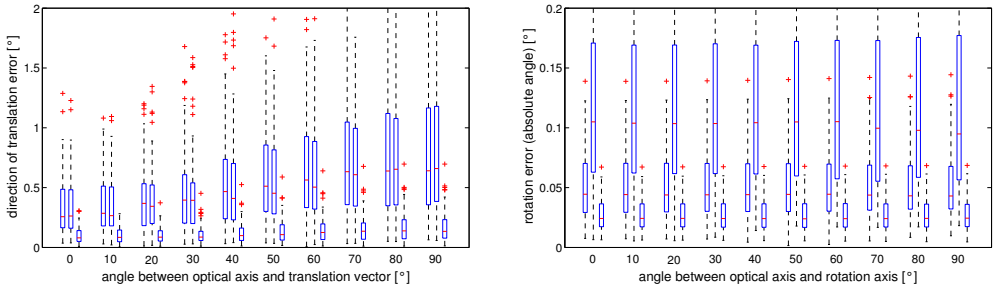
The Z_∞-algorithm shows to follow the same trends as the eight-point algorithm, but it always outperforms it. This can also be seen in Fig. 8.15, where the condition numbers of all variants are compared. The Z_∞-alorithm provides a much better conditioned estimation for any type of motion. This can be explained by the reduced number of degrees of freedom for motion estimation. The closed-form computation of the rotation and the translation always yields the global least-squares optimum, while an outlier rejection is provided by the RANSAC framework. However, the eight-point algorithm or nonlinear optimization techniques have also advantages over the Z_∞-algorithm. They do not require translation-invariant features because rotation and translation are estimated simultaneously. This also allows to disambiguate little translational portions in the optical flow vector which would be interpreted as (biased) noise by the Z_∞-algorithm. Moreover, if the feature distance is stored and optimized along



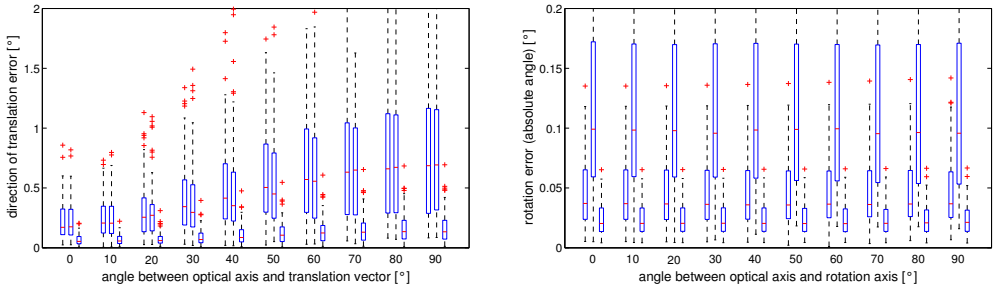
(a) translation estimation error ($f = 100$ px) (b) rotation estimation error ($f = 100$ px)



(c) translation estimation error ($f = 250$ px) (d) rotation estimation error ($f = 250$ px)



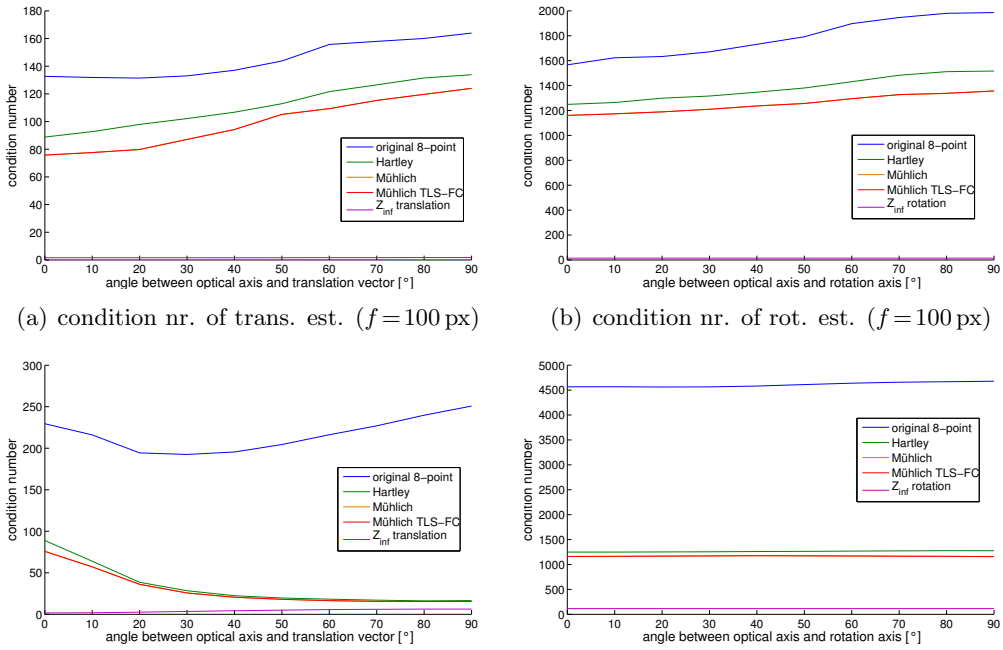
(e) translation estimation error ($f = 600$ px) (f) rotation estimation error ($f = 600$ px)



(g) translation estimation error ($f = 900$ px) (h) rotation estimation error ($f = 900$ px)

Figure 8.14: This graphic shows the error of translation and rotation estimation for various directions of translation and directions of rotation axis and different aperture angles (top to bottom: $f = 100$, 250, 600 and 900 px). Following algorithms are compared (from left to right): the eight-point algorithm (original and Mühlich variant) and the Z_∞-algorithm. Some boxplots of the eight-point algorithm might be cropped for the sake of visibility of the Z_∞-boxplots. However, the full boxplots are shown in Fig. 8.10 to Fig. 8.13.

the image-sequence the estimation accuracy will be increased. The drawback of nonlinear estimation techniques is that they are computationally rather expensive and they may find local optima. In addition, experiments have shown that the convergence of such bundle adjustment methods can be rather slow if also the distance of the features has to be estimated.



(a) condition nr. of trans. est. ($f = 100$ px) (b) condition nr. of rot. est. ($f = 100$ px)

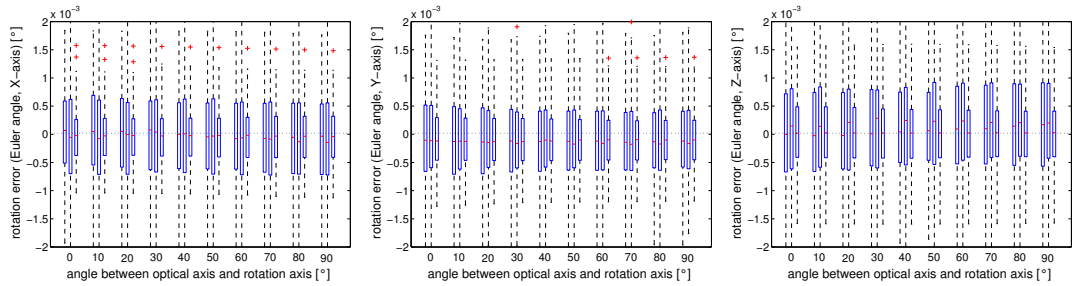
(c) condition nr. of trans. est. ($f = 900$ px) (d) condition nr. of rot. est. ($f = 900$ px)

Figure 8.15: This plots compare the median condition numbers for the eight-point variants and the Z_∞-algorithm. The left column shows the condition numbers for the translation estimation and the right one for the rotation estimation. The upper two plots were generated by simulating a wide angle camera ($f = 100$ px) and the lower ones by a narrow field of view ($f = 900$ px).

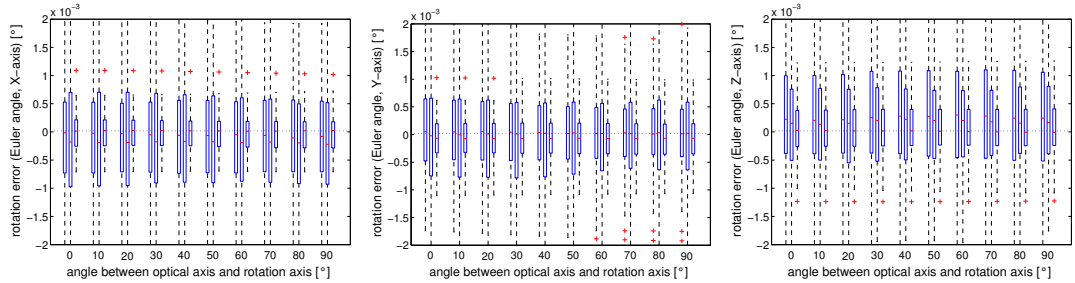
Fig. 8.16 illustrates how the error of the rotation estimation distributes on the different axes. For that we plot the Euler angle error, whereas we assume the small error approximation as introduced in Eq. 2.8, with the angle-axis vector $\boldsymbol{\sigma}$ replaced by the Euler angles $\begin{pmatrix}\phi_x & \phi_y & \phi_z\end{pmatrix}^T$, such that

$$\boldsymbol{\Delta}_{\boldsymbol{R}} \approx \begin{pmatrix} 1 & -\phi_z & \phi_y \\ \phi_z & 1 & -\phi_x \\ -\phi_y & \phi_x & 1 \end{pmatrix} = \boldsymbol{I} + \left[\begin{pmatrix} \phi_x \\ \phi_y \\ \phi_z \end{pmatrix} \right]_\times \quad \Rightarrow \quad \begin{aligned} \phi_x &\approx \boldsymbol{R}_{2,1} = -\boldsymbol{R}_{1,2} \\ \phi_y &\approx \boldsymbol{R}_{1,3} = -\boldsymbol{R}_{3,1} \\ \phi_z &\approx \boldsymbol{R}_{3,2} = -\boldsymbol{R}_{1,3} \end{aligned} . \tag{8.2}$$

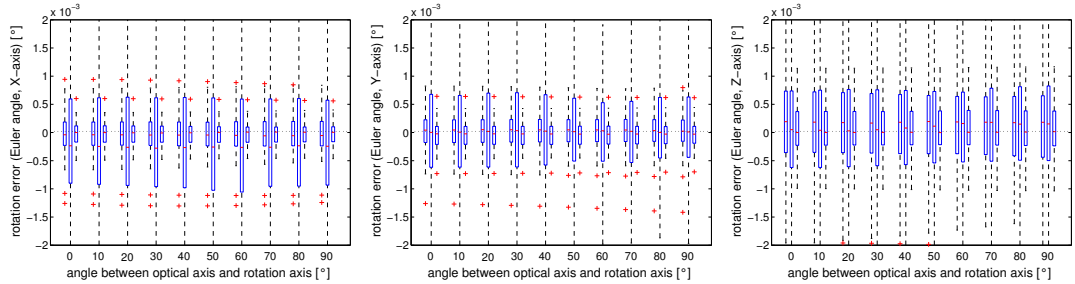
While the Mühlich eight-point variant does not show a significant difference between



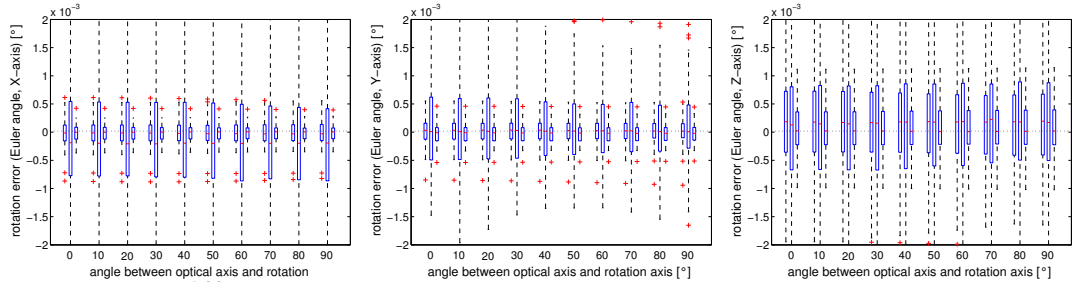
(a) Euler angle errors ($f = 100$ px)



(b) Euler angle errors ($f = 250$ px)



(c) Euler angle errors ($f = 600$ px)

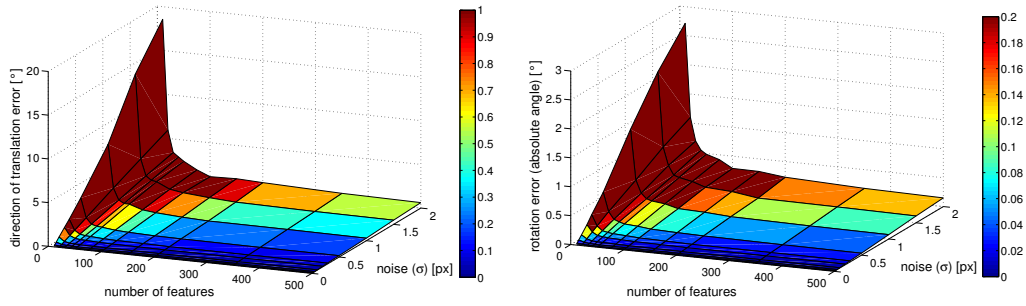


(d) Euler angle errors ($f = 900$ px)

Figure 8.16: This graphic shows the Euler angle error of all three axes, resulting from the eight-point variants and the Z_∞-algorithm for various aperture angles (top to bottom: $f = 100$, 250, 600 and 900 px).

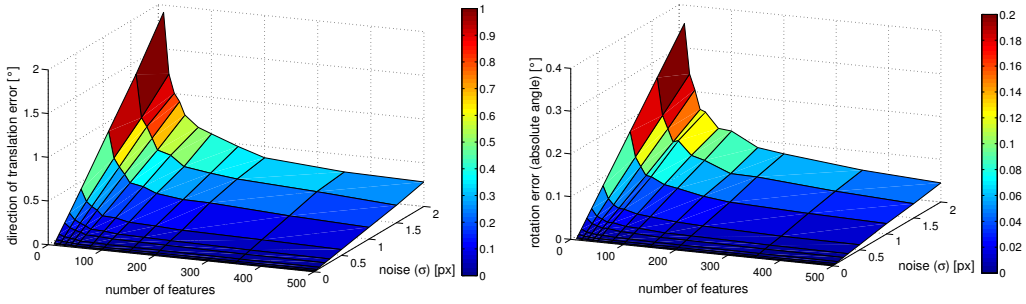
the axes and the various focal lengths, the two other variants (the original eight-point and the Z_∞-algorithm) provide a better performance for the X- and Y-axis, especially for small aperture angles. For large field of views the difference between the axes becomes neglectable. The rotation estimation about the Z-axis is also not influenced by the focal length.

The last graphic of this section, Fig. 8.17, illustrates the relation between the number of features and the detection accuracy. These two tracking properties are plotted against each other for the eight-point Mühlich-variant and the Z_∞-algorithm. The plots show that the detection accuracy becomes insignificant if enough features are used and the tracking errors annihilate each other (keep in mind that we assume an unbiased tracking).



(a) translation estimation error (eight-point)

(b) rotation estimation error (eight-point)



(c) translation estimation error (Z_∞)

(d) rotation estimation error (Z_∞)

Figure 8.17: This graphic plots the influence of the detection accuracy and the number of features against each other. The eight-point algorithm used is the Mühlich-variant. The simulation parameters are: noise in both images, 600×600 pixel resolution and 250 px focal length ($\sim 100°$ aperture angle).

8.2.2 Experimental Evaluation of the Z_∞-Algorithm

In the previous section, we have seen that the Z_∞-algorithm should be preferred, in case a separation of the landmarks into translation-variant and -invariant features is feasible. In the following section, we will present a comparison with an IMU to validate the approach. Fig. 8.18 illustrates the Z_∞-steps. We use the KLT implementation as introduced in Section 7.1 to find feature correspondences for the evaluation of our algorithm. Fig. 8.18(a) shows an example output of the tracker with 500 features. However, also global trackers, like SURF have been successfully tested as described later in this section.

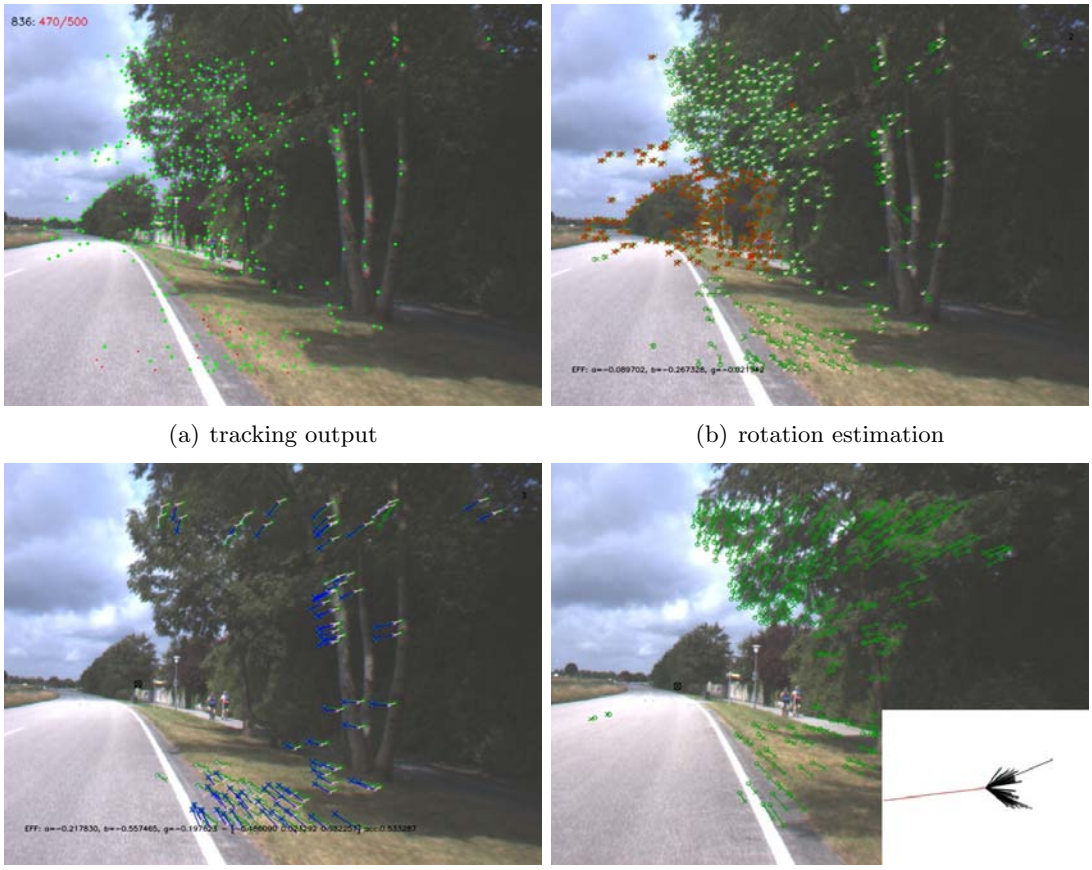
To evaluate the Z_∞-based motion estimation we compare it to the measurements of an IMU. For that we use an "MTi" IMU from Xsens[1] which is mounted on a Guppy F046C camera from Allied Vision. The IMU provides the absolute rotation, based on the Earth's magnetic field, three gyroscopes, and the acceleration of the device. The camera images and the IMU data are acquired with 25 Hz. Before the data can be evaluated the IMU-camera alignment has to be determined as described in Section 4.2. Fig. 8.19[2] visualizes two data sets acquired while holding the camera-IMU system out of a car. The left image shows a sinuous trajectory on a slightly left bent street. The right image describes the path through a turnaround. Even though we compute the relative motion from image to image and do not stick to a reference, like in keyframe techniques, the Z_∞-method shows less drifting than the Kalman filtered output of an Xsens IMU. The IMU apparently suffers from the centrifugal forces, which, once integrated, they are not removed from the velocity vector anymore and cause the estimated pose to drift. Fig. 8.20 shows the angles measured along the axis perpendicular to the street, when driving the sinuous line illustrated in the left image of Fig. 8.19. The measurements are consistent which validates the Z_∞-based rotation estimation.

SURF based Z_∞-Pose Estimation

The Z_∞-algorithm has also been successfully tested with the global tracker SURF. Like every global tracker, SURF also provides much more outlier and the accuracy is in general worse than with local trackers. We run SURF on 10 megapixel pictures of the church in Seefeld, Germany, which were acquired by an octocopter. Fig. 8.21 shows the rotation and translation estimation result for one such image pair. The images were acquired with large intermediate distances. Hence, SURF features have been extracted and the rotation and also translation was computed for each feature set. Despite the

[1] http://www.xsens.com

[2] The pictures at the right hand-side are "Google maps" screen-shots http://maps.google.com.



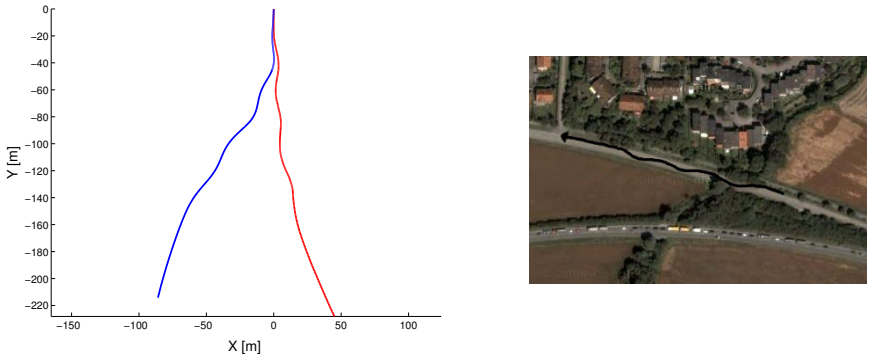
(a) tracking output

(b) rotation estimation

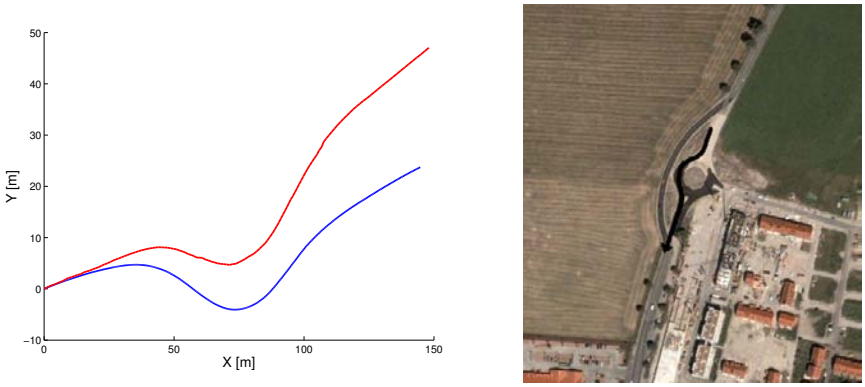
(c) translation estimation

(d) obstacle detection

Figure 8.18: Visualization of the Z_∞-steps. The landmarks are marked green if they could be tracked from the last image or red if they were lost and replaced by new features to track (a). The optical flow is represented as green vectors, where the original position of the feature is marked by a circle and the tracked location by a cross. The results of the rotation estimation are colored red. The endpoints of the optical flow vectors are rotated back according to the computed rotation – they are marked as crosses too. Thereby, translation-invariant features are represented by red vectors, while the outlier of the rotation computation are magenta (b). The translation estimation is illustrated blue. Similar to the rotation visualization, the vectors, which were used for epipole estimation, are dark blue, while the outlier and, therefore, wrong tracked features are light blue. The black star (circle and cross) represents the computed epipole (c). A result of the obstacle avoidance is shown in (d) and consists of two parts: the main image shows the optical flow vectors which are used for obstacle detection and the small sub-image shows the computed obstacle map. The vectors are mapped regarding their angle to the calculated epipole and a red line illustrates the suggested swerve direction (swerve angle) as it is explained in the text.



(a) driving a sinuous line



(b) passing a turnaround

Figure 8.19: Two trajectories measured by the Z_∞-algorithm (red) and an IMU (blue). In the pictures at the right-hand side the driven trajectories are sketched in black.

high percentage of outlier and the poor accuracy, the Z_∞-algorithm has successfully processed the image sequence. No ground truth could be provided for this experiment but the estimated motion is plausible for a similar inlier-percentage as used for KLT-matches which implies a reliable motion estimation.

Obstacle Avoidance

Potential obstacles can be detected by evaluating the translational component of the optical flow $\boldsymbol{d_t}$. The closer or the faster an object is moving respective to the camera, the larger the optical flow vectors become [107]:

$$\|\boldsymbol{d_t}\| \propto \frac{\|\boldsymbol{v}\|}{z^2} \,. \tag{8.3}$$

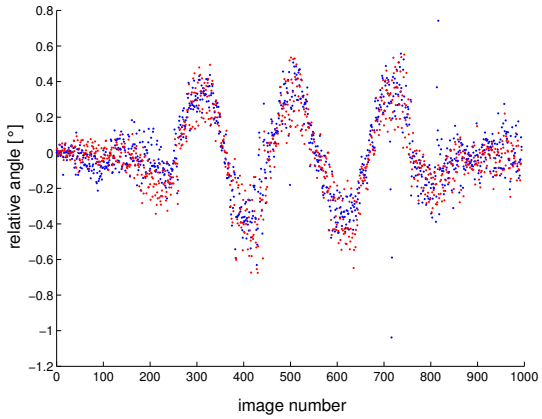


Figure 8.20: This image shows the relative angle measured during the same run as in the left image of Fig. 8.19. The Z_∞-rotation estimation is red, while the IMU data is blue.

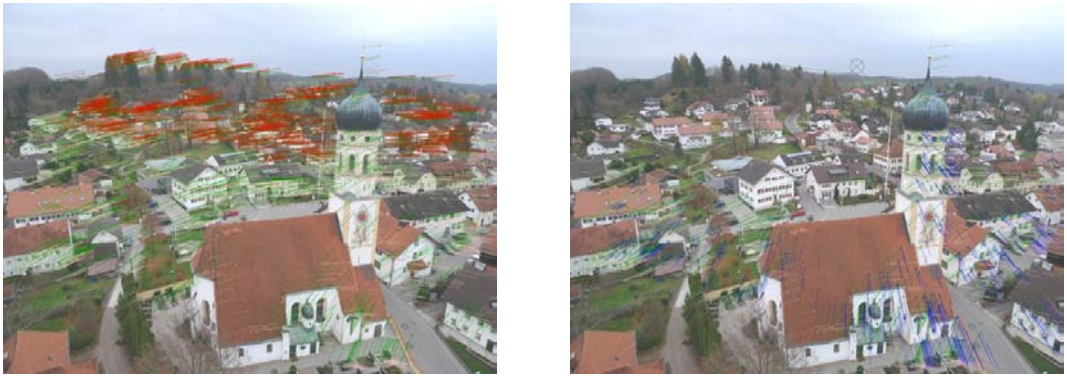


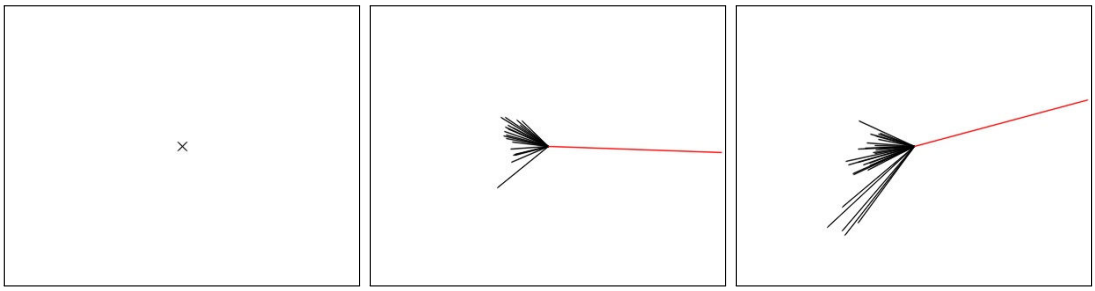
Figure 8.21: Motion estimation based on SURF features.

The velocity and the distance cannot be separated without any knowledge about the environment. Hence, the length of a translational optic flow vector expresses the *time-to-impact* (or *time-to-collision*) and a specific threshold on the length of the optical flow vectors has to be specified, which defines a dangerous object. This threshold depends on the camera characteristics (resolution, focal length, *etc.*) and the application (how much time is required to stop or plan a new trajectory). This concept works also in dynamic environments, where fast moving objects are earlier detected as dangerous obstacles than slow ones.

To allow for a reactive behavior which can be directly coupled to the robot control, we use the calculated epipole and, thus, the direction of translation as reference for the obstacle avoidance strategy. Once the directions in which the dangerous objects lie are computed, the middle of the largest free segment is proposed as swerve direction.

This simply means for a robot which operates in 3D space to steer in the suggested direction in order to avoid the objects. The obstacle maps in Fig. 8.22 show the angle of dangerous objects respective to the epipole, their level of danger as length (inverse time-to-impact) and a suggested swerve direction for obstacle avoidance. The images where acquired by a Pioneer3-DX equipped with a Marlin F046C camera as shown in the experiments of Section 8.3.

(a) Input images for the obstacle detection: the black star is the epipole and the green vectors are the optical flow.



(b) These are the corresponding obstacle maps to the images above - the red line is the suggested swerve direction for obstacle avoidance.

Figure 8.22: Within the Z_∞-algorithm the rotational component of the optical flow is removed and translation-dependent features are identified. These can be used to realize a reactive obstacle avoidance.

Processing Times

Fig. 8.23 shows the processing times of a C++-Implementation of the algorithm on a 1.6 GHz Intel Core Duo processor T2300.[1] The most important parameters influencing the processing performance are the maximum number of iterations for rotation estimation (40), the maximum number of iterations for translation estimation (20) and the maximum number of steps per translation estimation (5). The processing is only single-threaded and the code gives still some space for significant performance optimiza-

[1] The time was measured with the *gprof* profiling tool.

tion. Nevertheless, the algorithm requires in average only 4.13 ms for all Z_∞-steps. The rotation estimation processes most features and, hence, requires also the most time.

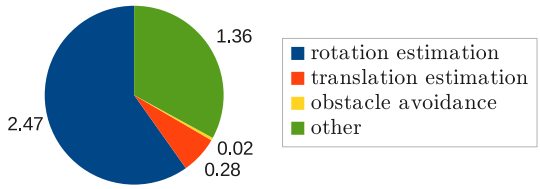


Figure 8.23: The computation time for each module of the Z_∞-algorithm in milliseconds.

8.2.3 Experimental Evaluation of the Keyframe-Based VSLAM

We mounted a stereo camera system on the wrist of a robotic manipulator, a KUKA KR16[1], to compare the accuracy improvement of the RVGPS method, introduced in Section 5.2.1, relative to the conventional VGPS. The robot's kinematics is used as a ground truth with a relative error of 0.1 mm in position and 0.1° in rotation. Based on this reference, the accuracy of the visual navigation algorithm can be evaluated.

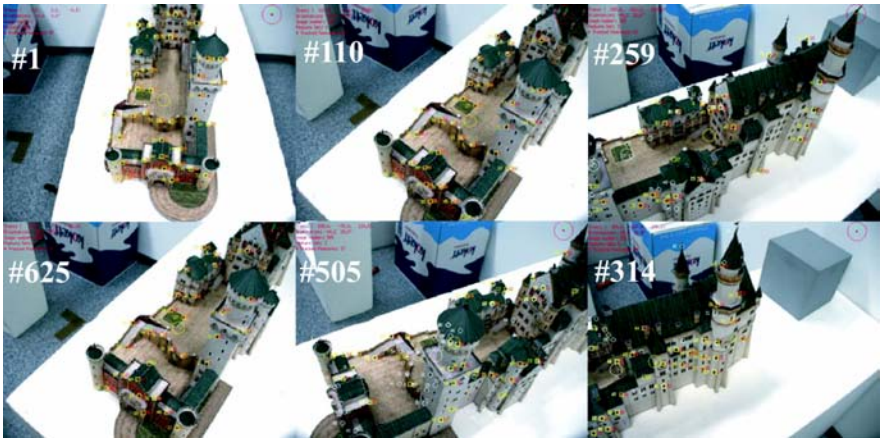


Figure 8.24: The motion is from one side of the castle (image #1) to the other (image #350), and then back again to the front in image #710.

In the experiment, the sensor is moved half around a model of castle Neuschwanstein shown in Fig. 8.24 and back again. 710 images were processed on a total path of 125 cm. The orientation changed up to 55°. Fig. 8.25 shows the executed trajectory and compares the motion estimation result with the inverse kinematics of the KUKA robot. The images have been processed twice: with the robustification enabled and disabled. The errors of the rotation and translation are illustrated in Fig. 8.26. The blue dots represent the RVGPS output and the red dots VGPS without robustification.

Disregarding the outliers in the performance charts, the translational error increases up to 4 mm and the maximum accumulated rotational error is 0.5°. The outliers are the result of bad, drifting features which are detected by RVGPS and removed from further processing.

In the run with standard VGPS, the orientational error remains almost the same, while the translation estimation accumulates an error up to 20 mm. This can be explained by the nature of such an iterative gradient descent method as VPGS: A difference in the measured landmarks and the reprojection can often be lessened by zooming

[1] http://www.kuka.com

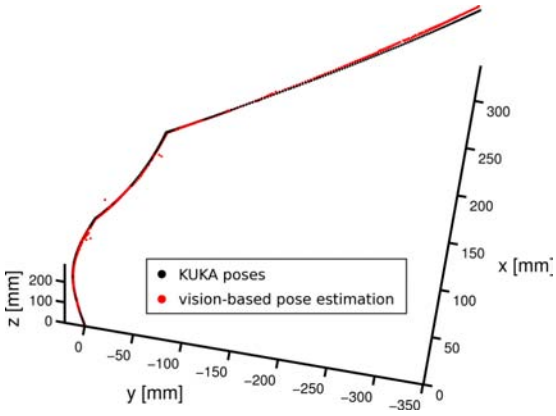
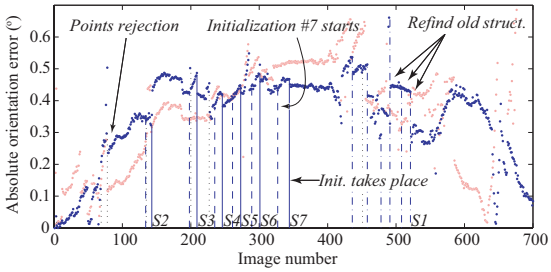
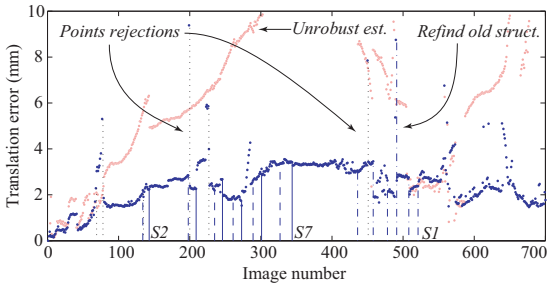


Figure 8.25: This figure shows the executed trajectory and compares the result of the keyframe-based SLAM (red) with the inverse-kinematics of the KUKA robot (black).



(a) rotation estimation



(b) translation estimation

Figure 8.26: Residual orientation and translation error with respect to the *KUKA* manipulator. The vertical lines mark the change of sets and the "Sx" denote the current number of set. *Blue*: using RVGPS. *Pink*: using VGPS. VGPS accumulates an error up to 20 mm at some point, which is not shown for the sake of visibility.

out the camera. The algorithm simply follows the gradient on the optical axis of the camera to diminish the error.

Moreover, the effect of the keyframe management becomes apparent in both runs. The error which is accumulated at each switch of set on the way to the end of the castle is again removed moving back to the initial position. The old feature sets are refound and used for the pose estimation on the way back. The error accumulated between the changes is removed again and the final residual error can be explained by the inaccuracies in the robot kinematics and tracking drifts.

Application related ground-truth experiments are shown in Section 9.1.

8.3 Comparison of the Z_∞-Algorithm and Keyframe-Based VSLAM

In this experiment, Z_∞-based visual odometry estimation is compared to stereo-initialized keyframe-based VSLAM. The latter method has been designed for short-distance applications as 3D-modeling (see Section 9.1), but it has been shown to be also accurate and reliable for mobile robot navigation (see Section 9.2). In contrast, the Z_∞-algorithm requires far distant landmarks which are translation-invariant in order to allow for rotation estimation.

In this experiment, two Marlin F046C cameras from Allied Vision[1] have been mounted on a stereo rig with 9 cm baseline on the top of a Pioneer3-DX from MobileRobots Inc.[2] (Fig. 8.27). The robot has been programmed to follow a circular path with 3 m diameter. During this run 1920 stereo image pairs were acquired. The cameras have been slightly tilted to the ground to increase the close areas in the images which are required for translation estimation, while keeping the horizon and the structures at infinity well visible. Although robot odometry is in general rather imprecise, the Pioneer has been able to close the circle with only a few centimeters of error. Based on the result of this experiment we will discuss some drawbacks and advantages of the motion estimation algorithms.

Figure 8.27: Experimental setup: a Pioneer3-DX equipped with two Marlin cameras.

Fig. 8.29(a) shows the trajectory computed by the VSLAM approach. It is apparent that there is no bundle adjustment applied: wrong initialized features lead to a wrong pose estimation, but are usually immediately removed by the M-estimator in RVGPS. However, if there are too many wrong features, it can last some images until they get

[1] http://www.alliedvisiontec.com
[2] http://www.mobilerobots.com

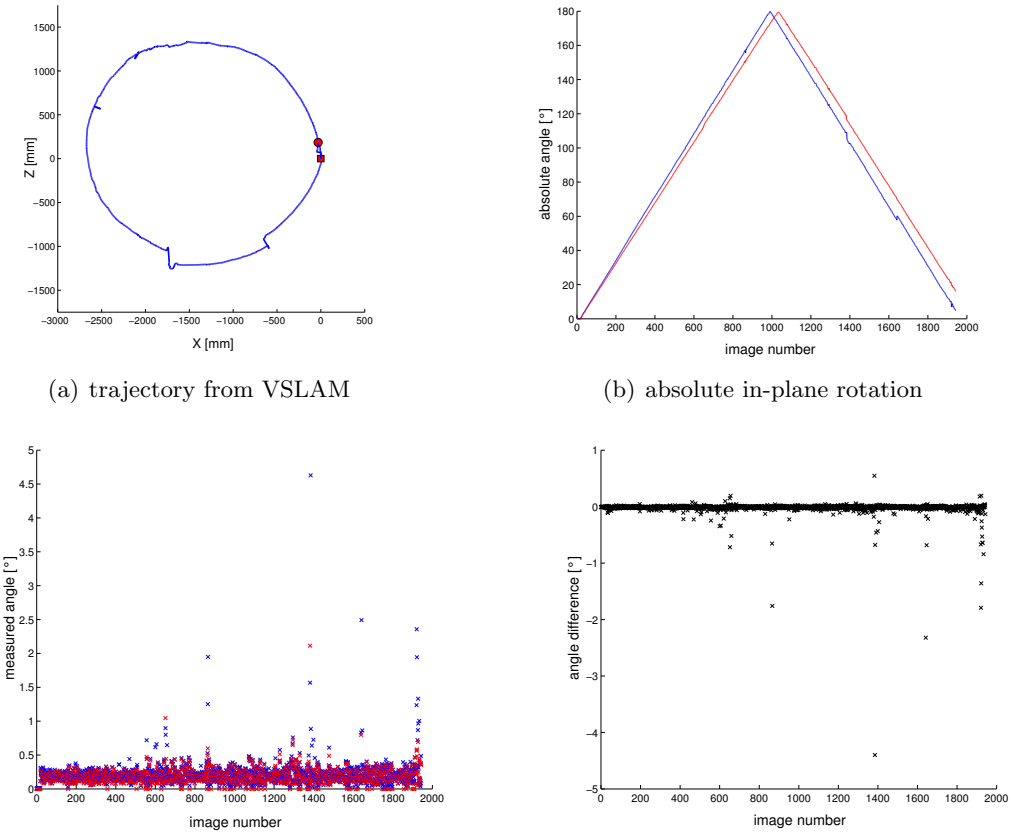
removed from the current feature set. Such a behavior is visible in the lower half of the circle, where the computed trajectory leaves the circular path and jumps back again after a while. Such outlier could also be suppressed by a probabilistic filter, like, *e.g.*, a Kalman filter.

To provide an accurate stereo initialization with this baseline, all landmarks have to lie within a range of approximately 10 m. Therefore, this image sequence is ill-conditioned for RVGPS-based localization due to several facts: only features in the lower half of the image can be used for pose estimation, due to the large rotation the feature sets are leaving the field of view quickly and the ground on which the robot is operating has only little structure which eases miss-matches and the loss of features during tracking (see Fig. 8.28(a)).

(a) VSLAM-screenshot (b) Z_∞-screenshot

Figure 8.28: The screenshots illustrate the difficulties for both methods in estimating the robot's motion in this environment. VSLAM only uses close landmarks which can be initialized by stereo-triangulation. Such features are rare on the homogeneous textured ground. This is also a problem for the translation estimation of the Z_∞-algorithm. The tiny translational motion component is often misleadingly interpreted as rotation. This becomes visible by the red features on the ground.

The large rotational component of the motion compared to the amount of translation prevents the Z_∞-algorithm to determine the direction of translation accurately. The small translation results in a z_∞-range of 10 cm per image. Therefore, the translation can only be evaluated after several images, when it gets accumulated to a measurable length. Furthermore, the small translational component in the optical flow vectors yields the features to be tested as translation-invariant and, therefore, to be misleadingly used for rotation estimation and to be rejected from further translation estimation (see Fig. 8.28(b)). A large number of images for translation accumulation increases also the probability of the features to get lost by the tracker – this is also favored by the poor structured ground.



(a) trajectory from VSLAM

(b) absolute in-plane rotation

(c) relative in-plane rotation

(d) difference between rotation estimates

Figure 8.29: *a)* The computed trajectory by the VSLAM algorithm. The red square is the starting point and the red circle the endpoint of the path. *b)* Comparison of the absolute angle estimated by the Z_∞ (red) and the VSLAM (blue) algorithm. *c)* The relative angles about the Y-axis are plotted, which are computed by the Z_∞- (red) and by the VSLAM-algorithm (blue). *d)* This plot shows the difference between the $Z_\infty$and VSLAM rotation estimates along the Y-axis. Only a few outlier can be identified, which are detected by the M-estimator in RVGPS. Hence, they have no effect on the final result.

Fig. 8.29(b) illustrates the computed angle of the Z_∞ and VSLAM algorithm and Table 8.3 shows some key-values regarding the result. The VSLAM method proves its accuracy with only 3.8° absolute error, even though the ill-conditioned images. On the other hand, Z_∞ accumulates an absolute error of 16° during the sequence. This seems to be a large error before taking into account two facts: The difference between both rotation estimations is not zero-mean as expected. This can be easily explained reminding again the problem to detect translation-variant landmarks at such a small translational motion. Close landmarks are also used for rotation estimation, as shown in Fig. 8.28(b), and because the translation is always done in the same direction on a circular path, the estimation error corresponds to a measurement bias.

Table 8.3: Keyvalues of the comparison between Z_∞ and VSLAM for the circle experiment. The absolute rotation error for all three axes, the number of error accumulations of each method, the average rotational error about the Y-axis for each error accumulation and the average rotation per frame are listed.

	abs. X-rot. error (°)	abs. Y-rot. error (°)	abs. Z-rot. error (°)	no. of error accumulations	aver. Y-rot. error per acc. (°)	average rotation (°)
VSLAM	3.8	2.87	-0.01	116	0.03	0.21
Z_∞	16.0	-1.14	1.37	1920	0.008	0.19

The other advantage of the VSLAM approach is its bias reduction strategy. As explained in Section 5.2, the pose estimation is based on initialized feature sets. The distance of each feature is estimated by stereo triangulation and is not updated anymore. The poses are always calculated respective to the image, where the landmarks were initialized and a bias is accumulated only at keyframe-switches. The memory consumption grows continuously as with all VSLAM approaches. That prevents the VSLAM algorithm to be used on platforms operating in large workspaces like outdoor mobile robots. In this experiment 116 keyframes were necessary. In contrast, the Z_∞-method computes only a vision-based odometry from image to image and, hence, does not require a keyframe management. This makes it adequate for outdoor applications without environment restrictions and makes it applicable to resource-limited platforms. As drawback it suffers from an error accumulation like all the non-map-based algorithms without a global reference. Considering the relative error in the sense of error per accumulation step, we achieve a much smaller error for the Z_∞-algorithm than the VSLAM approach. For the Z_∞-method with 1920 steps, only an average error of 0.008° arises despite the bias-prone conditioning of the experiment. This is almost four times less than for the VSLAM solution, where 116 steps yield an error of 0.03° each. The larger error can be explained by the KLT-tracking drifts between the

keyframes. The computed angles for each frame are plotted in Fig. 8.29(c). Only a few outlier can be identified, whereas VSLAM seems to have more outliers than Z_∞, but which do not worsen the overall performance of it due to its keyframe-based nature. Fig. 8.29(d) shows again the differences between the estimated rotations of both methods for each image. The mean difference corresponds to the explained bias of $-0.02°$ with a standard deviation of $0.14°$.

8.4 Evaluation of the First Order Error Propagation

We derived a first order accuracy prediction for the eight-point algorithm and the Z_∞-algorithm in Section 6. The following experiments will tackle and compare these error propagation formulas on synthetic data.

We use MATLAB simulations, to prove the quality of the first order error estimation and show how it correlates with the real error by varying the simulation parameters. Following parameters are evaluated here: aperture angle, number of features N, and feature detection accuracy (noise) which is equivalent to varying the resolution. If not mentioned differently we are using 100 runs per experiment whereas the landmarks are within a range of 50 m and we simulate a translation of 5 m and a rotation of up to 5°, both in random directions.

8.4.1 Error Propagation for the Eight-Point Variants

In Section 6.1, we derived formulas to predict the accuracy of the various introduced eight-point variants based on the tracking accuracy and/or noise assumption of the used feature correspondences. In the following, we evaluate these formulas on synthetic data, whereas we also vary some crucial imaging and feature tracking parameters. All four eight-point implementations introduced in Section 3.3.1 are used.

For the error propagation analysis we used $\boldsymbol{A}$ as computed in Eq. 3.5. However, in practice, we can use only the error-corrupted $\tilde{\boldsymbol{A}}$, because it is the value we measure. However, from a slightly different point of view, we can regard $\boldsymbol{A}$ as noise-corrupted matrix by adding $-\boldsymbol{\Delta}_{\boldsymbol{A}}$ to the matrix $\tilde{\boldsymbol{A}}$. Thus, the error is the deviation of the true solution from the noise-corrupted one. This observation justifies the use of $\tilde{\boldsymbol{A}}$ to estimate the errors.

To compare the real and the estimated error as a scalar we compute the errors of the estimates $f_{\boldsymbol{E}}$, $f_{\boldsymbol{R}}$ $f_{\boldsymbol{t}}$ and the estimated errors $\hat{f}_{\boldsymbol{E}}$, $\hat{f}_{\boldsymbol{R}}$ $\hat{f}_{\boldsymbol{t}}$ by normalizing the Frobenius norm of the error by the Frobenius norm of the respective matrix

$$f_{\boldsymbol{E}} = \frac{1}{\sqrt{2}} \left\| \boldsymbol{E}_S - \bar{\boldsymbol{E}} \right\|_F \quad , \quad f_{\boldsymbol{R}} = \frac{1}{\sqrt{3}} \left\| \boldsymbol{R}_S - \boldsymbol{R} \right\|_F \quad , \quad f_{\boldsymbol{t}} = \left\| \boldsymbol{t}_S - \boldsymbol{t} \right\|_F \tag{8.4}$$

with the subscript $_S$ denoting the simulated values. Furthermore, an equivalent to the normalized Frobenius norm can be computed from the covariance matrices by

$$\hat{f}_{\boldsymbol{E}} = \sqrt{\frac{\mathrm{tr}(\boldsymbol{\Gamma}_{\boldsymbol{E}})}{2}} \quad , \quad \hat{f}_{\boldsymbol{R}} = \sqrt{\frac{\mathrm{tr}(\boldsymbol{\Gamma}_{\boldsymbol{R}})}{3}} \quad \text{and} \quad \hat{f}_{\boldsymbol{t}} = \sqrt{\mathrm{tr}(\boldsymbol{\Gamma}_{\boldsymbol{t}})} \, . \tag{8.5}$$

Similar results were achieved by comparing the simulated and estimated absolute error angles $\|\boldsymbol{d}'_{\boldsymbol{R}}\|$ as proposed in Eq. 6.27.

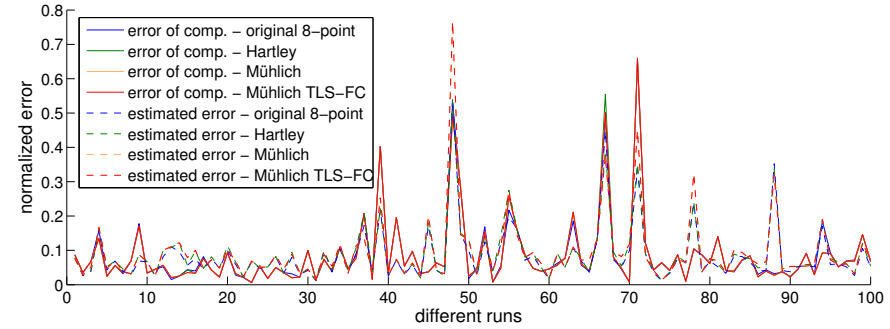
Fig. 8.30 illustrates the error between the computed and the simulated motion parameters (continuous lines) and the estimated error (dashed lines) for 100 random runs. The plots proof that the estimated errors correlate.

The quality of the first order error estimation regarding the aperture angle is denoted in Fig. 8.31. It shows that the error can be robustly propagated throughout the whole range of aperture angles of a projective camera and that the relative error remains approximately the same. The boxplots on the left hand side show the relative difference of the estimated error $\hat{f}$ and measured error f in percentage of f according to following formula: $\frac{f-\hat{f}}{f} \cdot 100$. The boxplots are grouped by the order: original eight-point, Hartley-, Mühlich, and Mühlich TLS-FC variation. However, the Mühlich variant and the Mühlich TLS-FC variant are overlapping so that only the latter one is visible. Please note that for the sake of visibility not all outliers are shown in the boxplots – *e.g.*, for the aperture angle plots the translation estimation has outliers down to approximately -8000% for wide aperture angles and, thus, seems to be more outlier-prone than the rotation estimation with only about -2500%. This can be explained by the fact that landmark sets which are only spread over a small region yield even worse estimates than a narrow field of view, due to less pixel resolution in wide angle cameras.

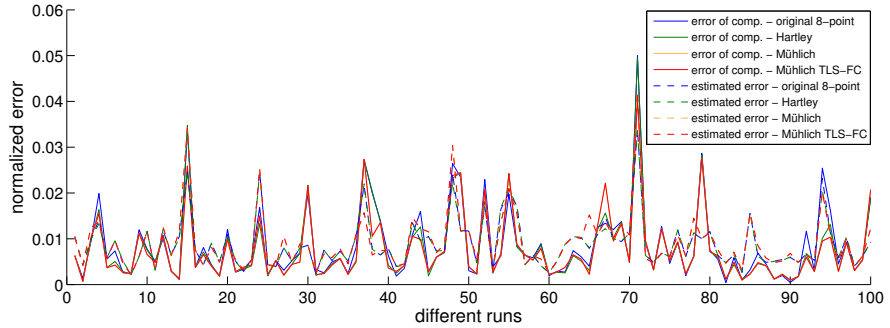
The influence of the number of features on the performance of the algorithm is illustrated in Fig. 8.32. It shows that even though the Essential matrix cannot be estimated more accurately by increasing the number of features, the prediction of the rotation matrix and the translation vector error becomes more reliable. This result can be explained by the normalization step in Eq. 3.16 where the singular values of $\boldsymbol{E}$ are fixed, which results in a non-predictive error.

Finally, Fig. 8.33 illustrates the performance of the error propagation for varying noise magnitudes. Again we can see that the error of the estimated Essential matrix does not go below a certain threshold, which is around 0.03, as we have seen in the previous plots. However, the rotation matrix and the translation vector error correlate in a linear way with the standard deviation of the noise and are predicted properly.

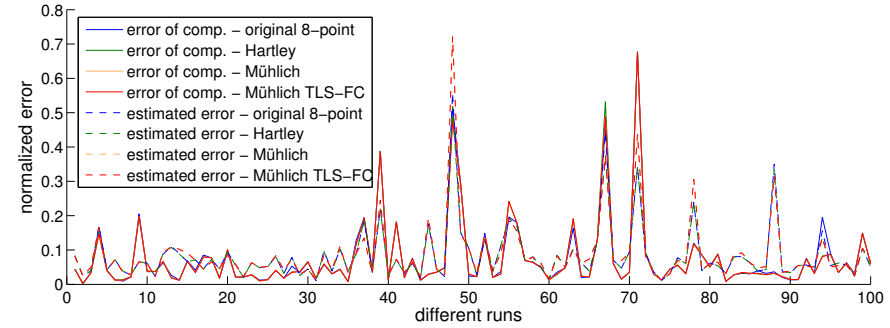
While the Essential matrix shows large discrepancies in the error prediction, the rotational and the translational error estimation works fairly enough. Since for a fusion, *e.g.*, in a Kalman filter like in Section 4.4, only the motion parameters and not the Essential matrix are used, this is no problem. Hence, the first order error propagation is accurate enough to compute the proper error magnitude, which allows the fusion with other sensors.



(a) Essential matrix ($\bar{\boldsymbol{E}}$)

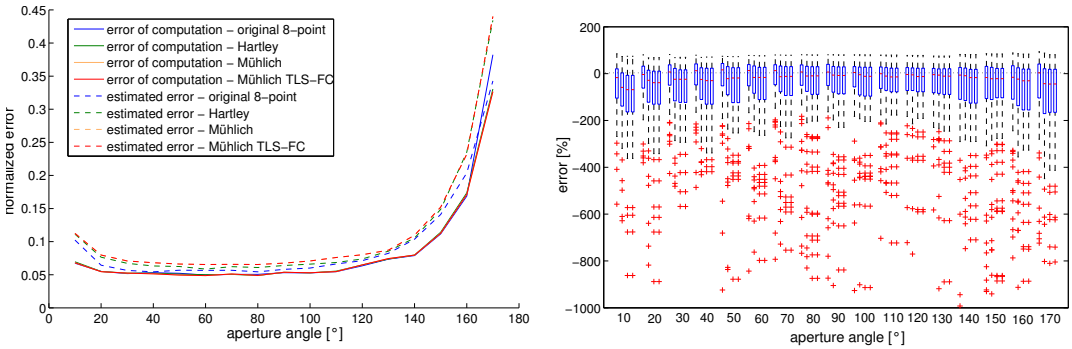


(b) rotation matrix ($\boldsymbol{R}$)

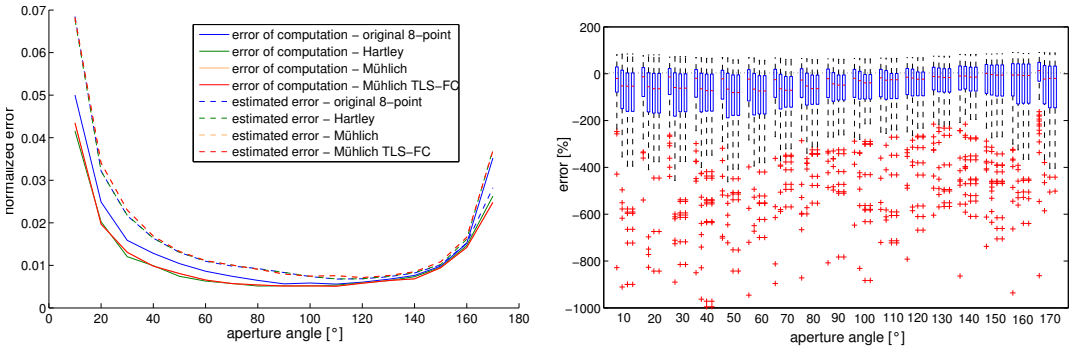


(c) translation vector ($\boldsymbol{t}$)

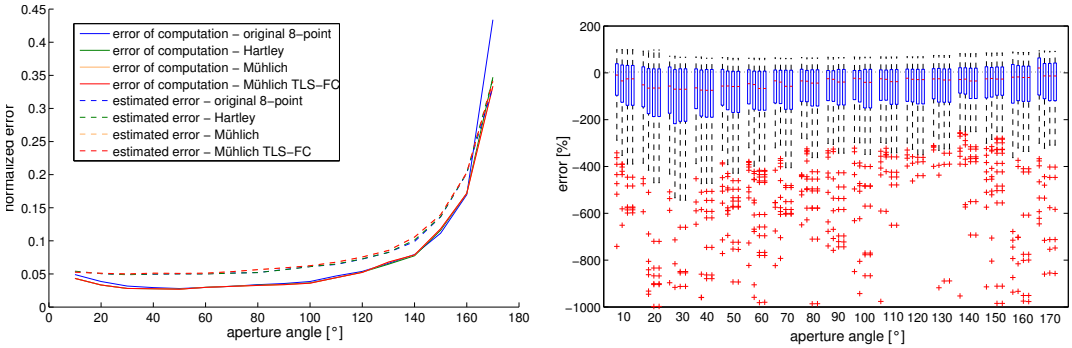
Figure 8.30: This graphic shows the error between the simulated and computed Essential matrix $\bar{\boldsymbol{E}}$, rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for 100 different runs (continuous lines). Moreover, the dashed lines denote the estimated error of the presented accuracy propagation. The simulated camera parameters are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$), 600×600 pixel resolution and 250 pixel focal length ($\sim 100°$ aperture angle.)



(a) Essential matrix ($\bar{\boldsymbol{E}}$)

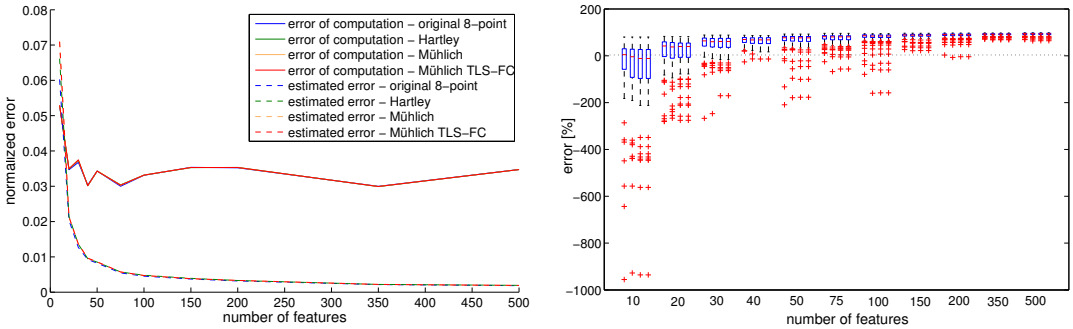


(b) rotation matrix ($\boldsymbol{R}$)

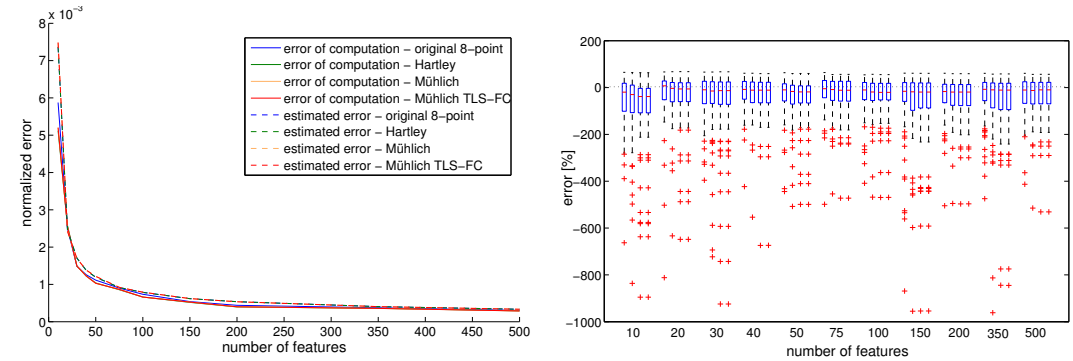


(c) translation vector ($\boldsymbol{t}$)

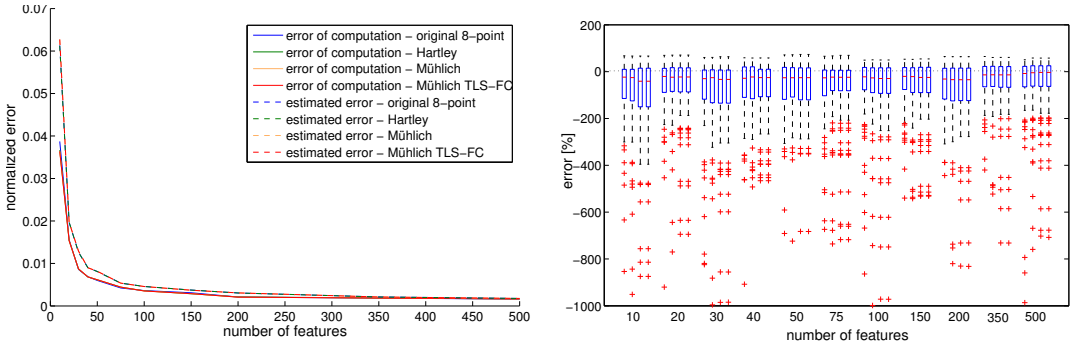
Figure 8.31: This graphic shows on the left hand side the median error of the computed and the simulated Essential matrix $\bar{\boldsymbol{E}}$, rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for different aperture angles and for all four introduced eight-point implementation variants. On the right hand side the relative errors of the uncertainty-propagation are illustrated as boxplots. The simulated camera parameters are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.



(a) Essential matrix ($\bar{\boldsymbol{E}}$)

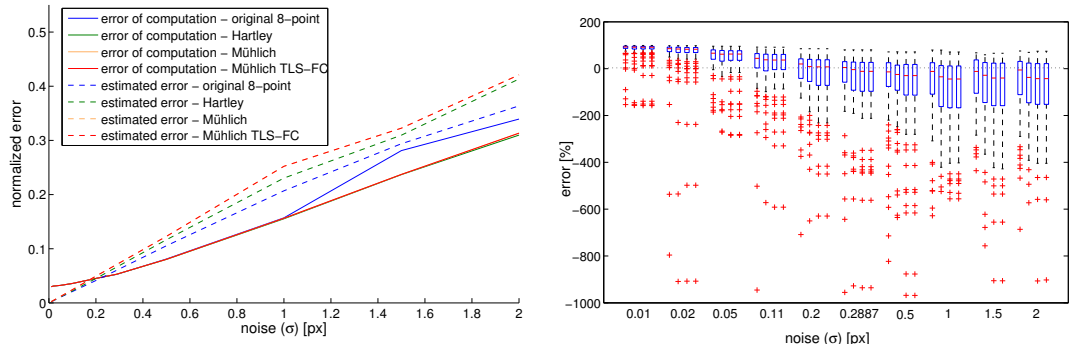


(b) rotation matrix ($\boldsymbol{R}$)

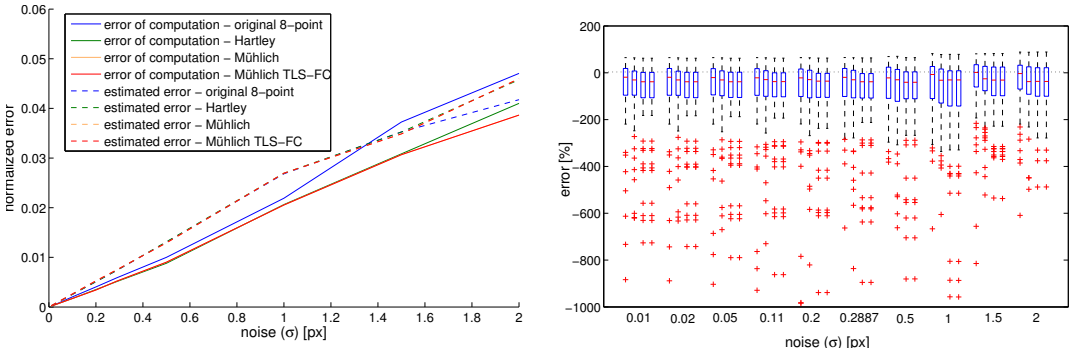


(c) translation vector ($\boldsymbol{t}$)

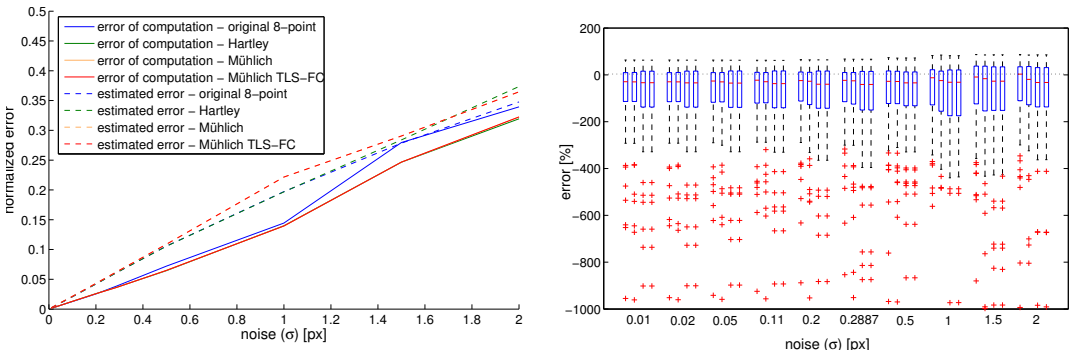
Figure 8.32: This graphic shows on the left hand side the median error of the computed and the simulated Essential matrix $\bar{\boldsymbol{E}}$, rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for a different number of features and for all four introduced eight-point implementation variants. On the right hand side the relative errors of the uncertainty-propagation are illustrated as boxplots. The simulated camera parameters are: noise in both images and 600×600 pixel resolution and 250 pixel focal length ($\sim 100°$ aperture angle).



(a) Essential matrix ($\bar{\boldsymbol{E}}$)



(b) rotation matrix ($\boldsymbol{R}$)



(c) translation vector ($\boldsymbol{t}$)

Figure 8.33: This graphic shows on the left hand side the median error of the computed and the simulated Essential matrix $\bar{\boldsymbol{E}}$, rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for different noise levels and for all four introduced eight-point implementation variants. On the right hand side the relative errors of the uncertainty-propagation are illustrated as boxplots. The same noise is simulated in both images. The value 0.2887 denotes noise by pixel-discretization as assumed for the former plots. The simulated camera parameters are: 10 features, 600×600 pixel resolution and 250 pixel focal length ($\sim 100°$ aperture angle).

8.4.2 Error Propagation for Z_∞-Algorithm

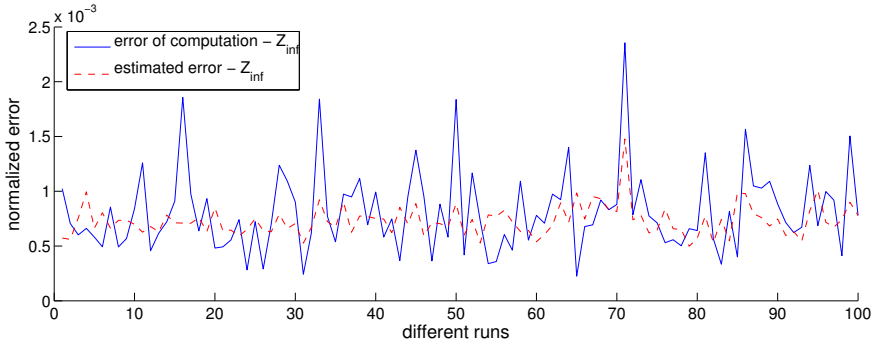
Like for the eight-point algorithm in the last section, we cannot use the real $\boldsymbol{R}$ and $\boldsymbol{t}$ for the Z_∞-error propagation, but only the error-corrupted estimates $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{t}}$ when propagating the error es described in Section 6.2. Analog to before, we can regard the real values as deviations of the same magnitude from the noise-corrupted estimates, which again justifies for the use of the estimation results for the error propagation. We evaluate the rotation and the translation estimation separately and together to show possible error correlations.

As before we compare the real and the estimated error as a scalar: we compute the errors of the calculated results $f_{\boldsymbol{R}}$, $f_{\boldsymbol{t}}$ and the estimated errors $\hat{f}_{\boldsymbol{R}}$, $\hat{f}_{\boldsymbol{t}}$ as in Eq. 8.4 and Eq. 8.5.

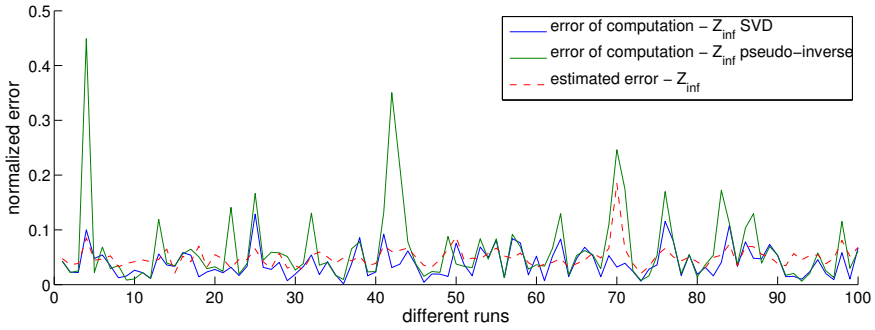
Fig. 8.34 illustrates the error between the computed and the simulated motion parameters (blue continuous lines) and the estimated error (red dashed lines). The plots proof the correlation of the propagated and the effective errors.

The first four plots in Fig. 8.35 to Fig. 8.37 show on the left hand side the median rotation and translation estimate and on the right hand side boxplots of the relative error according to the formula $\frac{f-\hat{f}}{f} \cdot 100$. Please note that not all outliers are shown in the boxplots for the sake of visibility. These four plots are generated by simulating only rotation or only translation. The last two plots are created by simulating a mixed motion and the dashed lines represent the prediction based on all features (green) or on the detected inliers only (red).

The first order error prediction of only rotation or only translation for a varying aperture angle (Fig. 8.35), a varying number of features (Fig. 8.36) or different levels of noise (Fig. 8.33) is quite accurate. However, the prediction differences for the rotation increase if the motion is mixed. This is an obvious consequence of small translational portions in the optical flow vectors of features which are identified to be translation-invariant. The effect can be lowered if the threshold of detecting far landmarks is decreased - a trade-off of increasing estimation accuracy and noise tolerance has to be found. Interestingly, even though the rotation is estimated worse than predicted, the translation estimation is not much affected by it. This let reason that the error in the estimated rotation is tolerable or correlates with the direction of translation. Latter assumption is highly probable, because the rotation error arises from the translational component and, apparently, the wrong rotation estimation does not significantly affect the estimation of the direction of translation.

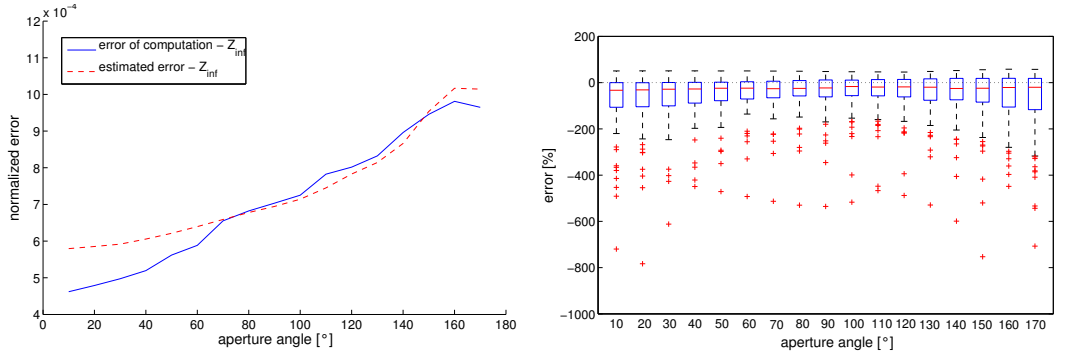


(a) rotation matrix ($\boldsymbol{R}$)

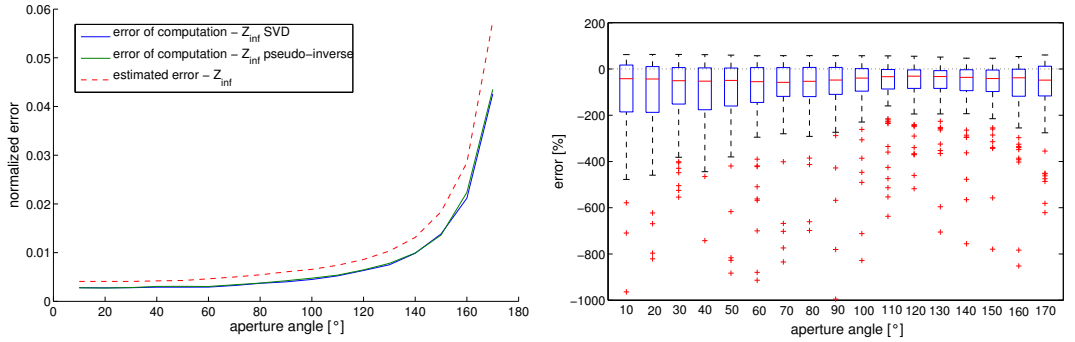


(b) translation vector ($\boldsymbol{t}$)

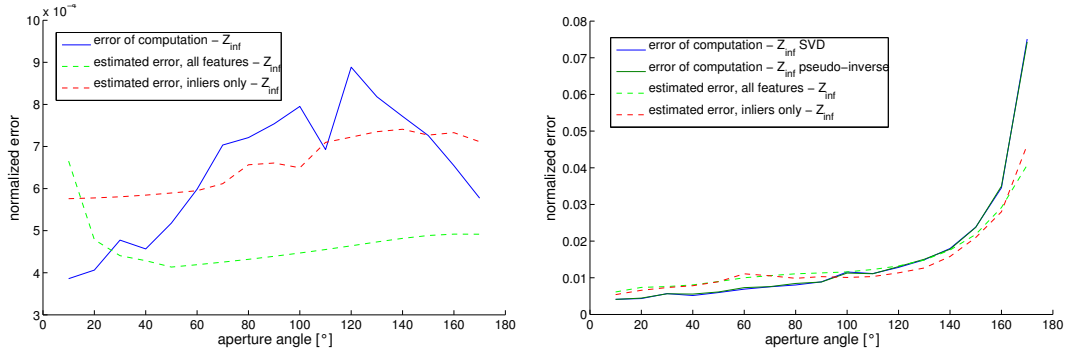
Figure 8.34: This graphic shows the error between simulated and computed rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for 100 different runs with only rotation or only translation (continuous lines). Furthermore, the dashed lines denote the estimated error of the presented accuracy propagation. The simulated camera parameters are: 10 features, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$), 600×600 pixel resolution and 250 pixel focal length ($\sim 100°$ aperture angle.)



(a) rotation matrix ($\boldsymbol{R}$)

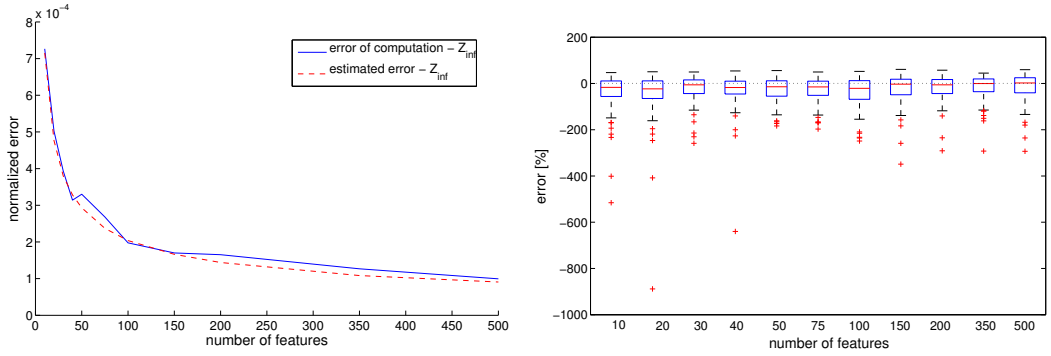


(b) translation vector ($\boldsymbol{t}$)

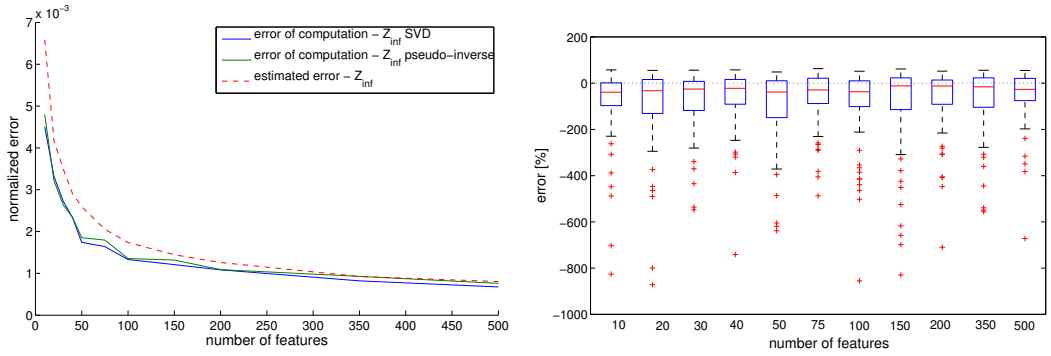


(c) mixed motion - rotation (left), translation (right)

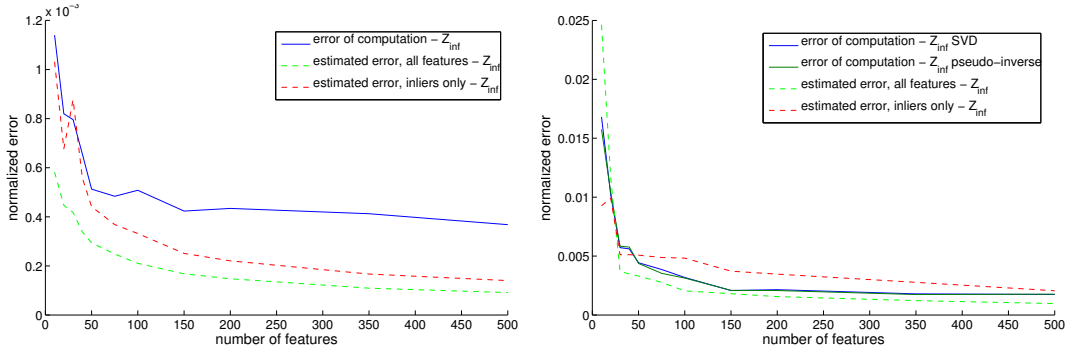
Figure 8.35: This graphic shows on the upper two left plots the median error of the computed and the simulated rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for different aperture angles. On the right hand side the corresponding relative errors of the error-propagation are illustrated as boxplots. The lower two images show the performance if both motions, rotation and translation, are simulated and estimated at the same time. The simulated camera parameters are: 10 or 20 features depending on whether only one motion component (rotation or translation) or both components are simulated, noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$) and 600×600 pixel resolution.



(a) rotation matrix ($\boldsymbol{R}$)

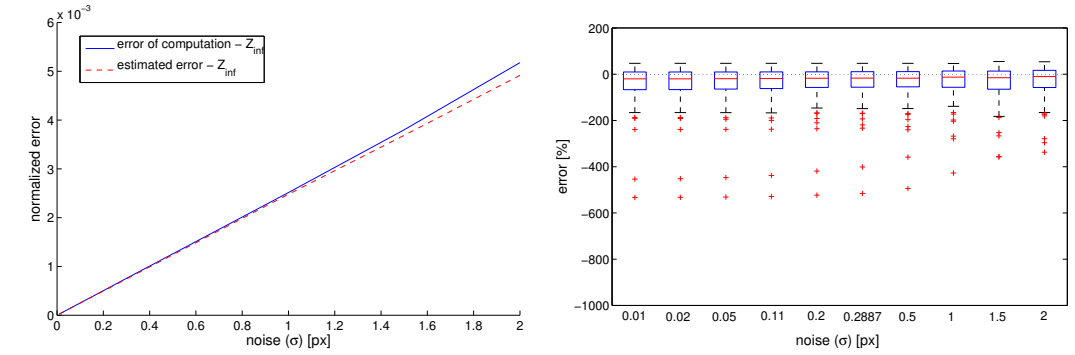


(b) translation vector ($\boldsymbol{t}$)

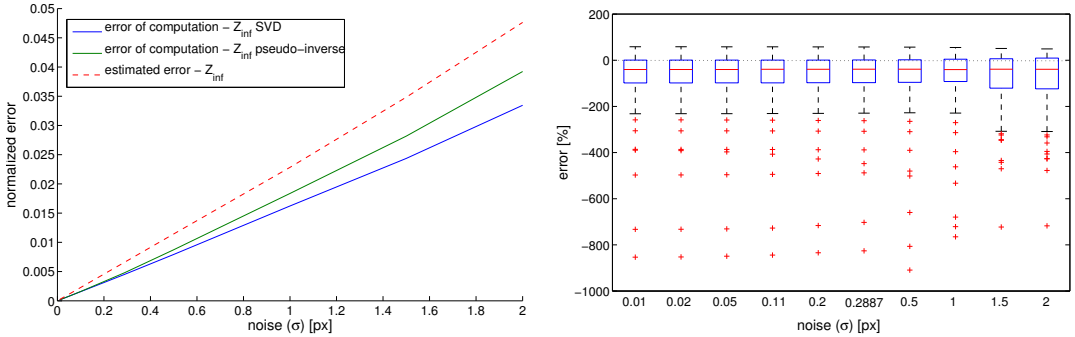


(c) mixed motion - rotation (left), translation (right)

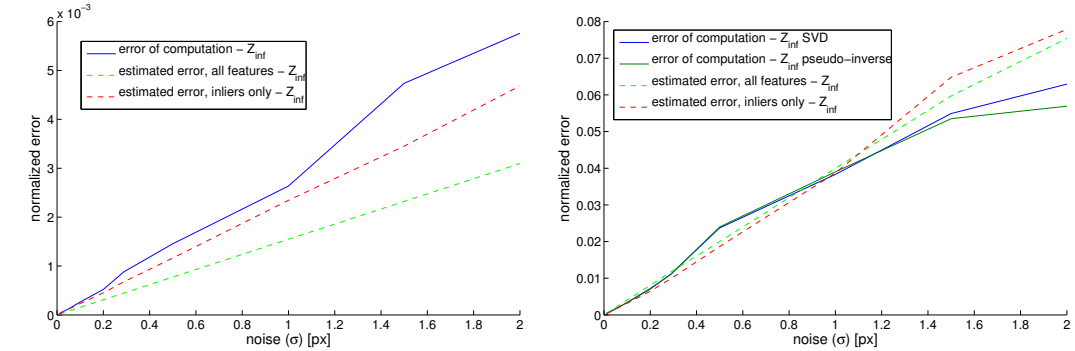
Figure 8.36: This graphic shows on the upper two left plots the median error of the computed and the simulated rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for a varying number of features. On the right hand side the corresponding relative errors of the error-propagation are illustrated as boxplots. The lower two images show the performance if both motions, rotation and translation, are simulated and estimated at the same time. The simulated camera parameters are: noise in both images due to pixel-discretization of the features ($\sigma^2 = \frac{1}{12}$), 600×600 pixel resolution and 250 px focal length ($\approx 100°$ aperture angle).



(a) rotation matrix ($\boldsymbol{R}$)



(b) translation vector ($\boldsymbol{t}$)



(c) mixed motion - rotation (left), translation (right)

Figure 8.37: This graphic shows on the upper two left plots the median error of the computed and the simulated rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ for various levels of noise. On the right hand side the corresponding relative errors of the error-propagation are illustrated as boxplots. The lower two images show the performance if both motions, rotation and translation, are simulated and estimated at the same time. The simulated camera parameters are: 10 or 20 features depending on whether only one motion component (rotation or translation) or both components are simulated, 600×600 pixel resolution and 250 px focal length ($\approx 100°$ aperture angle).

8.5 Evaluation of Sensor Registration and Fusion Techniques

We presented different techniques for temporal and spatial alignment of an IMU-camera setup as well as concepts for the fusion of the measurements in Chapter 4. First, we evaluate the temporal and spatial sensor registration on synthetic and real data and then we show experiments where the motion measurements of the sensors are combined. For the temporal and spatial alignment we use the same real data, acquired by the IMU-camera pair illustrated in Fig. 8.38.

Figure 8.38: A PointGrey Flea2G camera with wide angle lens and a Xsens-MTi IMU (orange box) as it is used in our experiments.

The PointGrey Flea2G camera[1] is equipped with a wide angle lens, providing an approximately 120° aperture angle. It is mounted on a Xsens-MTi IMU. Eight runs have been recorded, where the camera is moved in front of a checkerboard for 10-40 s each, gathering images and inertial data. The frame-rate is 15 Hz while the IMU has a sampling frequency of 120 Hz. The camera rotation is computed by extracting the corners of a checkerboard and estimating the camera position in an optimization framework using Calde and Callab [109]. We neglect the correlation of the image-based rotation estimation with the translation in the experiments where only the rotation is required. However, this correlation is considered in the batch-optimization where a full spatial registration is provided. Alternatively, one can also use translation invariant (far distant) landmarks to estimate the rotation uncoupled from the translation, like in the Z_∞-algorithm explained in Section 5.1. In the case at hand, the scale factor α for the translation measurements of the monocular camera is known due to the available geometry information of the checkerboard. Nevertheless, we will estimate it in our batch-optimization, because the resulting value of α has turned out to be a good indicator for the success of an optimization run: A value close to 1.0 usually

[1] http://www.ptgrey.com

means that a reasonable data alignment could be achieved, while large deviations of the optimal value indicate a failure of the algorithm. Errors in the intrinsic calibration, the accelerometer calibration or the checkerboard dimensions typically also cause slight deviations of α from the ideal value.

8.5.1 IMU-Camera Temporal Alignment

Before the measurements can be aligned, the time-stamps have to be fixed as described in Section 4.1. Therefore, the sampling period has to be estimated and sample jams and gaps have to be detected as illustrated in Fig. 8.39.

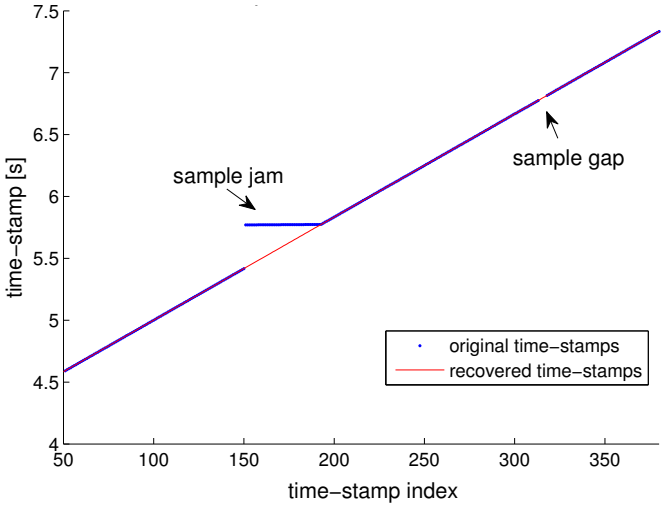


Figure 8.39: The blue dots represent the time-stamps of a clipped image sequence. The red line shows the fixed-time line without data jams and gaps.

In the following, we want to show the advantages and drawbacks of the temporal methods discussed in Section 4.1. We tested the delay estimation methods with simulated data based on a weighted overlap of nine sine frequencies on all axes, whereas the frequencies are in between 0.1 Hz and 90 Hz. The sampling period of the simulated gyroscopes is 10 ms and the one of the camera 40 ms. As window function for the EDFT in the phase congruency method we chose a Hamming window, which is a good trade-off between high dynamic-range and sensitivity. For the maximum weight phase shift estimation we used the rectangular window, because it provides the highest sensitivity.

Fig. 8.40 illustrates a simulated delay between IMU and camera of 0.5 s and the corresponding cross correlation result. The delay is rather long compared to real world cases, but it has been chosen to show the limits of the correlation method. Fig. 8.41

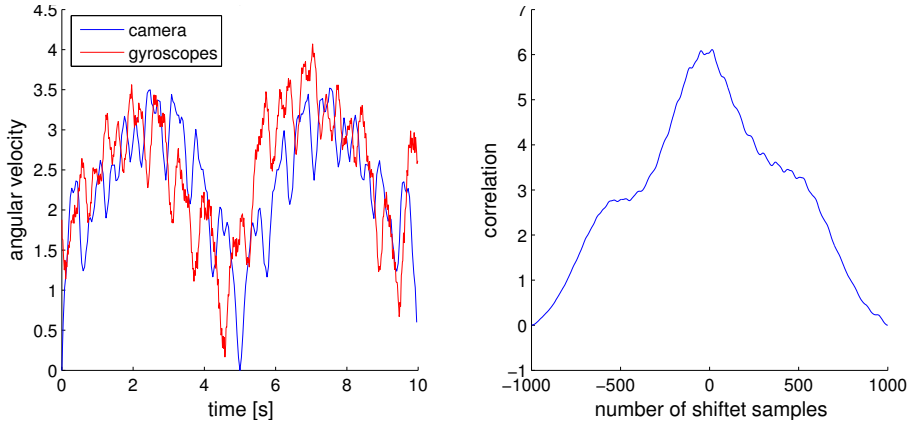


Figure 8.40: The left figure illustrates the unaligned simulated gyroscope (red) and camera data (blue), while the right figure shows the cross correlation of the data.

depicts the estimated magnitudes and the estimated phases for both sensors. The phases of corresponding peaks are used for temporal alignment.

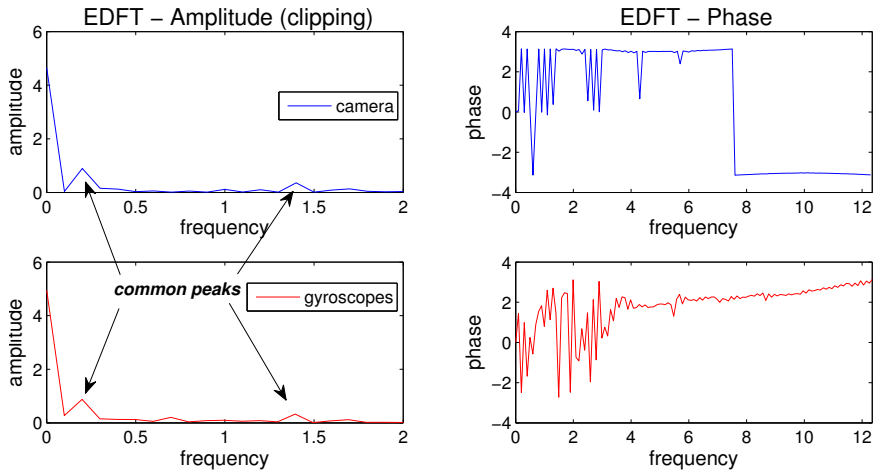


Figure 8.41: This figures illustrate the phase shift based temporal alignment for the simulated data. The left images show the amplitude and the right the phase spectrum of the gyroscope (red) and camera (blue) data.

The result of the average phase shift based approach is 0.5193 s, while the phase shift of the most significant frequency yields 0.5344 s. The correlation-based approach estimates a 0.14 s time lag. The explanation for this can be found in Fig. 8.42. There we evaluated the performance on various delays. While the cross-correlation works highly accurate and reliable for small delays, it finds a wrong optimum if the delay is large relative to the sample length. The average and maximum phase shift methods seem to

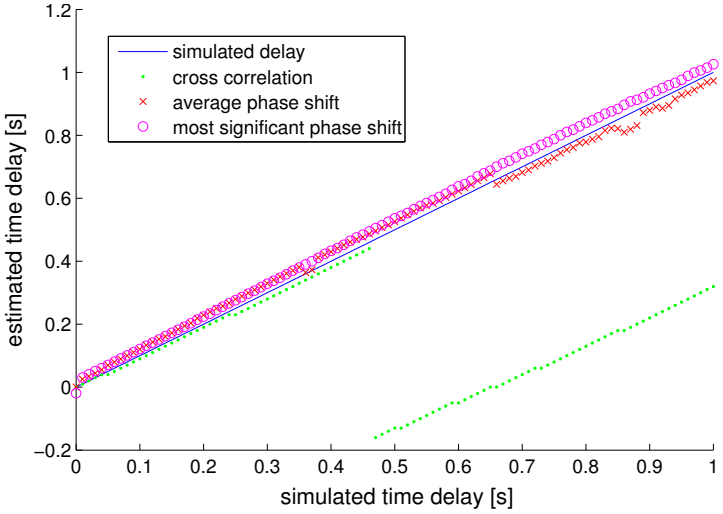


Figure 8.42: The blue line represents the simulated delay. The green dots are the correlation based results. The red crosses and the magenta circles are the average and the most significant phase shift output respectively.

be less accurate but more robust as the delay increases. The difference between these two methods becomes apparent when comparing the result with the real data, as done in Fig. 8.43. Table 8.4 summarizes some statistical quantities of the outcome. While

Table 8.4: This table lists the statistical quantities, mean, median, standard deviation, and range (maximum minus minimum), for the results of the presented methods.

method - unit [s]	*mean*	*median*	*std. dev.*	*range*
cross correlation	0.1195	0.1179	0.0047	0.0115
average phase shift	0.1831	0.1246	0.1841	0.5659
most significant phase shift	0.4870	0.2348	0.8704	2.7979

the cross correlation method achieves coherent results in all eight runs, the phase based approaches proof to be not as reliable for real application. This is because white noise affects all frequencies and, hence, the phase shift can not be used for accurate temporal alignment of noisy data.

8.5.2 IMU-Camera Spatial Registration

Before we assess the batch-optimization based registration introduced in Section 4.2 we evaluate the rotational alignment as described in Section 4.2.4, which is used as rough registration or as initialization for the optimization.

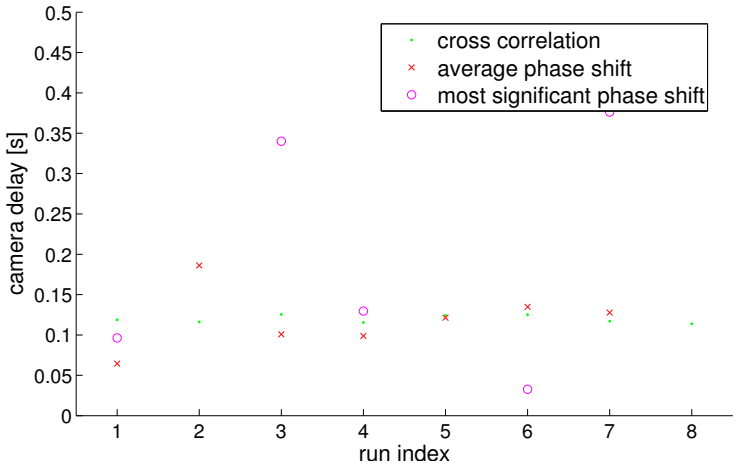


Figure 8.43: The cross-correlation reliably estimates the time delay, while the phase congruency methods are not as robust. Some of their estimates are not even visible because of the scaling which has been chosen for the sake of detail.

Initialization of Rotational Alignment

We added white Gaussian noise to the synthetic data used for the evaluation of the temporal alignment to evaluate the noise sensitivity of the rotation initialization. Fig. 8.44 shows the performance of the weighted and the unweighted rotation estimation. The weighted rotation estimation always outperforms the unweighted method.

It is difficult to determine an accurate ground truth for the rotational alignment between a camera and an IMU without having a CAD drawing of the setup. Therefore, we performed a brute force search with a resolution of 0.5° to find a sort of "ground truth". As objective function we used Eq. 4.41. Unfortunately, in one of the runs a local minimum far off any possible solution has been found which proofs the non-convexity of the problem and, hence, that filtering and optimization techniques may easily find a local instead of the global minimum if the starting point is far off the actual solution. This run has not been considered in Fig. 8.45, which shows the deviation of the various methods described in Section 4.2.4 relative to the brute-force estimate. The estimates of the unweighted (original) and the weighted rotation are optimized according to Eq. 4.42 and not after the nonlinear Eq. 4.54 as it should be. Thus, we propagate the results by the unscented transformation described in Section 4.2.4, with the outcome, that the difference between the propagated and original values is negligible and, thus, the unscented transformation can be spared. Such a transformation becomes only necessary if the covariance matrix is large, which is not the case for our data. The weighted approach proves to be more reliable and accurate compared to the original

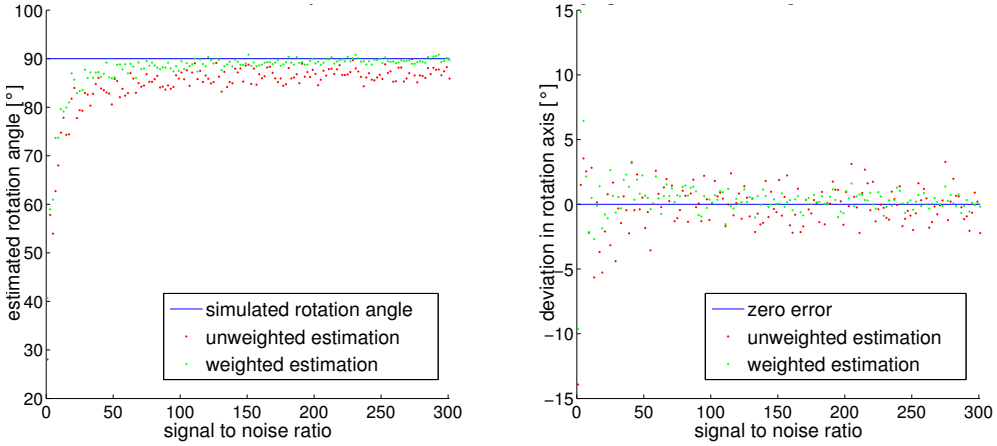


Figure 8.44: The left image shows the results for the absolute rotation angle, the right picture illustrates the angular error between the estimated and simulated rotation axis. The blue line represents the simulated absolute angle and the zero-error line respectively. By increasing the signal to noise ratio the performance of both, the weighted (green) and the unweighted (red) rotation estimation, increases.

hand-eye calibration technique. Moreover, the bias estimates are more coherent with the weighting method than the covariance minimization approach.

In the next experiments, we want to underline the importance of a proper temporal alignment. It is difficult to argue based on simulations which temporal accuracy is relevant for spatial alignment, because it depends strongly on the dynamics of the registration run, the noise of the sensors, the accuracy of the camera based pose estimation, and so on. Therefore, we run the rotation estimation on misaligned real data. The time delays have been chosen between the mean estimate of all runs and a delay of zero, which means that no temporal alignment is provided. Fig. 8.46 compares the different runs. While the median does only change a little for the different time delays, the variance increases significantly for bad aligned data. Furthermore, the plot shows that the weighting of the data samples also increases the robustness against temporal misalignment.

The batch-based nonlinear optimization shows also to be sensitive to proper temporal alignment. In this experiment, we varied the estimated time delay from 0 % to 200 % and plotted the box plots of the estimates for the translational scale to evaluate the effect of a time lag between the measurements. The result worsens significantly if the measurements are not aligned correctly (100 %), as shown in Fig. 8.47. This experiment stresses the importance of proper temporal alignment also for complex registration methods. Adding the delay between the IMU and camera measurement as

Figure 8.45: These box plots show the performance of the presented rotational alignment methods. Therefore, the estimated Euler angles between IMU and camera have been substracted from a "ground truth" estimate gathered by a brute force search with resolution 0.5°. The leftmost box plot of each set corresponds to the four short runs, the rightmost to three of the long runs and the center box plot represents all seven runs.

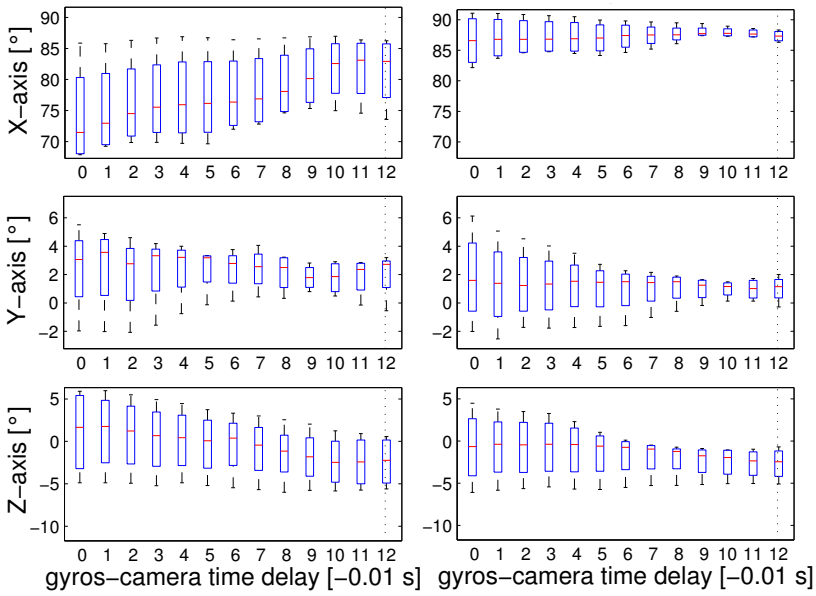


Figure 8.46: These boxplots show the performance of the unweighted and weighted rotation estimation on not correctly temporally aligned real data. The black dotted line corresponds to the mean of the estimated delays between camera and gyros of all runs.

parameter to $\boldsymbol{\theta}$ would worsen the convergence properties of the optimization and make the analytical calculation of the Jacobian significantly more complex.

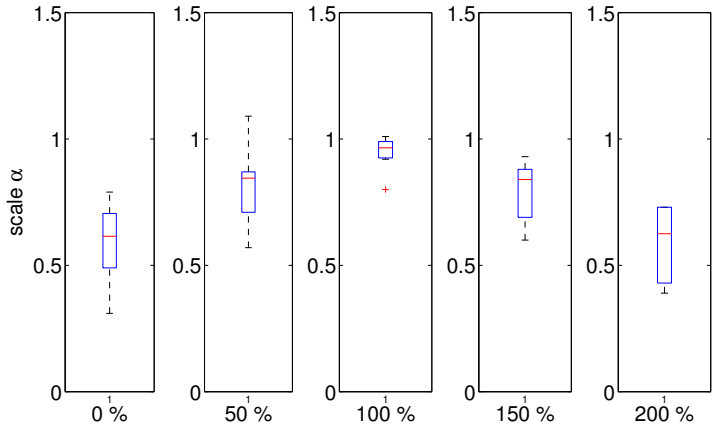


Figure 8.47: Estimated scale factor α with respect to the temporal alignment: 0 %, no temporal alignment; 100 %, correct temporal alignment; 200 %, twice the estimated delay between camera and IMU.

Evaluation of the Batch-Optimization

Let us now evaluate the spatial registration as introduced in Section 4.2. In the following plots, the X-axis components are colored blue and the Y- and Z-components are in green and red, respectively. The IMU measurements are marked with dots and the ones of the camera with crosses.

First, we will show the results of our approach on simulated data to evaluate its performance based on ground truth. In our simulation we assume, that the camera provides images with a frame-rate of 15 Hz, while the IMU measurements arrive with a rate of 120 Hz. The length of the simulated registration sequence is about 10 s, which is similar to our real registration sequences. The simulated rotational and translational motion corresponds to three superimposed sine waves with varying frequencies and amplitudes for each degree of freedom. The signals are corrupted by noise with the following standard deviations, which have been chosen to be similar to the ones measured in the real data: $\sigma_{\boldsymbol{\omega}^I} = 28.6\,^\circ/\mathrm{s}$, $\sigma_{\boldsymbol{a}^I} = 0.5\,\mathrm{m/s^2}$, $\sigma_{\boldsymbol{\omega}^C} = 4.3\,^\circ/\mathrm{s}$ and $\sigma_{\boldsymbol{v}^C} = 0.05\,\mathrm{m/s}$. The covariance matrix Σ_P of the estimated parameters is calculated according to [108] by

$$\boldsymbol{\Sigma}_P = \left(\boldsymbol{J}^T \boldsymbol{\Sigma}_X^{-1} \boldsymbol{J}\right)^+ \tag{8.6}$$

with $\boldsymbol{\Sigma}_X = \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\omega}^I}\ \boldsymbol{\sigma}^2_{\boldsymbol{a}^I}\ \boldsymbol{\sigma}^2_{\boldsymbol{\omega}^C}\ \boldsymbol{\sigma}^2_{\boldsymbol{v}^C})$. A comparison of the estimated values and their standard deviation with the simulated values shows that the results are consistent. Fig. 8.48 and Table 8.5 illustrate the result of the presented optimization approach – ${}^I\boldsymbol{\phi}_C$ denotes the Euler angles corresponding to ${}^I\boldsymbol{q}_C$.

Table 8.5: This table lists the simulation results for the distance between IMU and camera, ${}^I r_{IC}$, and the corresponding standard deviations σ_r.

run	${}^I\boldsymbol{t}_{IC}$ $[mm]$			${}^I\boldsymbol{\phi}_C$ $[^\circ]$		
	x	y	z	x	y	z
simulated	10	0	-5.0	90	0	0
estimated	10.4	3.0	-5.0	89.7	0.8	-0.8
std. dev. σ	5.3	5.5	5.6	0.4	0.4	0.4

The optimization parameters for the experiments on the real data are initialized as described in Section 4.2.4. The results of the translation estimation and its computed uncertainty for all eight runs are illustrated in Table 8.6. It shows the estimated values for ${}^I\boldsymbol{t}_{IC}$ and the corresponding standard deviations $\boldsymbol{\sigma}_{{}^I\boldsymbol{t}_{IC}}$.

Table 8.6: This table lists the estimated distances between IMU and camera, ${}^I r_{IC}$, and the corresponding standard deviations σ_r for the real data sets.

run	${}^I\boldsymbol{t}_{IC}$ $[mm]$			$\boldsymbol{\sigma}_{{}^I\boldsymbol{t}_{IC}}$ $[mm]$		
	x	y	z	x	y	z
1	-20.6	69.6	-30.7	37.9	49.6	60.1
2	-3.0	62.9	-31.3	25.9	33.7	38.2
3	-18.7	64.6	-39.4	28.2	32.2	36.9
4	-18.9	68.5	-33.2	18.4	24.5	22.4
5	-21.3	52.5	-33.4	6.5	9.0	8.2
6	-17.9	58.1	-33.5	4.7	12.3	4.3
7	-25.8	60.4	-30.9	6.7	8.5	6.7
8	-20.5	46.9	-25.1	13.9	41.4	16.7

We will now compare our results with the system identification (grey box) approach described in [188] and a filter approach using an UKF as presented in [183]. The only modification for the grey box approach is that we do not use an EKF but a more accurate UKF instead to propagate and update the system state. The parameter vector $\boldsymbol{\theta}_{\text{UKF}}$ of the optimization consists of:

$$\boldsymbol{\theta}_{\text{UKF}} = \left(\boldsymbol{b}^T_{\boldsymbol{\omega}^I}\ \boldsymbol{b}^T_{\boldsymbol{a}^I}\ {}^{I_0}\boldsymbol{g}^T\ {}^I\boldsymbol{q}^T_C\ {}^I\boldsymbol{t}^T_{IC}\right)^T \tag{8.7}$$

The boxplots in Fig. 8.49 illustrate the calibration results of the three different approaches. The estimation of the relative pose ${}^I\boldsymbol{t}_{IC}$ by the B-spline clearly outperforms

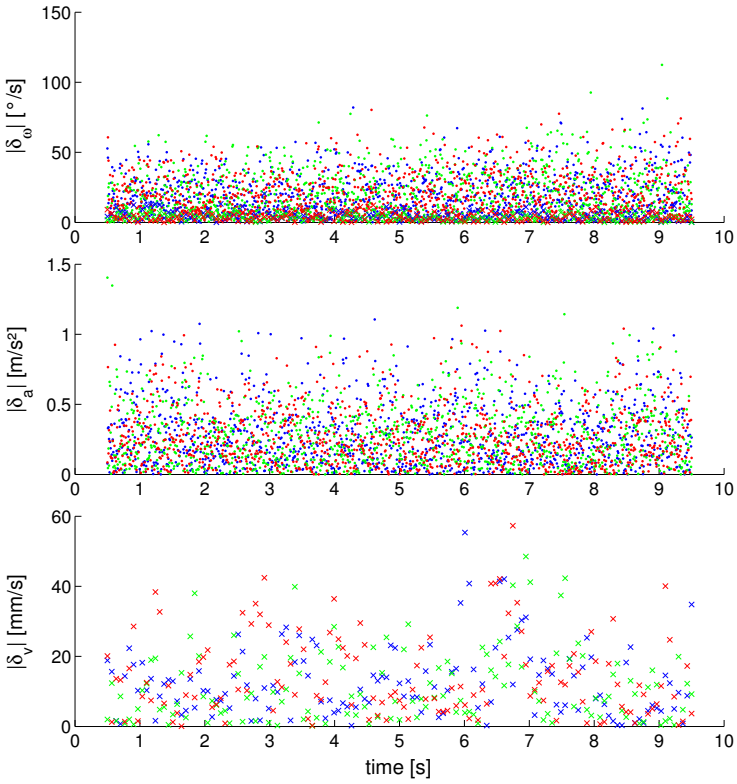


Figure 8.48: Error between the simulated measurements and the estimated trajectories. Please note that the crosses represent the camera measurements which are closer to zero due to less noise assumed for the camera. The upper plot illustrates both angular velocities $\boldsymbol{\omega}^I$ and $\boldsymbol{\omega}^C$, the middle plot shows $\boldsymbol{a}^I$ and the lower one $\boldsymbol{v}^C$. The errors of the estimation correspond to the simulated noises.

the filter-based methods, while the angular alignment ${}^{I}\boldsymbol{q}_{C}$ yields comparable results. A closer look at the results shows that especially the runs with less dynamics account for this difference. The B-spline approach seems to be more sensitive and, thus, it achieves adequate results even though the dynamics of the calibration motion are low.

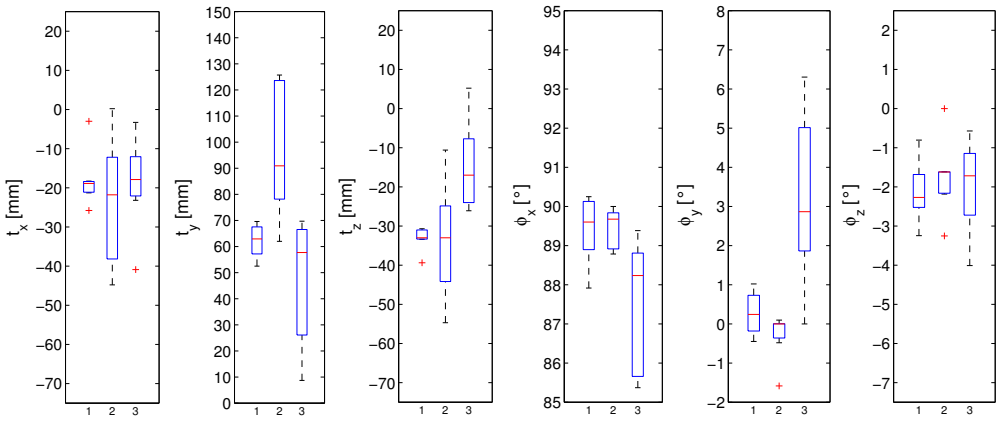
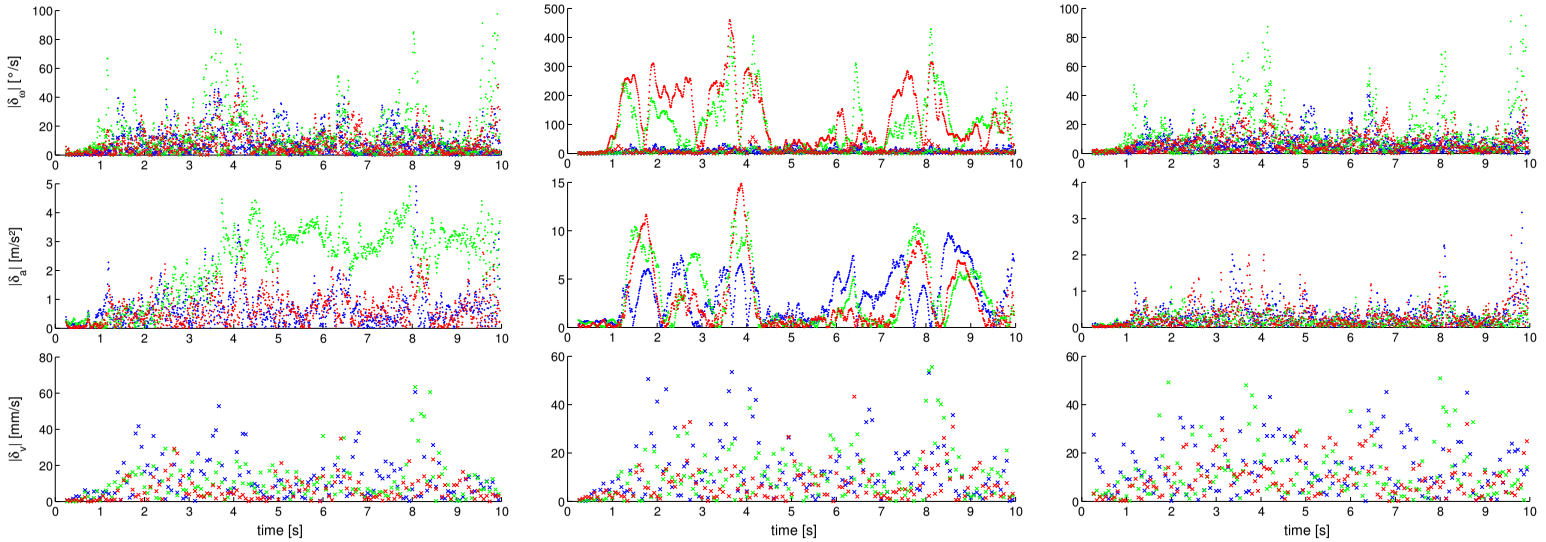


Figure 8.49: Estimation of ${}^{I}\mathbf{t}_{IC}$ and the Euler angles ${}^{I}\boldsymbol{\phi}_{C}$ corresponding to ${}^{I}\boldsymbol{q}_{C}$ (boxplots). *1-left*: B-spline, *2-middle*: grey box, *3-right*: UKF.

To evaluate the sensitivity of the presented algorithm regarding the quality of the initialization, we used a bad initialization to run the optimization. For that, we assumed no knowledge about the relative and the absolute orientation, setting ${}^{I}\boldsymbol{q}_{C}$ to zero and ${}^{I_0}\boldsymbol{g}$ to $(0\ 0\ -9.81)$. The errors between the trajectories and the measurements resulting from the two different initializations are illustrated in Fig. 8.50(a) and Fig. 8.50(b). The wrong initialization of the rotation between camera and IMU becomes apparent in the angular velocity plots. The bad initialization of the gravity vector becomes obvious in the much larger scale of the acceleration plot in Fig. 8.50(b) - the gravitation component in the acceleration measurements can not be compensated properly. Nevertheless, in our experiments the optimization converges always to the same result, shown in Fig. 8.50(c), which indicates good convergence properties of this problem. The same experiment is performed using the grey box and UKF approach. Both can handle the bad initial orientation, but run into serious problems with the bad initial gravity ${}^{I_0}\boldsymbol{g}$ and cannot estimate properly the relative pose anymore.

The last two experiments shall illustrate the sensitivity of the optimization with respect to the number of control points and the quality of the temporal alignment. The relative pose ${}^{I}\boldsymbol{t}_{IC}$ and the scale factor α react most sensitive to changes of these parameters and, thus, they are used to evaluate the quality of the estimation. Fig. 8.51 shows the convergence of the optimization results with increasing number of knots.



(a) good initialization (b) bad initialization (c) estimation result

Figure 8.50: The first two plot-columns show the errors of $\boldsymbol{\omega}^I$, $\boldsymbol{\omega}^C$ (both in the upper row), $\boldsymbol{a}^I$ (middle row) and $\boldsymbol{v}^C$ (lower row) relative to a good and a bad B-spline initialization as described in the text. Note the difference in the angular velocity plots and the difference in the scale of the acceleration plots. The rightmost plot denotes the estimation result, which is the same for both initializations.

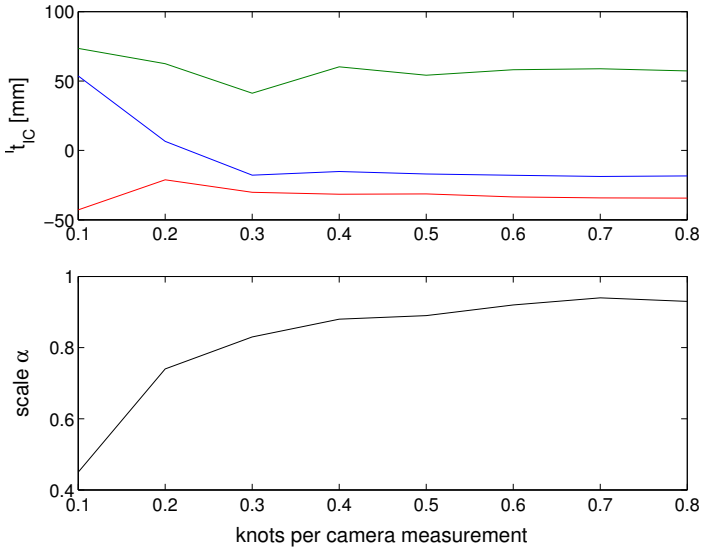


Figure 8.51: Estimated relative pose ${}^{I}\boldsymbol{t}_{IC}$ (above) and scale α (below) with respect to the number of knots.

According to this experiment, there are no significant improvements to expect using more than 0.6 knots per camera frame for our registration sequences. If the spline has too few knots, it will not allow accurate modeling of the acceleration measurements and, thus, the velocity measured by the camera does not fit the approximated acceleration. A large number of knots increases the number of control points and, consequently, the overall optimization duration.

The accuracy of the spatial registration depends strongly on the measured motion. No calibration framework will be able to improve the result beyond the calibration errors which affect the fusion. Thus, assuming that the observability conditions mentioned in [186] are met, the accuracy of the spatial alignment strongly depends on the signal to noise ratio. In general, the noise of the sensors cannot be reduced further, thus, one should aim to provide a calibration run with high dynamic motions. If at least the same dynamics as a specific application requires are provided, the residual calibration error can be neglected because the resulting fusion error is not significant compared to the measured motion.

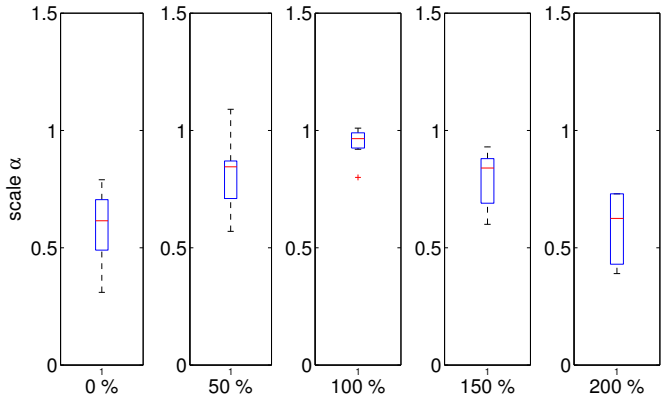


Figure 8.52: Estimated scale factor α with respect to the temporal alignment: 0 %, no temporal alignment; 100 %, correct temporal alignment; 200 %, twice the estimated delay between camera and IMU.

8.5.3 IMU Aided KLT Tracking

Significantly faster rotations are feasible if any local tracking is supported by an IMU as explained in Section 4.3. Figure 8.53 depicts a fast sweep of the camera with an average rotation of 11° and a maximum of 14° per frame. A maximum feature displacement of about 45 pixels has been measured. Nevertheless, the landmark location can be estimated accurately: the yellow squares (feature propagations) match the red circles (KLT tracking results). The exact feature propagation makes also a change of the active feature set during such motions feasible. Good matches of the red circles and the yellow circles (projections of the landmarks according to their 3D location and the camera pose) confirm an accurate stereo-based initialization (see Section 7.1.1).

8.5.4 EKF Based IMU-Camera Fusion

In the following, we want to show some first results on synthetic data for the EKF based fusion of a camera and an IMU, as described in Section 4.4. We simulate an oscillating motion with translational velocities up to 2 m/s and angular velocities up to 114 °/s as shown in Fig. 8.54. The parameters for the sensors were chosen according to our experiences with real sensors. The IMU samples are acquired at 500 Hz and the camera images at 120 Hz. We simulate a measurement sequence of 60 seconds and, thus, in total 3000 IMU measurements and 620 image-based motion estimates, to test the EKF concept. The biases on the accelerometers and gyroscopes drift by a random process with standard deviation of 10^{-5} and 10^{-4} respectively. Each axis of the gyros suffers from noise of the magnitude $\sigma = 10^{-3}$ and the ones of the accelerometers of

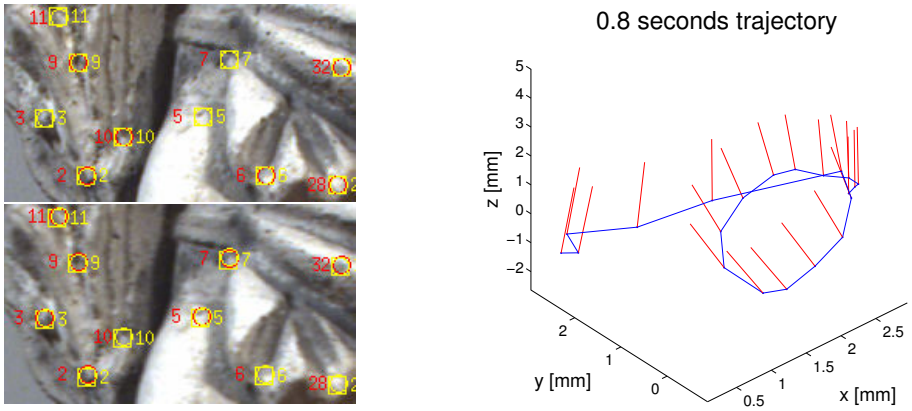


Figure 8.53: The left two images show two screenshots acquired at the beginning and at the end of a fast camera motion (up to 14° per frame). The yellow squares mark the IMU-supported propagation of the landmarks, while the red circles are the tracked features. The yellow circles are the reprojection of the features according to the 3D structure. The right picture illustrates the trajectory (blue) and the camera viewing directions (red).

$\sigma = 0.1$. The noise on the image based motion estimation is simulated as $\sigma = 0.01\,\mathrm{m}$ and $\sigma = 0.1\,^\circ$ for each component of the translation and rotation, respectively. For every new keyframe we simulate a random switching-error with $\sigma = 10\%$ of the estimated scale.

In our first experiment, we assume no translation between IMU and camera and an exact initialization of the camera scale. Furthermore, we simulate that every 10 s a new keyframe is used. Fig. 8.55 illustrates the result. The blue line represents the error of the estimates and the red lines denote the 3σ-bounds of the calculated uncertainty. The continuous growth of the uncertainty is clearly notable, which is due to the relative measurements between the keyframes. However, the estimation error remains small. Especially for the attitude, the uncertainty jumps when switching between two keyframes become apparent. Nonetheless, the filter is able to correctly level off the new scale after each switch of keyframe. The real trajectories, with ground truth, filter estimate and 3σ-bounds are shown in Fig. 8.56. We also draw the time instances of camera measurements and the keyframe switches in these plots, showing that the motion can be smoothly estimated despite these events.

The question how stable the feature tracking has to be, is evaluated in Fig. 8.57. We run the simulation for different times between a keyframe switching, with the expected result, that if the features get lost too quickly, the filter does not have enough time to converge. Hence, a large number of features and a high frame-rate should be preferred, which allow for a more robust tracking.

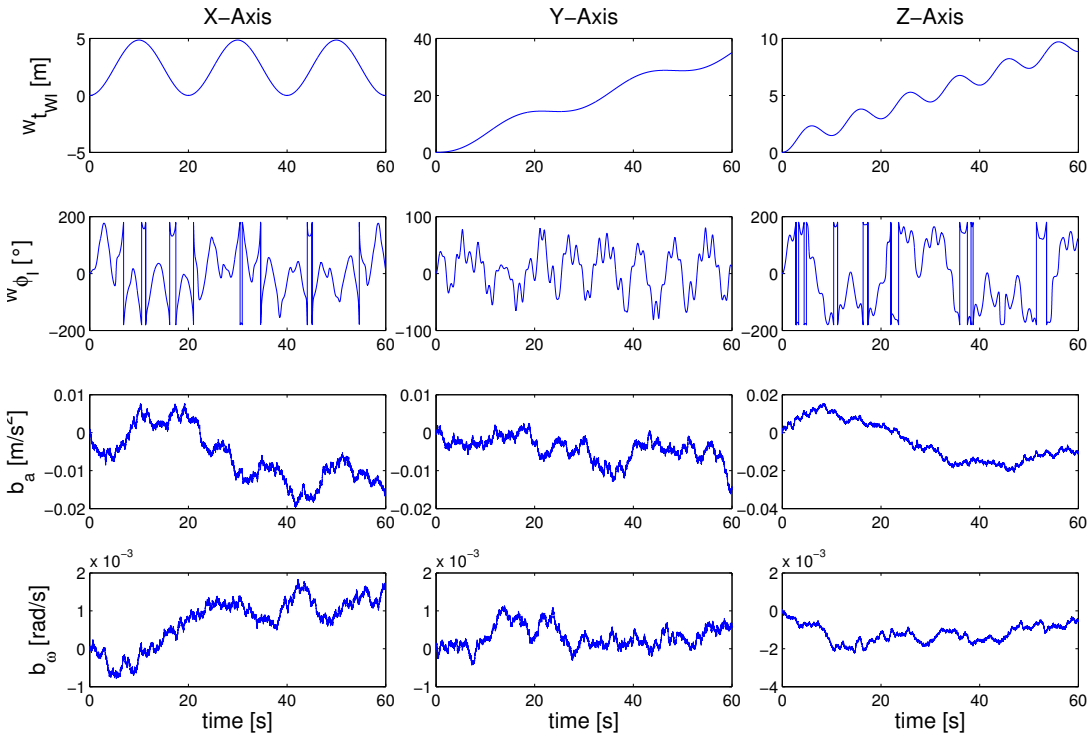
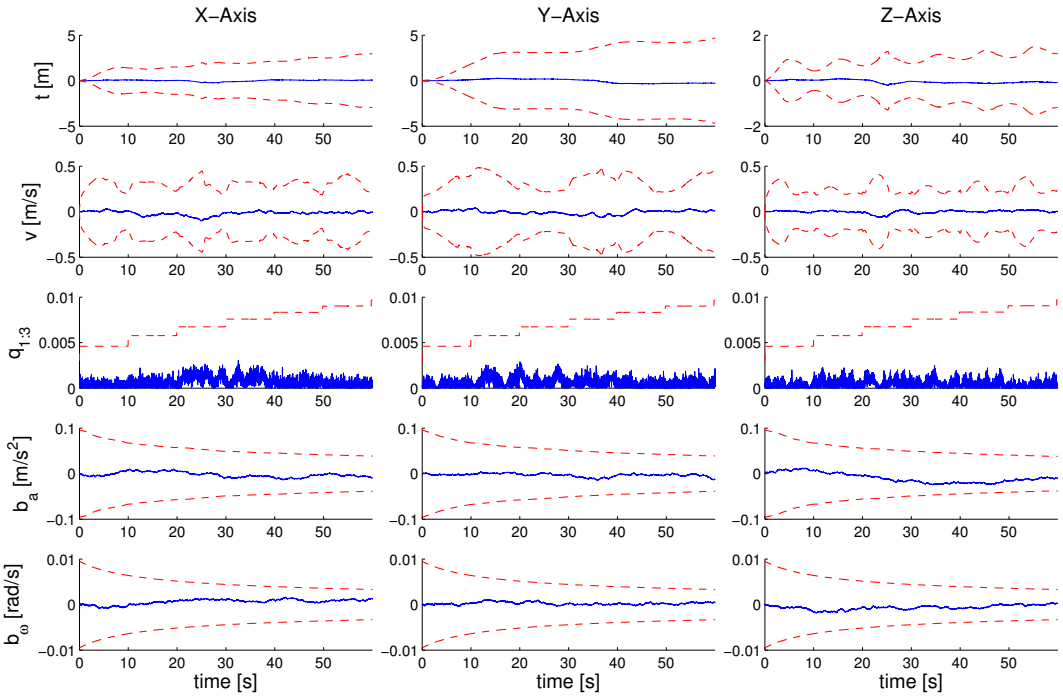
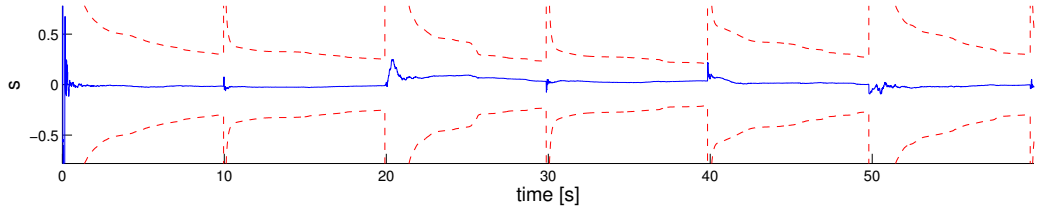


Figure 8.54: These plots denote the simulated motion for the filter experiments. We implemented an oscillating translational acceleration and rotational velocity, together with white Gaussian noise of the same magnitude as seen for real sensors.

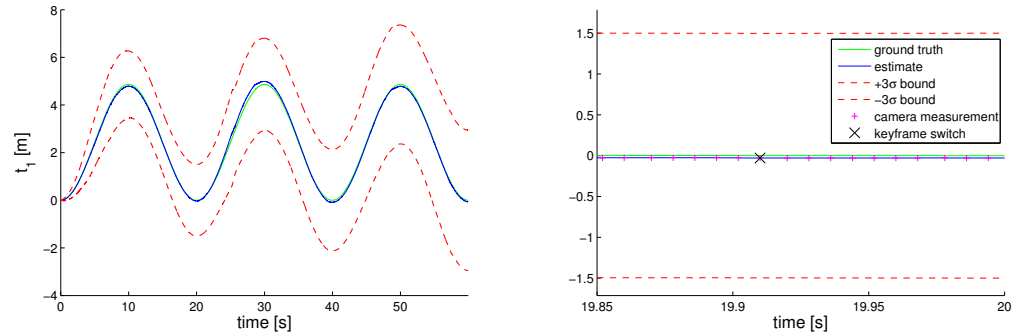


(a) motion and biases

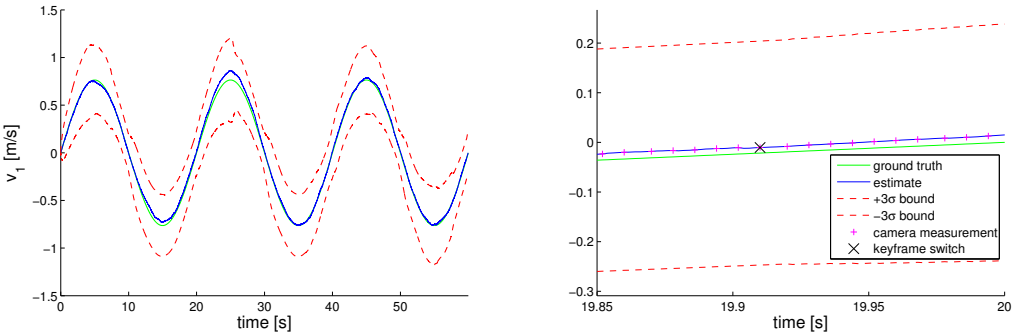


(b) scale of camera translation

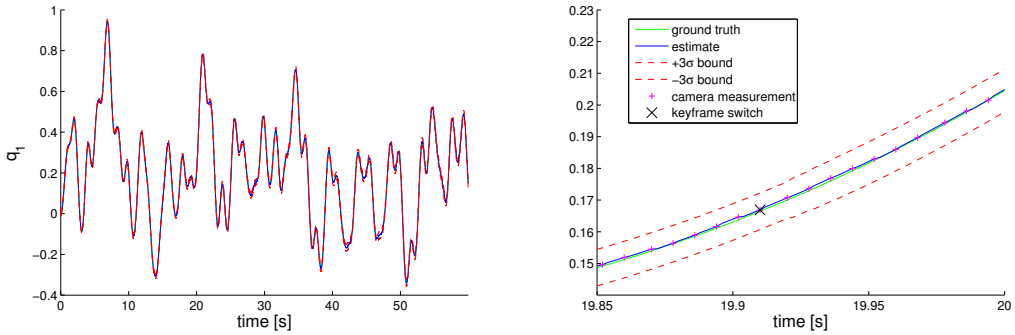
Figure 8.55: This figure illustrates the errors between estimate and simulation (blue) as well as the 3σ-bounds of the calculated uncertainty (red). The errors are always within these bounds, which proofs the proper operation of the filter.



(a) translation estimation

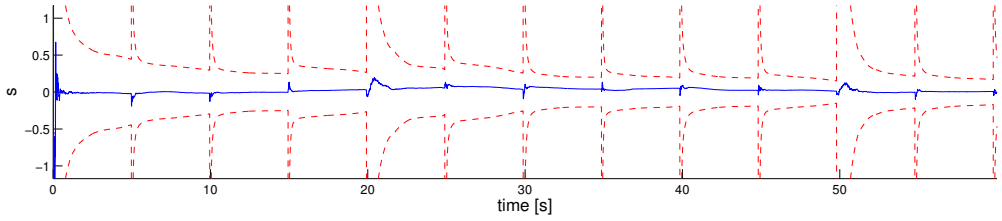


(b) velocity estimation

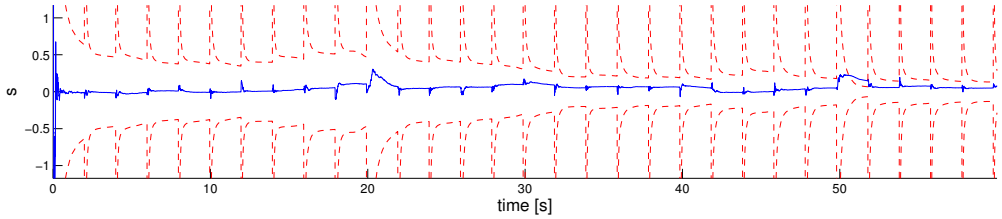


(c) attitude estimation

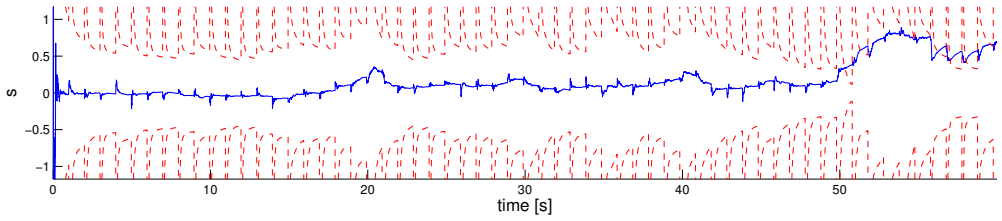
Figure 8.56: These plots show the full motion estimation for the X-axis on the left and a small clipping in detail on the right.



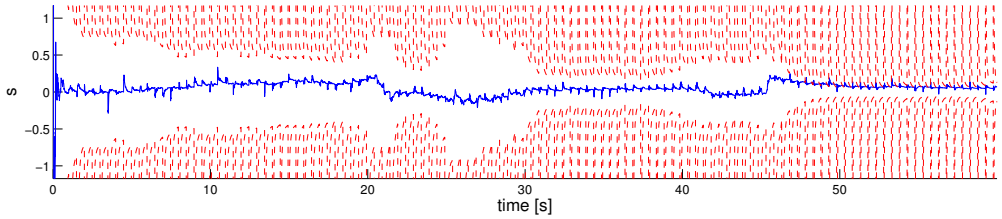
(a) 5 seconds for each keyframe



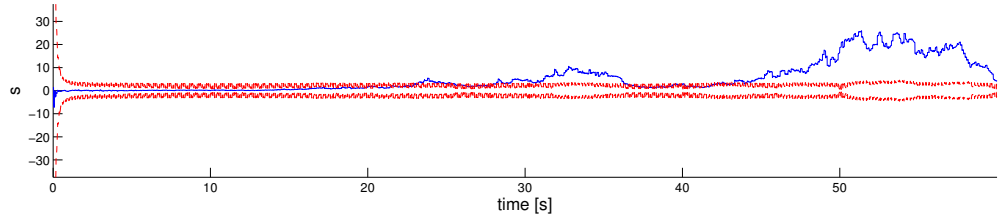
(b) 2 seconds for each keyframe



(c) 1 second for each keyframe



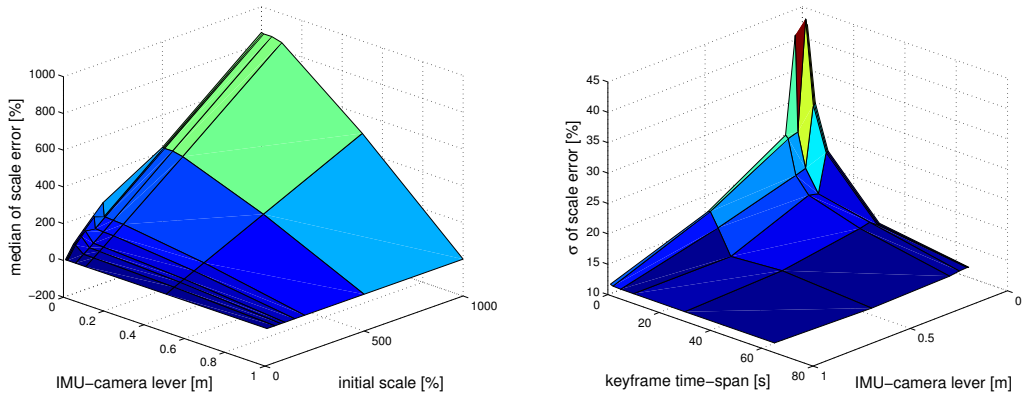
(d) 0.5 seconds for each keyframe



(e) 0.1 seconds for each keyframe

Figure 8.57: The effect on the convergence of the EKF for different keyframe time-spans is illustrated in these plots. Please note the different scaling for the last plot.

Another aspect which we evaluated is the influence of the lever arm between camera and IMU on the filter convergence and stability, as suggested in [211]. For that, we plot the scale estimation results for different lever-lengths against various values for the scale initializations and the time-spans between keyframes in Fig. 8.58. A large lever stabilizes the scale estimation significantly due to the lever effect in $\boldsymbol{H}_{1,8}$ of Eq. 4.100. Wrong scale initializations can be compensated by a large distance between IMU and camera (provided that there are significant rotations). Moreover, the scale converges much faster, which can be seen by a small standard deviation of the scale estimation also for short time-spans between two keyframes.



(a) median scale error: lever vs. initial scale

(b) σ of scale error: time-span vs. lever

Figure 8.58: The left plot illustrates the relation between IMU-camera lever and the quality of the scale initialization. It shows, that a large lever can compensate for a bad initialization. The right plot illustrates the standard deviation of the scale estimation for a different time-span between keyframes and changing IMU-camera lever sizes. It confirms that a large lever size leads to a much quicker scale convergence than a short one.

Chapter 9

Applications

The presented methods and algorithms were also integrated in more complex applications than the experimental validations discussed in Chapter 8. In the following, we will present three of them: the self-referenced 3D-modeling with the DLR 3D-Modeler, virtual environment modeling for surprise detection and a fully autonomous quadrocopter flight.

9.1 Self-Referenced 3D-Modeling

The DLR 3D-Modeler is a multi-purpose platform for geometric and visual perception [222]. It combines multiple sensors in a compact way. Current applications comprise 3D-modeling, tracking, visual servoing, exploration, path planning, and object recognition, *e.g.*, as perception system of the humanoid robot Justin [223]. The system is equipped by following main sensors as illustrated in Fig 9.1:

- The DLR *Laser-Range Scanner* (LRS) [224] is based on laser light triangulation. A visible, harmless laser ray is continuously rotated and its reflection is recorded and used for triangulation of the distance.
- The DLR *Laser Stripe Profiler* (LSP) [225] includes one or two (depending on the configuration) laser beams that project stripes on a surface. These stripes are recorded by cameras which allow to triangulate the distance. The laser stripe detection is done without optical filters to allow for other image processing methods, such as stereo vision, visual egomotion or texturing, to be performed simultaneously.
- The stereo camera consists of two AVT Marlin F-046C progressive scan cameras with 6 mm Sony VCL-06S12XM objectives mounted on them. They provide a

resolution of 780×582 at a baseline of 50 mm[1]. The implemented stereo algorithm is detailed in [226, 227].

- The rigidly attached IMU is the AscTec AutoPilot from Ascending Technologies GmbH[2]. It estimates the 6-DoF based on three gyroscopes, three accelerometers, and three magnetometers at 1 kHz.

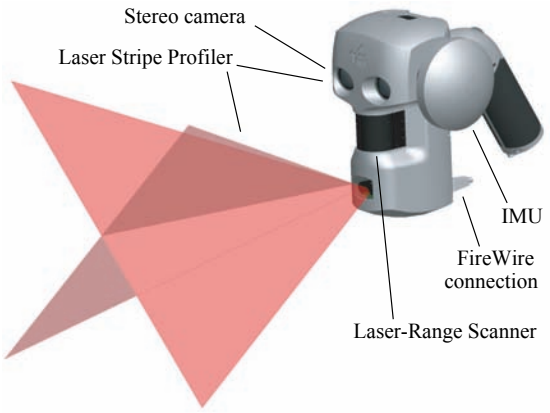


Figure 9.1: The major sensors of the DLR 3D-Modeler are the stereo cameras, the LSP, the LRS and an IMU.

One main application of the DLR 3D-Modeler is as hand-held 3D-modeling system for close-range applications. Several close-range 3D-modeling devices are currently offered - for a large review please refer to [228, 229]. However, the DLR 3D-Modeler is the first extensively mutisensory, passively self-referenced high-rate modeling device.

In general, it is not sufficient to reconstruct an object from a single view, due to object self-occlusions, the object size, or the limited field of operation of the used sensor. Hence, several measurements from different locations have to be fused, which results in two tasks for a 3D-modeling device: The distance to the object's surface has to be measured (range measurement) while for all the different viewpoints a relative alignment in terms of rotation and translation has to be known (relative localization) to allow for proper data fusion.

The 3D-modeling solutions available in literature commonly use external devices to estimate the sensor pose: external tracking systems, turntables, active or passive robotic manipulators, electromagnetic devices, and so on. They all have the drawbacks to limit the user's workspace, to require an error-prone registration of the spatial and temporal

[1] The base distance and the focal length allow for a range sensing from approx. 30 cm up to 2 m.
[2] www.asctec.de

alignment and, lastly, they are in general the most expensive part of the modeling system. The DLR 3D-Modeler provides three solutions to measure the distance to an object: the LRS, stereo-triangulation or the LSP (as illustrated in Fig. 9.2).

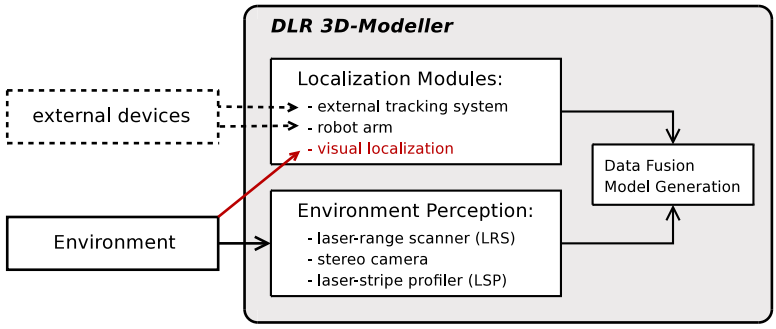


Figure 9.2: This figure illustrates the modules provided within the DLR 3D-Modeler. Due to the visual localization module, there is no external device necessary anymore.

The pose of the device, which is required for measurement fusion, can be measured by an external tracking system[1] or by the inverse kinematics when mounted on the wrist of a robot arm as illustrated in Fig. 9.3.

Figure 9.3: The upper left-hand side picture shows the DLR 3D-Modeler with the IMU module and the PC running the software. The other two images demonstrate each a global registration variant for the scan device: a robot arm (*lower left*), and an external tracking system (*right*).

[1] ARTtrack1 cameras from ART, www.ar-tracking.com

By applying image-based motion estimation techniques which are based on the on-system stereo cameras, we do not depend on external devices anymore, but can simply rely on the compact 3D-Modeler to solve the two necessary tasks for 3D-modeling. The 3D-model is generated online, hence the depth-measurements are fused immediately which requires the poses in real-time. It cannot be fixed by optimizing all measurements, because not all the information is available anymore due to the online fusion and mesh generation which makes it difficult to realize an offline bundle-adjustment. Hence, the estimated poses have to be highly accurate and reliable. Furthermore, it is appreciable that the localization is not only valid within one scan, but it provides a global reference, so that repeated scans, which are acquired at different time instances, can be merged. This results in three major requirements for visual localization algorithms when applied to online 3D-modeling: *real-time pose estimation*, *high positioning accuracy* and a *global reference*.

Our solution for the real-time image-based egomotion estimation of the 3D-Modeler consists of the following modules (as illustrated in Fig. 9.4):

- Shi and Tomasi features are extracted from both images of the stereo camera system and initialized as described in Section 7.1.1.
- The features are tracked by the sped up extended KLT tracker implementation introduced in Section 7.1. The tracking is supported by the IMU as denoted in Section 4.3.
- A feature-set management system allows for keyframe-based VSLAM as mentioned in Section 5.2.
- The pose is estimated using RVGPS as explained in Section 5.2.1.
- To provide a global reference we use SURF features as described in more detail in Appendix A.3.

Fig. 9.5 shows modeling results acquired by a two-turn sweep of the DLR 3D-Modeler. Because the device is manually guided, the points are not uniformly distributed on the object's surface.[1]

[1] An example of the operation is available at http://www6.in.tum.de/~maire/videos/3dMo_putto.mp4.

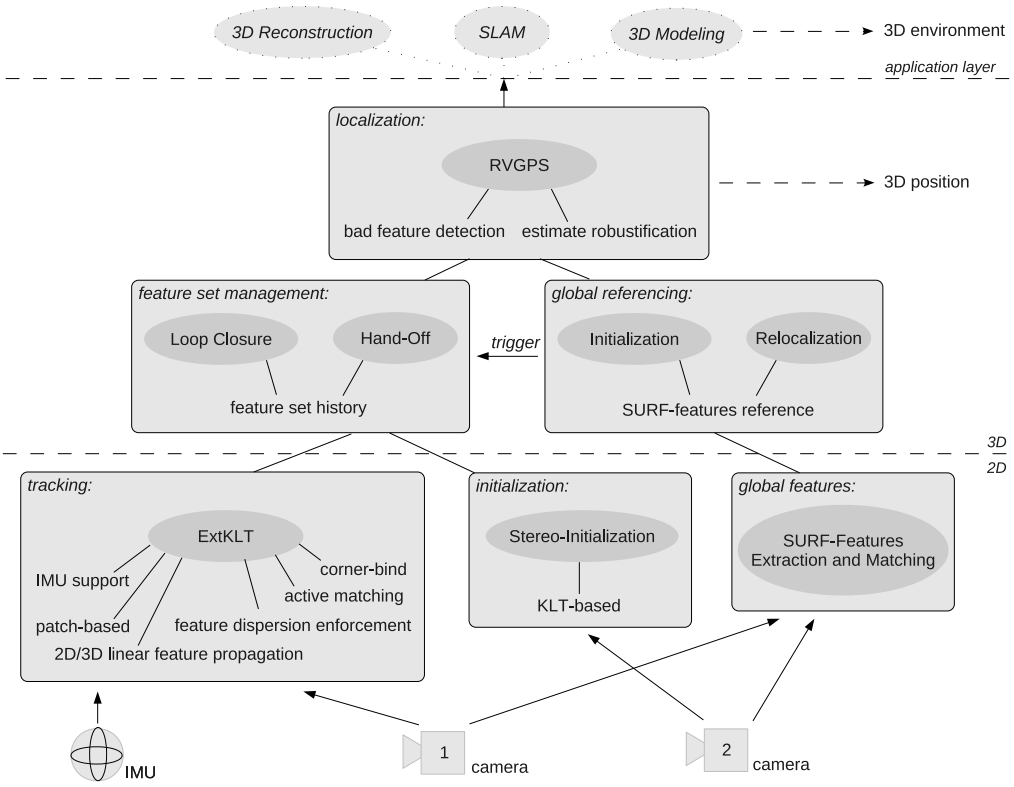


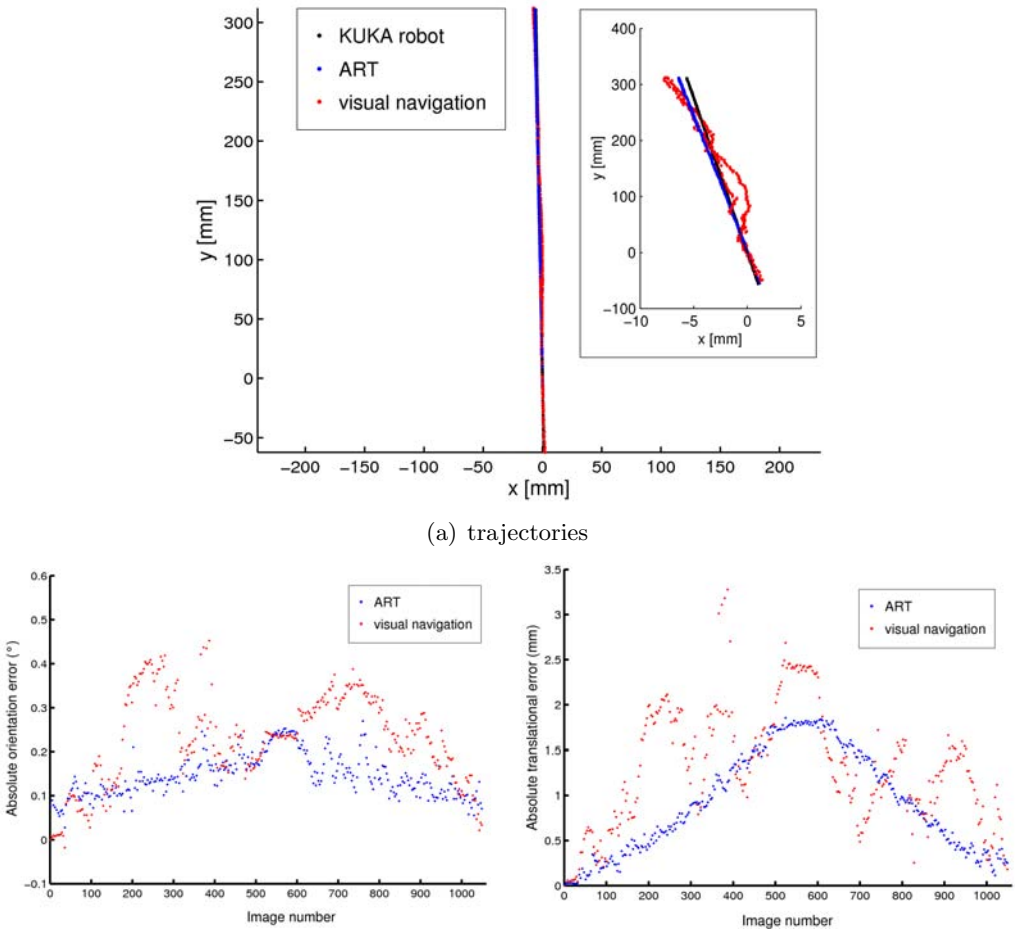
Figure 9.4: This block diagram illustrates the modules, which are used for the image-based localization.

(a) 3D point-cloud: statue in front of a box. (b) 3D point-cloud of a putto.

Figure 9.5: Two 3D point clouds acquired by the DLR 3D-Modeler using the presented visual navigation algorithm.

9.1.1 Accuracy Evaluation

In the following experiment, the ART-system, a robot arm (KUKA KR16) and our visual navigation approach are compared (see Fig. 9.3). Therefore, the 3D-Modeler is mounted on the flange of the KR16 and while measuring the poses with the ART-system and the robot's kinematics, the images for the visual localization algorithm are acquired. To simulate an ordinary scan sweep, the robot arm was commanded to move down 40 cm and up again in front of an object. Fig. 9.6 illustrates the estimated



(a) trajectories

(b) rotational error

(c) translational error

Figure 9.6: *a)* The scan trajectories computed by three different references. By zooming in the x-axis (small plot) the error becomes evident. The absolute rotational error (*b*) and the absolute translational error (*c*) are referenced to the KUKA poses.

trajectories of the scanner and the absolute rotational and translational difference of

the ART-system and the visual navigation to the KUKA robot. While with equally scaled axes the differences in the trajectories are not really recognizable, the zoom in the left figure makes them visible. Due to the inaccuracies of the robot-to-camera (hand-eye) and ART-to-camera calibration, a small discrepancy between the results of the robot and the ART exists. This error is also illustrated in the blue dots of the right plots, where the KUKA robot estimates are used as ground truth due to its high accuracy. The visual localization result is noisier, but the average error is comparable to the ART-system. Five keyframes are acquired during the sweep and refound on the way back.

During longer sweeps without loop closure, the visual navigation results are drifting like in every SLAM-based method, while the ART-system provides drift-less results within its workspace. This is the most problematic drawback regarding external registration devices and the price for compactness, low costs and especially arbitrary workspace location and size.

Fig. 9.7 shows online computed triangle-meshes for the statue in Fig. 9.7(a), once acquired with the ART-based global registration (Fig. 9.7(b)) and once with the presented visual navigation approach (Fig. 9.7(c)). The 3D-Modeler was hand-guided in front of the putto, in the same way as in the experiment with the KUKA[1]. The distance to the object is measured by a laser-stripe profiler (LSP) and the acquired data is fused, together with the global pose, to 3D-meshes [230]. Due to the smoothing by the data fusing process, the noise in the visual navigation estimates is removed and the results are comparable. The differences in the meshes result from the fact that two hand-held runs were necessary to evaluate the online processing performance, which provides unequal views for the LSP.

Regarding the quality of the pose estimation, the following points are worth mentioning: The visual navigation is well-conditioned if the features are spread out over the whole image. Furthermore, accurate results are provided if the landmarks are accurately initialized, which depends on the features' distance and the base-line of the cameras. The ART-system has the systematical drawback of its inaccurate calibration capabilities. It is difficult to calibrate the setup only by the six small and tight spheres, which are quite far away from the cameras. If the markers are occluded by the sensor device or by the object to scan, the accuracy of the pose estimation is massively reduced. Moreover, the errors of the LSP and the ART do not rely on the same images and, thus, the error correlation is worse compared to the visual navigation. The robot arm is the most precise of the three variants, provided an accurate hand-eye calibration.

[1] An example of the operation is available at
http://www6.in.tum.de/~maire/videos/3dMo_liveModeling.avi.

(a) putto statue (b) mesh based on ART (c) mesh based on visual ego-motion estimation

Figure 9.7: *a)* the scanned putto-statue. *b)* triangle-meshes based on the ART-system. *c)* triangle-meshes based on the visual localization. All measurements were done in real-time and also the meshes where computed online.

However, it lacks the same drawbacks as the ART-system and, in addition, the degrees of freedom of the scan device are reduced at the workspace border.

In Section 3.3.4, we explained the concept of active matching (AM). While related approaches use this technique for monocular motion estimation with unknown structure, we can measure the features' depth which drastically reduces the search area for features by applying AM. Non-stochastic AM with known structure can be seen as best case scenario for AM because it achieves an especially good performance if the priors on the relative location is strong - we actually assume to have an error-free 3D structure. On the other side the performance of AM is not so much influenced by weak priors on the absolute location. Again we assume a linear translational motion model in 2D and propagate the features according to this model. As next step, we look for a strong feature in an image corner by exhaustive template matching over a large search area. Based on this match we can solve for two DoFs of the rotation. This reduces the search area of the second feature significantly, which is chosen from the opposite image corner. Using the second match one can solve for the total rotation and propagate the rest of the features. The real features are in general found within a few pixels of the estimated location. By applying the AM concept, we can track features at similar high dynamic motions, solely based on the camera, as by using the IMU based feature propagation[1].

[1] An example of operation is available at
http://www6.in.tum.de/~maire/videos/3dmo_AM_iros10_video.avi.

9.2 Visual Environment Modeling

Geometric environment models are still the preferred space representations in robotics. They are intuitive and allow for a reliable navigation and manipulation of mobile robots. Laser-range devices are the preferred sensors to generate such models. However, there are many objects which cannot be reconstructed reliably, *e.g.* glasses, specular objects or complex structured plants. To deal with such objects it is often of advantage to not consider them in the geometric but in the visual domain.

One type of image-based scene representation that recently has become very popular uses view-dependent geometry and texture. Instead of computing a global geometry model which is valid for any view point and viewing direction, the geometry of the scene is estimated locally and holds only for a small region in the viewpoint space. It has been shown that this approach is suitable especially when the scene contains specular and translucent objects. Hence, per-pixel depth maps are calculated for each reference image. While this is done off-line, the view selection and view synthesis are performed on-line. The view selection only chooses a small subset of all reference images which contribute to view interpolation, each time a new frame is rendered. In real-world environments, most surfaces are non-Lambertian which means that the reflected intensity depends on the position of the viewer. Hence, in our approach, the reference cameras are ranked in terms of their orthogonal distance with respect to predefined rays within the viewing frustum of the virtual camera. Novel virtual views are synthesized in a two-pass procedure: the color data of the selected reference images is warped into the virtual view where afterwards the final color of each pixel is determined using probabilistic models. These virtual environment modeling has been used within the elite cluster "*Cognition for Technical Systems*" (CoTeSys[1]) to develope image-based surprise trigger which can detect missing objects or changes on objects, which are difficult to detect by LSR sensors or/and to model in geometric space. Three steps are necessary to achieve this:

- **Scene acquisition and pre-processing**: An image sequence of the scene for which the virtual environment model needs to be generated has to be acquired. The camera-pose of each viewpoint has to be known. For that we use the same keyframe-based visual localization algorithm as described in Section 9.1. Once the images are registered, image-depthmaps are generated by stereo-triangulation. If several runs are performed at varying lighting conditions it is possible to compute an illumination-invariant scene representation, which allows render lighting-independent views of a scene (see [5] for more details).

[1] http://www.cotesys.org/

- **Relative localization and virtual view rendering**: Once a current view shall be compared with the previously acquired data set, the actual position relative to the scene acquisition origin has to be determined. For that we use global features from natural landmarks as described in Appendix A.3. Knowing the spatial relation, the virtual image can be rendered.

- **Surprise detection**: We propose a scheme for Bayesian visual surprise detection based on the probabilistic concept for view synthesis. Similar to the processing of color information in the human visual system, we compute from each RGB reference image a luminance signal and two color opponency signals (red-green and blue-yellow), respectively. Thus, surprise detection does not have to be performed jointly in RGB-space but can be done independently in three decoupled pathways. The luminance samples warped into the virtual view from the reference images are assumed to be drawn from a Gaussian distribution. We put a prior distribution over the precision, which is equivalent to the reciprocal variance of the Gaussian distribution. The Kullback-Leibler divergence (KLD) as the difference between the posterior distribution and the prior distribution serves as a quantitative measure for surprise. The KLD is evaluated for each pixel in the virtual image which yields a pixel-wise surprise trigger.

In the following we are going to denote two CoTeSys-scenarios where we applied the presented surprise trigger.

9.2.1 Cognitive Household

The episode we envisioned within the assistive household scenario is the acquisition of an image-based model of a typical household environment, which is the basis for cognitive processes. With our module a cognitive robot in the household should be able to update its environment model at any time by surprise detection and classify the objects around it into static or dynamic ones. Moreover, surprise about unexpected events should influence the robot's action plans. Robust visual localization should enable it to retrieve its current position in the environment and to evaluate its current observation with respect to the already acquired reference model.

Fig. 9.8 illustrates the localization results on an image sequence acquired while performing a quarter circle with 0.9 m diameter. The odometry of the Pioneer 3-DX is not accurate enough to build a model of the environment from the measurements. The results of the different processing steps on the acquired image-sequence are shown in Fig. 9.9. The glasses, which have been removed in a second run are clearly detected by the surprise trigger as shown in Fig. 9.9(c).

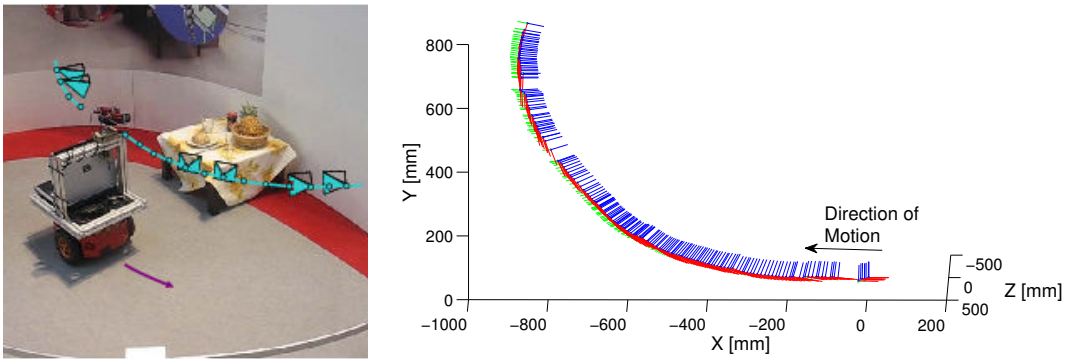


Figure 9.8: Localization results of a Pioneer 3-DX performing a quarter circle as illustrated in the left image. The blue lines show the viewing direction of the capturing camera. The white gaps in the trajectory are due to swapping on the hard disk. The cameras were slightly tilted to the floor and so the trajectory is not only within the X-Z plane.

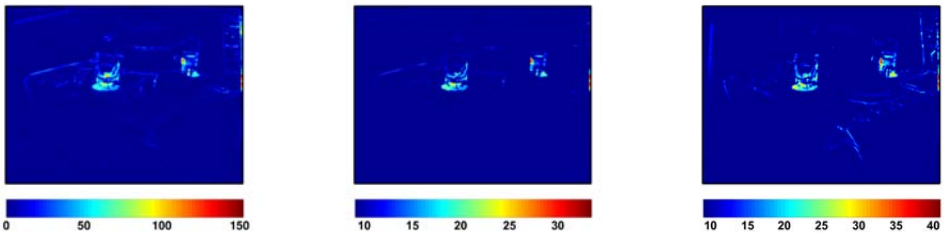
9.2.2 Cognitive Factory

This application was a joint project with JAHIR ("*Joint-Action for Humans and Industrial Robots*"). The goal is that a robot mounts a product while communicating with a human worker by speech and virtual buttons which are projected onto the working table. We are going to integrate our algorithms described in this work into the quality assessment step after the product is mounted. To this end, an image sequence of the mounted error-free product is first captured and processed for a reference image-based model. Our algorithm for surprise detection detects unforeseen changes in the product which might be due to failures in the single production steps. It is desirable, that the inspecting camera is localized with respect to the product and not with respect to the robot's coordinate system since this does not require that the product is exactly at the same position as during the acquisition of the reference model. The cognitive robot uses this surprise trigger for making decisions about repair strategies and the next production steps.

The results of the VEM-based surprise trigger on the JAHIR sequence are shown in Fig. 9.10. As one can easily recognize, the visual pose estimation is not always accurate. This is due to the highly specular surface of the workpiece and that features can only be found in a limited field of view which reduces the localization accuracy significantly as seen in Section 8.2.1. Nevertheless, the virtual rendering works fine, because the motion estimation and the rendering rely on the image domain and, hence, the errors correlate. The camera cannot be localized accurately if there is not much difference between two views, but this also means that one cannot find any visible error in the rendered image neither.

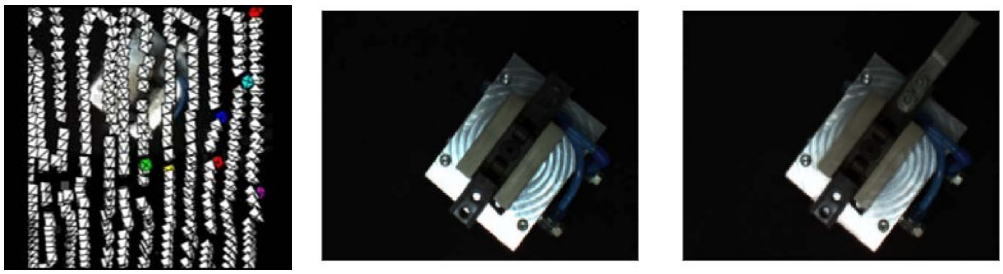
(a) preprocessing: visual localization, depth map generation and virtual rendering.

(b) virtual view rendering: *left*: observation, *middle*: rendered view and *right*: reference image for comparison (closest view - manually chosen)

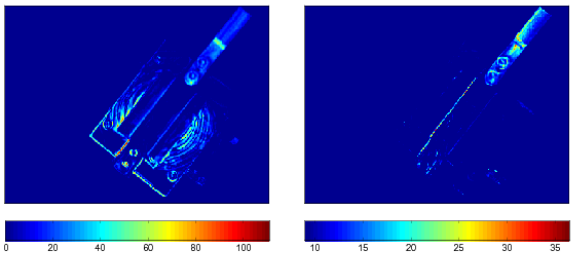


(c) surprise detection: *left*: image differences (manual sub-pixel accurate alignment using KLT), *middle*: Bayesian inference (manual sub-pixel accurate alignment using KLT) and *right*: Bayesian inference (automatic global referencing using SURF)

Figure 9.9: This image sequence visualizes the various processing steps for the VEM-based surprise trigger. In order to test our algorithm for surprise detection, we captured another image sequence on a trajectory which was close to the first one but not identical. We changed the scene by removing the two glasses. The task of the cognitive system is to detect these changes. This image is depicted in the middle row on the left together with a photorealistic virtual image (middle) rendered from reference images which were selected only from the first image sequence. The right picture shows a view of the reference sequence which has been manually picked as comparison with the virtual rendered image – Please note that there is no real camera image which was acquired exactly at the position of the observation.



(a) *left* estimated camera locations, *middle* observation and *right*: rendered view



(b) surprise detection: *left*: image differences and *right*: Bayesian inference

Figure 9.10: This image sequence visualizes the VEM-based surprise trigger results on the JAHIR sequence. The upper left image shows the acquired camera views and the upper middle and right picture illustrate the observation and the virtual rendering. The bottom row denotes the surprise trigger result based on image differences and Bayesian inference.

9.3 Autonomous Quadrocopter Flight

A first step towards fully autonomous flying *micro aerial vehicles* (MAVs) has been shown at the Automatica 2010[1]. We equipped a Hummingbird quadrocopter from Ascending Technologies GmbH[2] with a Beagleboard[3] (revision C3) and a Philips webcam SPC900NC[4]. The demonstration should show that it is possible to process camera images on a resource-limited system as available on MAVs in real-time for autonomous flight control. In the case at hand, we were limited to the ARMv7 core of the Beagleboard running at 600 MHz. The quadrocopter platform is equipped with an IMU and a low-level controller which stabilizes the quadrocopter in-flight but cannot prevent its drifting. Our image-based motion estimation provides a local reference and, thus, suppresses drifting.

(a) system setup (b) in-flight

Figure 9.11: The left picture illustrates the bottom-view of the system setup, showing the Beagleboard and the Philips webcam. On the right hand side the quadrocopter is shown while hovering fully autonomously above a hand-held board with the artificial markers printed on top.

In this scenario, five blue blobs are chosen as artificial markers, defining a coordinate frame. A highly efficient blob detection algorithm is used to identify the sub-pixel accurate location of the blobs in the image. We use RVGPS, as described in Section 5.2.1, to estimate the image based pose relative to the blob-frame. This pose is then fused with the IMU measurements in an EKF and the result is fed into the non-linear controller. Due to the artificial markers we can move the reference frame while the quadrocopter is following it[5]. The setup of the system and an image of the flight are illustrated in

[1] http://www.automatica-munich.com/
[2] www.asctec.de
[3] http://beagleboard.org/
[4] http://www.philips.com/
[5] An example of operation is available at http://www6.in.tum.de/~maire/videos/quadro_automatica.wmv.

Fig. 9.11.

The images are processed at 25 Hz but due to a long delay between image acquisition and availability at the host (up to 100 ms) the artificial landmarks left several times the FOV of the quadrocopter. If the blobs can not be seen for a long time, the drifts of the strapdown computation lead to the divergence of the pose estimation and the quadrocopter can not find his way back anymore. Due to the lack of robustness of this concept no further experiments with artificial markers have been made and we have been focusing on a solution with natural landmarks. The required modules for such an approach have been presented in this thesis. The integration of the motion estimation on a flying platform is part of current research.

Chapter 10

Conclusion

In this work, we presented technical solutions for the biologically motivated navigation approach as introduced in Chapter 1. Solutions for several parts of the proposed navigation flow, illustrated in Fig. 10.1, are provided, together with experimental validations and comparisons of the algorithms. Starting at the low-level processing, we worked on feature detection and tracking solutions, motion estimation and its error propagation as well as sensor registration and sensor fusion methods. Assuming feasible state-of-the-art camera and IMU calibration techniques as well as a valid strapdown computation, we are able to provide efficient inertial-visual odometry which also works for high dynamic motions, while limiting its drift.

In the following few sections, we discuss the presented algorithms in a methodic and task-specific context, we summarize our contribution and discuss future work.

10.1 Discussion

In this work, we aim for an efficient and robust navigation concept which works also with high dynamic motions. A sensor and navigation framework based on the physiology of insects and behavioral experiments on them has been derived in the introduction. Based on this proposal, we extracted constraints for the image-based motion estimation and attempted to realize them in our algorithms. We achieved efficient feature detection and tracking solutions, like AGAST, which allow for the processing of a large number of features at high frame-rates similar as in the visual system of insects. Furthermore, we mentioned that insects behaviorally separate rotational and translational motion. This is a restriction which is in general difficult to realize on technical systems. Nevertheless, we presented the Z_∞-algorithm, a method which implements a separated translation and rotation estimation under certain circumstances, which are in general provided for

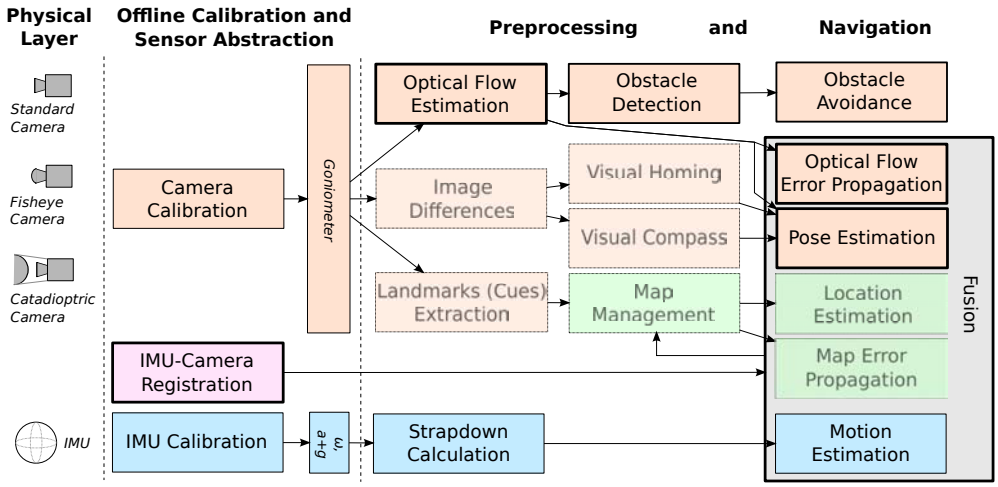


Figure 10.1: In this figure, we illustrate again the flow diagram which we proposed in the motivation and already depicted in Fig. 1.3. However, this time we highlight the modules for which we presented contributions in this thesis and greyed out the ones which are in the focus of future work. At this point we are able to provide efficient inertial-visual odometry. The next major steps consist in providing a sort of topological map as global reference and implementing efficient homing algorithms.

outdoor applications. In simulations we have shown, that such a motion estimation yields a more accurate pose computation than state-of-the-art closed-form solutions which estimate rotation and translation together[1].

A crucial aspect is also the identification of a camera system which yields the most accurate motion estimation. Thus, we evaluated the performance of motion estimation algorithms regarding tracking accuracy, number of features and aperture angle. The compound eye of insects offers a large FOV which covers almost the whole sphere, while providing only a rather small angular resolution compared to conventional cameras. According to our experiments, the accuracy of the motion estimation does not increase continuously with the FOV. The results reveal, that a trade-off between conditioning of the landmark set and accuracy of the feature correspondences has to be found. An unduly small aperture angle yields bad results since the motion estimation is ill-posed, while an inadequate large FOV for the same number of pixels also yields bad results due to the decreasing accuracy. There is an optimum around 90° aperture angle where the conditioning is saturated and cannot get better. The large FOV of insects could be explained by the fact that their visual sensor does not only have to provide odometry

[1]This is, of course, only valid under the premise that sufficient translation-invariant features are available.

measurements, but it also has to allow for a reliable scene monitoring and other vision-related tasks (*e.g.*, the visual compass). Furthermore, the accuracy of the visual motion estimation does not only depend on the FOV and the tracked landmarks, but also on the computation method. As already mentioned in Section 3.3, several motion computation techniques are known which require an omnidirectional view. It is not clear yet, how the visual odometry is determined in the insect's brain and, thus, a large FOV could also be a stringent condition for insects, even though it is not a general requirement for motion calculation. The simulations have also shown that the accuracy of the pose estimation can be improved by increasing the number of features, the tracking accuracy or, of course, both. However, it is obvious, that the former yields a more robust solution in the presence of outliers.

For a proper fusion of measurements, it is essential to provide an estimate of the expected motion accuracy based on a given feature set. The presented first order propagation has shown to provide adequate error estimates. However, we still needed to provide the embodiment of the camera and the IMU in terms of a spatial and temporal registration. We have shown, that a proper temporal alignment is crucial if high dynamics are applied. Once the registration is known, it is still not trivial to fuse both sensors due to the lack of translational scale of the camera. This estimate suffers from long convergence rates, which can be reduced by high frame-rates or a large distance between IMU and camera, assuming that a significant rotation is provided. We proposed an indirect Kalman filter where the state transition and the measurement uncertainties can be taken into account in their respective coordinate frame. Moreover, it is possible to fuse relative pose measurements, which is required for methods which do not rely on global references, like a map. We also suggest to support the feature tracking with the IMU measurements, which allows to use local trackers also for high dynamic motions.

10.2 Contributions

The three most significant contributions of this thesis are:

- **AGAST-based tracker**: The currently most efficient corner detector and feature tracker. At the time of writing this thesis the open-source package has been downloaded about one thousand times. Further, it is planned to be released as part of the next OpenCV[1] and PCL[2] release. Nonetheless, it is already used in other algorithms with high impact, such as BRISK. [4]

[1] http://opencv.willowgarage.com
[2] http://openpcl.sourceforge.net

- **Batch-optimization based IMU-camera registration**: Our presented spline-based spatial alignment achieves more consistent results than state-of-the-art solutions, due to the batch-optimization and a more accurate motion representation. [2]

- **DLR-3dMo egomotion estimation**: The VSLAM-based egomotion estimation reduces the DLR 3D-Modeler to a compact, handy device without physical workspace limitations. As result, the system is the first extensively mutisensory, passively self-referenced high-rate modeling device. [8, 6]

Other major contributions are the **$\mathbf{Z}_\infty$-algorithm**, which is an accurate and efficient alternative for motion estimation in outdoor applications [7]. We derived an **error propagation** for this algorithm as well as for the eight-point algorithm. For the latter we consider also the different implementation variants and derive the uncertainty propagation for the rotation matrix. Moreover, we proposed an EKF based **fusion of IMU and camera** which allows for relative motion updates and considers the measurement and state uncertainties in their respective spatial dimensions. Beside this, we were one of the first showing a **fully autonomous MAV** processing images for pose estimation on-board at high frame-rates.

Our contributions have been published in multiple publications listed in the publications section.

10.3 Future Work

Before the discussion of the future work towards the task of efficient and reliable mobile robot navigation, we will treat each of the presented algorithms independently.

10.3.1 Methods Specific Future Work

Spatial Alignment

To allow for higher dynamics, we want to evaluate sequences with natural landmarks in the future. However, without any knowledge about the environment we can only estimate the camera translation up to scale. The presented framework already provides such a scale parameter, which makes it easy to adapt. It would be interesting to evaluate, whether the higher signal to noise ratio of the IMU would allow for still more accurate calibration even with less precise landmark tracking.

The number and the location of the B-spline knots has been determined empirically in this work. Estimating the optimal number and location of the knots would speed

up processing and make it adaptive to arbitrary dynamics. First experiments in this direction are promising.

Another possible improvement is a direct fusion of the landmark locations with the IMU motion, avoiding the currently preceding camera pose estimation. While this will increase the complexity of the optimization framework, we expect higher accuracy.

Sensor Fusion

The presented sensor fusion framework is going to be tested and evaluated on real data, combining efficient tracking, pose estimation and fusion. Furthermore, it will be interesting to see which effects the cross-correlation, introduced by the pseudo-measurement update, has on the fusion stability. We do not exclude, that one gets a more stable filter with better convergence properties by keeping the unmodeled cross-correlations in the measurement uncertainties. Furthermore, we would like to address the cross-correlation between the camera based scale and pose estimation as well as the bias of the accelerometers to improve the stability of the filter.

Motion Estimation

A combination of the eight-point and the Z_∞-algorithm could overcome the limitations of the Z_∞-approach to outdoor environments, while still yielding a higher accuracy than the eight-point algorithm. A nonlinear optimization step at the end, which uses the closed-form solution as starting point, could further improve the estimate. However, it depends on the application whether such a high accuracy at the cost of processing time is required.

An intelligent parameter adaption for the Z_∞-algorithm would simplify and generalize its application. The parameters could, *e.g.*, be adapted based on the previous estimation step.

Feature Tracking

It would be interesting to improve the tracking of AGAST to sub-pixel precision. Thus, one could adapt the accuracy of the motion estimation depending on the application and the processing resources. Beside that, image pyramids can be used to preserve the local tracking assumptions even at low frame-rates.

10.3.2 Task Specific Future Work

Considering again the proposed navigation concept as illustrated in Fig. 1.3 and 10.1, we presented solutions for the modules IMU-camera registration, measurement fusion, optical flow estimation, motion estimation, and its error propagation. Moreover, we assume that the state-of-the-art provides feasible approaches for camera and IMU calibration and strapdown computation. Hence, the next steps will be to implement a topological map based on image features and to come up with efficient homing strategies, which allow the robot to find back to an exact position. Which features to use and how the map should be organized is still an open question and part of current research. However, we are of the opinion that not only one kind of features, but several different and complementary features should be used. Moreover, the idea of a visual compass based on omnidirectional cameras is also highly interesting and in focus of current work.

Publications

[1] Mair, E., M. Fleps, M. Suppa, and D. Burschka. **Spatio-Temporal Initialization for IMU to Camera Registration**. In *International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2011. ***Best Paper Finalist***.

[2] Mair, E., M. Fleps, O. Ruepp, M. Suppa, and D. Burschka. **Optimization Based IMU Camera Calibration**. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2011.

[3] K.H. Strobl, Mair, E., and G. Hirzinger. **Image-Based Pose Estimation for 3-D Modeling in Rapid, Hand-Held Motion**. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2011.

[4] Mair, E., G. Hager, D. Burschka, M. Suppa, and G. Hirzinger. **Adaptive and generic corner detection based on the accelerated segment test**. *European Conference on Computer Vision (ECCV)*, pages 183–196, 2010.

[5] W. Maier, F. Bao, Mair, E., E. Steinbach, and D. Burschka. **Illumination-invariant image-based novelty detection in a cognitive mobile robot's environment**. In *International Conference on Robotics and Automation (ICRA)*, pages 5029–5034. IEEE, 2010.

[6] Mair, E., K.H. Strobl, T. Bodenmüller, M. Suppa, and D. Burschka. **Real-time Image-based Localization for Hand-held 3D-modeling**. *KI-Künstliche Intelligenz*, **24**(3):207–214, 2010.

[7] Mair, E. and D. Burschka. *Mobile Robots Navigation*, chapter Z_∞ - Monocular Localization Algorithm with Uncertainty Analysis for Outdoor Applications, pages 107 – 130. In-Tech, March 2010.

[8] Mair, E., K.H. Strobl, M. Suppa, and D. Burschka. **Efficient camera-based pose estimation for real-time applications**. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2696–2703. IEEE/RSJ, 2009.

[9] K.H. Strobl, Mair, E., T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger. **The self-referenced DLR 3D-modeler**. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 21–28. IEEE/RSJ, 2009. ***Best Paper Finalist***.

[10] W. Maier, Mair, E., D. Burschka, and E. Steinbach. **Visual homing and surprise detection for cognitive mobile robots using image-based environment representations**. In *International Conference on Robotics and Automation (ICRA)*, pages 807–812. IEEE, 2009.

[11] D. Burschka and Mair, E. **Direct Pose Estimation with a Monocular Camera**. In *2nd International Workshop on Robot Vision*, pages 440–453, 2008.

[12] D. Burschka and Mair, E. **Biologically Motivated Optical Flow-Based Navigation**. In *Workshop Biological Models in Spatial Cognition*, 2008.

[13] W. Maier, Mair, E., D. Burschka, and E. Steinbach. **Surprise Detection and Visual Homing in Cognitive Technical Systems**. In *1st International Workshop on Cognition for Technical Systems*, 2008.

[14] Mair, E., W. Maier, D. Burschka, and E. Steinbach. **Image-Based Environment Perception for Cognitive Technical Systems**. In *1st International Workshop on Cognition for Technical Systems*, 2008.

Appendix A

Appendix

A.1 Image Pyramids

The concept of image pyramids is a powerful way to reduce processing time by iteratively refine an image processing task. For that, the image is downsampled into one or more images of lower resolution and the required task is executed first on the smallest image and its result is used as starting point on the next higher level of resolution. This process is repeated until the original image is reached.

Most processing libraries support image pyramids but most of them have prebuilt methods which reduce the image size by a factor of four in each step. There is no way to define other subsampling step-sizes or to change the smoothing factor before subsampling. In this section, we want to show how a proper filter-length for a certain step-size is calculated.

Subsampling suffers from aliasing if the signal, in our case the image, is not lowpass-filtered (smoothed) in advance. In general Binomial filters are used to allow for an efficient implementation. This filter type is an approximation of Gaussian filters with the characteristic to converge to a Gaussian for large filter orders. The coefficients of a Binomial filter can be computed by the binomial formula

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \tag{A.1}$$

and correspond to the respective level n of the Pascal's triangle as illustrated in Table A.1.

The question we want to answer here is which filter mask should be applied for a certain downsampling. To prevent aliasing in the downsampled pyramid levels, all the frequencies larger than the cut-off frequency f_c should be suppressed. The cut-off

Table A.1: Binomial coefficients (Pascal's triangle) as used for the Binomial filter of order n, with the normalization factor $s = 2^{-n}$ and the filter variance $\sigma^2 = \frac{n}{4}$. To compute an image pyramid only the bold masks should be used due to their symmetry characteristic required for image pyramids.

n	s	*filter coefficients*	σ^2
0	1	1	0
1	$\frac{1}{2}$	1 1	$\frac{1}{4}$
2	$\frac{1}{4}$	**1 2 1**	$\frac{1}{2}$
3	$\frac{1}{8}$	1 3 3 1	$\frac{3}{4}$
4	$\frac{1}{16}$	**1 4 6 4 1**	1
5	$\frac{1}{32}$	1 5 10 10 5 1	$\frac{5}{4}$
6	$\frac{1}{64}$	**1 6 15 20 15 6 1**	$\frac{3}{2}$
7	$\frac{1}{128}$	1 7 21 35 35 21 7 1	$\frac{7}{4}$
8	$\frac{1}{256}$	**1 8 28 56 70 56 28 8 1**	2

frequency has to be chosen according to the Shannon-Nyquist theorem which says that only frequencies up to half the sampling frequency f_s can be represented

$$f < \frac{f_s}{2} \quad \Rightarrow \quad f_c \stackrel{!}{=} \frac{f_s}{2\,p_{ss}}\,, \tag{A.2}$$

with p_{ss} denoting the number of pixels skipped by the subsampling.

The attenuation of a certain frequency can be obtained by the frequency response of the 1D binomial filter ${}^n\boldsymbol{B}_1$ which is

$${}^n\boldsymbol{B}_1 \quad \longleftrightarrow \quad {}^n\boldsymbol{b}_1 = \cos^n\left(\frac{\pi}{2}k\right)\,, \tag{A.3}$$

where k is the normalized wavenumber with the following relation to the sampling frequency

$$f = k\frac{f_s}{2}\,. \tag{A.4}$$

The relation to the subsampling can be determined by inserting Eq. A.4 and the cut-off frequency in Eq. A.3

$${}^n\boldsymbol{b}_{1,c} = \cos^n\left(\pi\,\frac{f_c}{f_s}\right) = \cos^n\left(\frac{\pi}{2\,p_{ss}}\right)\,. \tag{A.5}$$

Solving the above equation for n we yield

$$n = \frac{\log\left(\boldsymbol{b}_{1,c}\right)}{\log\left(\cos\left(\frac{\pi}{2\,p_{ss}}\right)\right)}\,. \tag{A.6}$$

In general, 3 dB are used as cut-off attenuation ($\boldsymbol{b}_{1,c} = 0.5$). Finally, the next larger even integer number $\hat{n}$ should be used to fulfill the symmetry condition of the filter, which is required to preserve the pixel-correspondences between the pyramid levels

$$\hat{n} = \lceil n \rceil_{even}\,. \tag{A.7}$$

The transfer function of a 2D Binomial filter can be easily derived by the one dimensional one (see [231]) and yields

$${}^{n}\boldsymbol{B}_2 \quad \longleftrightarrow \quad {}^{n}\boldsymbol{b}_2 = {}^{n}\boldsymbol{b}_{1,x}\,{}^{n}\boldsymbol{b}_{1,y} = \cos^n\left(\frac{\pi}{2}k_x\right)\cos^n\left(\frac{\pi}{2}k_y\right)\,. \tag{A.8}$$

A Taylor series evaluation in cylinder coordinates (k,θ) shows that the resulting filter is isotropic up to the second order

$${}^{n}\boldsymbol{b}_2 \approx 1 - \frac{n}{8}\left(\pi k\right)^2 + \frac{2n^2 - n}{256}\left(\pi k\right)^4 - \frac{n\cos\left(4\theta\right)}{768}\left(\pi k\right)^4\,. \tag{A.9}$$

The fourth order term contains an isotropic and an anisotropic part. However, the influence of the anisotropic part gets reduced by increasing the filter order, because it increases only linearly as opposed to the quadratic growth of the isotropic term. However, frequencies which are diagonal to the image axes are sampled by less than the factor of $\sqrt{2}$, but, due to the larger sampling, they are also attenuated stronger, which guarantees aliasing suppression in all directions.

A.2 Boxplots

A box-and-whisker diagram or simply boxplot is a convenient way of graphically depicting a set of samples. Plotting only the median or mean of a set of samples does not reveal much information about its distribution. To compare the robustness of different algorithms, also the variance and the amount of outliers should be visualized. All this is provided by boxplots.

Boxplots depict following characteristics of a population which are illustrated in Fig. A.1:

- Bottom and top of the box are the 1^{st}- and 3^{rd}-quartile (25^{th} and 75^{th} percentile)
- The red line within the box is the median (50^{th} percentile)
- Points are interpreted as outliers and drawn as red plusses, if they are smaller or larger than 1.5 of the interquartile range (IQR=3-quartile minus 1-quartile) from the first or third quartile respectively. Hence, the lower and the upper limits are 1-quartile - 1.5 IQR and 3-quartile + 1.5 IQR. The default of 1.5 corresponds to approximately $\pm 2.7\sigma$ and 99.3% coverage if the data is normally distributed.
- The whiskers denote the adjacent value, which is the most extreme data value that is not identified as an outlier according to the above criteria.

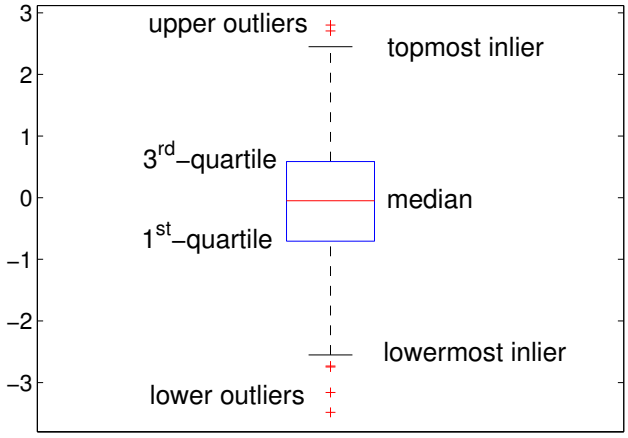


Figure A.1: A boxplot is a convenient way to illustrate the distribution characteristics of statistical population. The median, the 1^{st}- and 3^{rd}-quartile, the extend of the distribution and outliers are visualized by this representation. This plot has been generated by 1000 normal distributed random samples.

A.3 Image-Based Global Referencing

The localization introduced in Section 9.1 uses natural landmarks. However, based on local tracking it is not possible to set up a pose estimation with the same origin and coordinate frame again. Hence, a global reference is required. This can be achieved by using global feature trackers which allow to find natural landmarks even after larger affine distortion. The most popular examples are SIFT [120, 121] or SURF [122, 123]. The latter is more efficient providing similar performance and, thus, we use this in our implementation.

The SURF features can be initialized by structure-from-motion or by stereo. Experiments have shown that the accuracy is higher, if the feature depth is computed by stereo triangulation with a shorter baseline instead of a larger but error-prone baseline. One has to consider that the matching accuracy of SURF is also reduced if the affine transformation increases at large baselines.

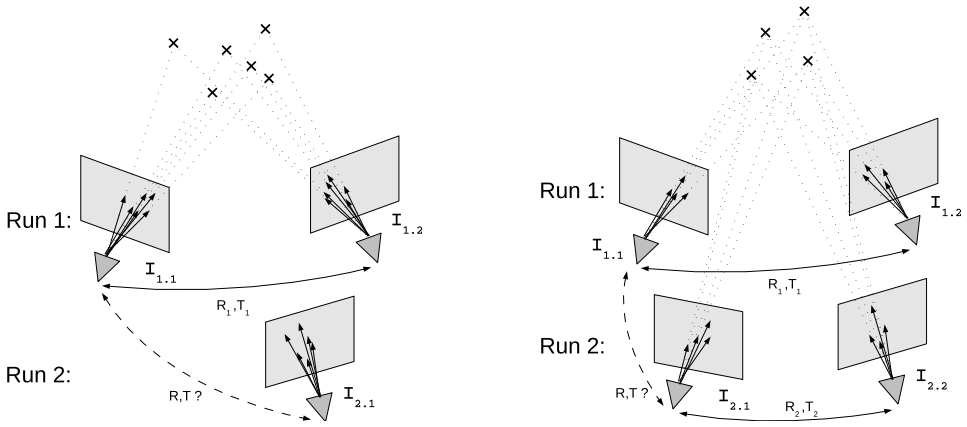
Further, it is possible to estimate the pose in 3D space or using RVPGS (see Section 5.2.1). The former method needs four images while the latter requires only three. Fig. A.2 depicts the principle of the algorithms. In the following we sketch the two methods:

The **3-images global referencing** approach consists of two steps: First, the 3D-structure of the SURF features of the image at the origin has to be initialized. Then, an arbitrary image with enough SURF correspondences can be spatially related to the reference landmark set using RVGPS.

On the other hand the **4-images global referencing** approach consists of three steps: Not only the 3D-structure of the SURF feature set at the origin has to be computed, but also of the landmarks which shall be spatially aligned. Then, one can simply align the two 3D point clouds using, *e.g.*, the Umeyama algorithm described in [214].

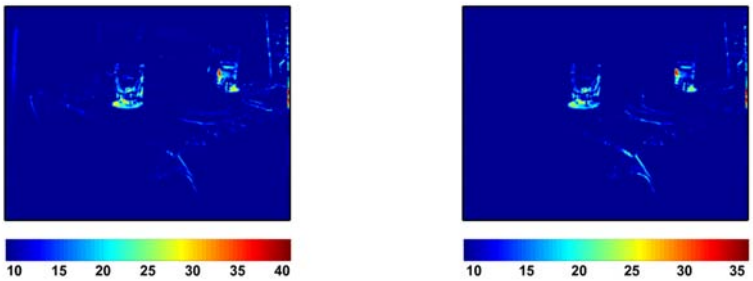
The 4-image-based approach is supposed to be more accurate, because we do not estimate the transformation matrix and the structure of the point set at the same time, like in the 3-images approach. The drawback is, that we need to find SURF matches in four images, which is more problematic than with three due to less intersection of common features. Which algorithm to use, therefore, strongly depends on the application and the scene. Since the errors do not vary much (compare the subfigures in Fig. A.3), the more robust but less accurate 3-images algorithm should be preferred in most cases.

One problem arises using an image-based global reference for 3D-modeling. A drawback of global feature trackers is that they, in general, are not real-time capable



(a) global referencing using three images (b) global referencing using four images

Figure A.2: These drawings sketch the principles of the 3- and the 4-images global referencing methods.



(a) (b)

Figure A.3: This pictures show the output of the Bayesian surprise trigger as described in Section 9.2, whereas the left image is generated by using the 3-images global referencing algorithm and the right one using the 4-images method. Even if the results for the 4-images approach seem to be more accurate, the one based on 3-images is preferable due to its improved robustness.

on conventional processors. Hence, three tasks run in parallel: feature tracking, feature initialization and global referencing. This requires a state machine to synchronize the different image processing modules, as it is sketched in Fig. A.4.

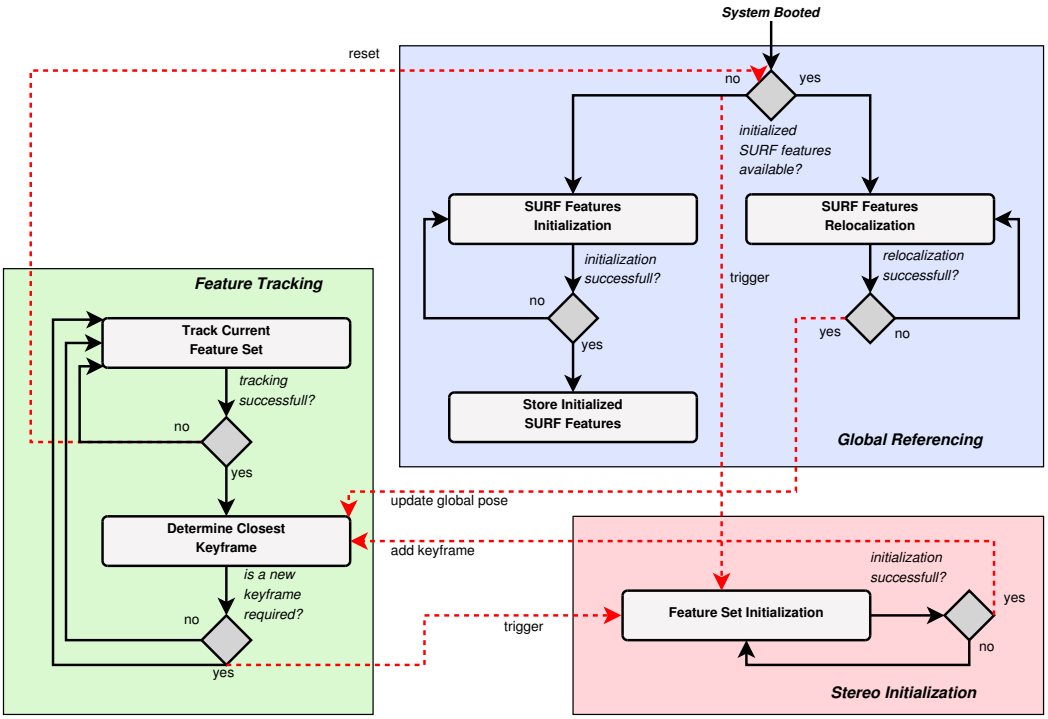


Figure A.4: The image-processing modules used for localization and global referencing. The synchronization triggers to ensure that the images of the different modules are synchronized are not illustrated for the sake of clarity.

A.4 Linear System Reduction

The skew symmetric matrices in Eq. 4.42 represent cross-products. By using a dot-product at this point and, thus, stacking all $\left({}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i}\right)^T$ row-wise, the size of the matrix for the SVD, $\boldsymbol{B}_T^{C+G}$, would be reduced to a third

$$\boldsymbol{U}_T \boldsymbol{\Sigma}_T \boldsymbol{V}_T^T = \text{SVD}\left(\boldsymbol{B}_T^{C+G}\right) . \tag{A.10}$$

In the following we show how the results of Eq. A.10 can be used to calculate the covariance of $\boldsymbol{P}_{{}^{G}\boldsymbol{p}'_C}$.

The right singular vectors of a matrix $\boldsymbol{B}$ correspond to the eigenvectors of $\boldsymbol{B}^T\boldsymbol{B}$ and the the singular values of $\boldsymbol{B}$ are the square root of the eigenvalues of $\boldsymbol{B}^T\boldsymbol{B}$. Let us compare the eigenvalues and eigenvectors of $\boldsymbol{B}_T^{C+G^T}\boldsymbol{B}_T^{C+G}$ and $\boldsymbol{B}_{[\cdot]_\times}^{C+G^T}\boldsymbol{B}_{[\cdot]_\times}^{C+G}$:

$$\boldsymbol{B}_T^{C+G^T}\boldsymbol{B}_T^{C+G} = \sum_{i=1}^{N} a_i a_i^T \tag{A.11}$$

$$\boldsymbol{B}_{[\cdot]_\times}^{C+G^T}\boldsymbol{B}_{[\cdot]_\times}^{C+G} = -\sum_{i=1}^{N} [a_i]_\times^2 = \sum_{i=1}^{N} a_i^T a_i \boldsymbol{I} - a_i a_i^T \tag{A.12}$$

with a_i being the single vector sums ${}^{C}\hat{\boldsymbol{p}}_{\Delta_i} + {}^{G}\hat{\boldsymbol{p}}_{\Delta_i}$. In the following $\boldsymbol{S} = \sum_{i=1}^{N} a_i a_i^T$ and $d = \sum_{i=1}^{N} \|a_i\|^2$. Eq. A.11 and A.12 can then be rewritten as

$$\boldsymbol{B}_T^{C+G^T}\boldsymbol{B}_T^{C+G} = \boldsymbol{S} \quad \text{and} \quad \boldsymbol{B}_{[\cdot]_\times}^{C+G^T}\boldsymbol{B}_{[\cdot]_\times}^{C+G} = d\boldsymbol{I} - \boldsymbol{S} \tag{A.13}$$

The eigenvalues can now be calculated by solving equation $\det(A - \lambda I) = 0$, which yields for the dot-product

$$\det(\boldsymbol{S} - \lambda_T I) = 0 \tag{A.14}$$

and for the cross-product

$$\det\left((dI - \boldsymbol{S}) - \lambda_{[\cdot]_\times} I\right) = 0 \tag{A.15}$$

$$\det\left(\boldsymbol{S} - \left(d - \lambda_{[\cdot]_\times}\right) I\right) = 0\,, \tag{A.16}$$

resulting in the relations

$$\lambda_T = d - \lambda_{[\,\cdot\,]_\times} \qquad \text{and} \tag{A.17}$$

$$\sigma_T = \sqrt{d - \sigma^2_{[\,\cdot\,]_\times}} = \sqrt{\sum_{i=1}^{3} \sigma^2_{[\,\cdot\,]_\times} - \sigma^2_{[\,\cdot\,]_\times}} \tag{A.18}$$

for the singular values σ. This corresponds to the sine and cosine relation between the magnitudes of the cross- and dot-product

$$\|a_i\|\sin(\varphi) = \sqrt{\|a_i\|^2 - \|a_i\|^2\cos^2(\varphi)} \tag{A.19}$$

and leads to

$$\Sigma_{[\,\cdot\,]_\times} = \boldsymbol{I}_M \sqrt{\operatorname{tr}\left(\boldsymbol{\Sigma}_T^2\right) \boldsymbol{I} - \boldsymbol{\Sigma}_T^2}\, \boldsymbol{I}_M \tag{A.20}$$

with the mirrored identity matrix $\boldsymbol{I}_M$ which switches the rows and columns to preserve the descending order of the singular values

$$\boldsymbol{I}_M = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

The eigenvectors $\boldsymbol{v}_T$ and $\boldsymbol{v}_{[\,\cdot\,]_\times}$ and, hence, V_T and $V_{[\,\cdot\,]_\times}$ of $\boldsymbol{B}_T^{C+G^T}\boldsymbol{B}_T^{C+G}$ and $\boldsymbol{B}_{[\,\cdot\,]_\times}^{C+G^T}\boldsymbol{B}_{[\,\cdot\,]_\times}^{C+G}$ are equal, which can be shown by applying Eq. A.17:

$$\boldsymbol{S}\boldsymbol{v}_T = \lambda_T \boldsymbol{v}_T \tag{A.21}$$

$$\begin{aligned} (d\boldsymbol{I} - \boldsymbol{S})\, \boldsymbol{v}_{[\,\cdot\,]_\times} &= \lambda_{[\,\cdot\,]_\times} \boldsymbol{v}_{[\,\cdot\,]_\times} \\ d\boldsymbol{v}_{[\,\cdot\,]_\times} - \boldsymbol{S}\boldsymbol{v}_{[\,\cdot\,]_\times} &= d\boldsymbol{v}_{[\,\cdot\,]_\times} - \lambda_T \boldsymbol{v}_{[\,\cdot\,]_\times} \\ \boldsymbol{S}\boldsymbol{v}_{[\,\cdot\,]_\times} &= \lambda_T \boldsymbol{v}_{[\,\cdot\,]_\times}\,. \end{aligned} \tag{A.22}$$

However, the SVD may yield different signs for the eigenvectors, but since we multiply $\boldsymbol{V}_T$ with its transposed to compute the covariance matrix $\boldsymbol{P}_{G_{\boldsymbol{p}'_C}}$, the signs of the eigenvectors become insignificant in our application in Section 4.2.4. Further, the SVD yields the singular values in descending order and, hence, the columns of the Vs and the rows of the Σs will be inverted. Combining the matrices to compute $\boldsymbol{P}_{G_{\boldsymbol{p}'_C}}$, one will see that the permutations are neutralized, such that

$$\boldsymbol{P}_{G_{\boldsymbol{p}'_C}} = \boldsymbol{V}_T \left(\operatorname{tr}\left(\boldsymbol{\Sigma}_T^2\right) \boldsymbol{I} - \boldsymbol{\Sigma}_T^2\right)^{\frac{1}{2}} \boldsymbol{V}_T^T\,. \tag{A.23}$$

A.5 Electronically Synchronized IMU-Camera Setup

We have seen in Section 8.5.1, that it is crucial to provide a proper temporal alignment of the sensors, especially if high dynamic motions are applied. In general, IMUs do not allow to trigger their measurements, thus, the synchronization concepts are not trivial anymore.

In a student project, we tried to realize an electronically synchronized embedded sensor-system. As embedded processing platform we chose the Beagleboard. To allow for a fast and DMA (*direct memory access*) capable image transmission, without blocking the USB-bus, we designed a frame-grabber which is based on the MMC (*multi media card*) bus. The major components of the frame-grabber are an FPGA, a video converter and two fast SRAM units. For more details please refer to [232].

The vertical synchronization signal of the video codec is used to trigger a self-made IMU (consisting of 3 gyroscopes and 3 accelerometers). An on-board RISC processor (ARM) pre-processes and stores the high-rate measurements of the sensors. Each time a trigger arrives, the samples are sent as bundle to the Beagleboard where the image and the corresponding IMU information can be fused [233]. The Linux kernel on the Beagleboard is enhanced by respective modules which allow for a regular communication with the new devices [234].

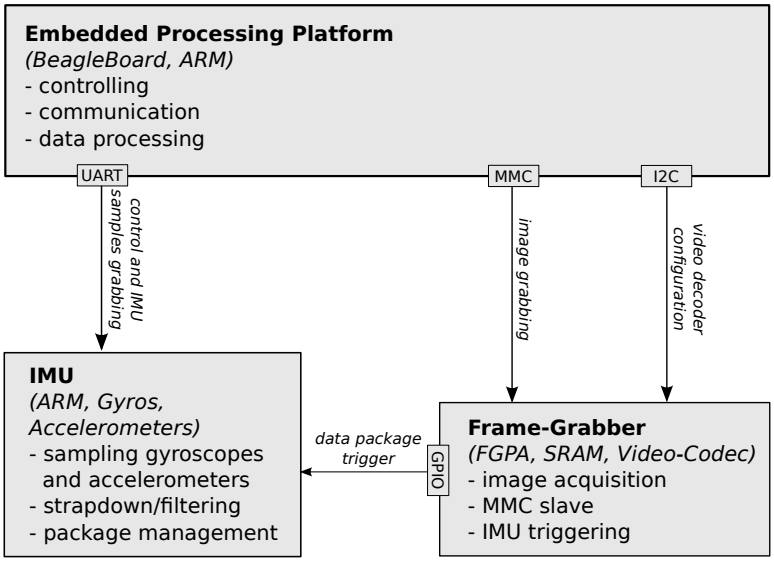


Figure A.5: This graphic illustrates the components of the hardware-synchronized IMU-camera setup. The IMU generates image specific inertial measurement bundles, triggered by the vertical synchronization signal of the camera.

Fig. A.5 sketches a block diagram for the hardware modules and communication

interfaces of the synchronized IMU camera setup and Fig. A.6 shows the first iteration of the system's prototype.

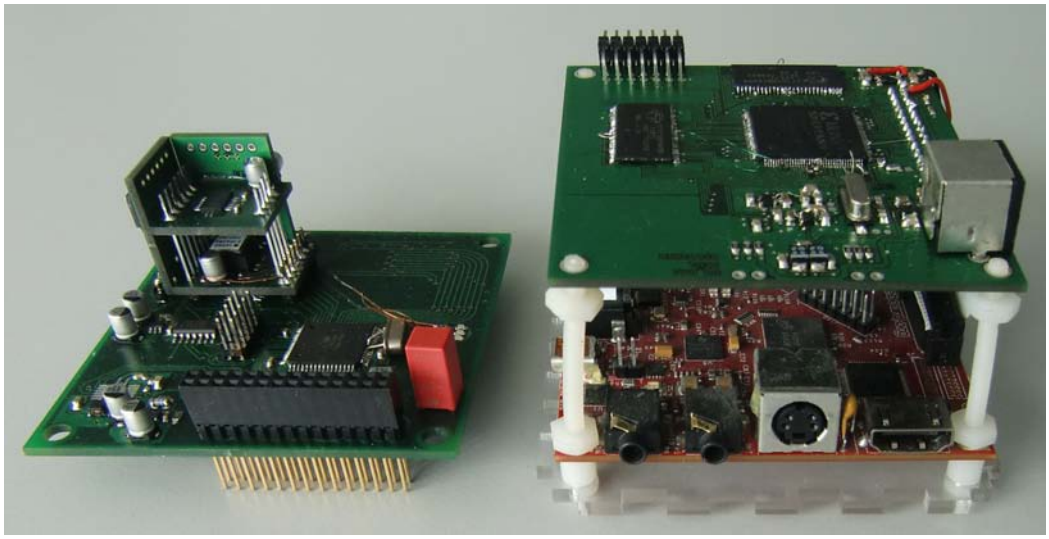


Figure A.6: This picture shows the first iteration of a prototypic hardware-synchronized IMU-camera setup. The processing platform is the Beagleboard (red bottom board on the left hand side). The MMC-based frame-grabber is on the top of it and the IMU-board, which would be the top layer of the staple, is put aside for the sake of visibility.

Mathematical Notation

Table A.2: Explanation of the mathematical notation.

Symbols / Examples	Description
n	Scalar variables are italic lower case letters.
N	Scalar constants are italic upper case letters.
$\boldsymbol{v}$	Vectors are bold, italic lower case letters.
$\boldsymbol{M}$	Matrices are bold, italic upper case letters. This notation is also used for points in 3D space.
$\mathrm{f}(x)$, $\cos(x)$, $\mathrm{R}(x)$	Functions are one or more regular lower case letters or upper case letters, depending on whether scalars or matrices are returned.
$\mathcal{S}$, $\mathcal{W}$	Sets and coordinate frames are represented by upper case calligraphic literals.
$\lvert\mathcal{S}\rvert$, $\lvert x\rvert$	Represents the number of elements in the set $\mathcal{S}$ or the absolute value of a scalar x.
$\{\,\cdot\,\}_{i=A}^{B}$	Denotes a set which contains all elements for $i \in [A \dots B] \subset \mathbb{Z}$.
$\boldsymbol{M}_{2,3}$, $\boldsymbol{M}_{-,3}$	Elements of matrices are accessed by index-tuples as following subscripts in the order row then column, whereas a dash represents the whole row or column respectively. Counting starts at one.
$\boldsymbol{M}_{3\times4}$	The dimensions of a matrix are described as subscript, *e.g.* a matrix with 3 rows and 4 columns would be written as shown left.

Continued on next page

Table A.2 – continued from previous page

Symbols / Examples	Description
$\boldsymbol{I}_N$	The N-dimensional identity matrix is denoted as shown left.
$\mathbf{0}_N$ and $\mathbf{1}_N$	These are column vectors of 0 or 1, respectively. The N may also be spared, if the size of the vector is clear from the context.
$\mathbf{0}_{R\times C}$ and $\mathbf{1}_{R\times C}$	These are matrices of 0 or 1 respectively, where R denotes the number of rows and C the number of columns.
$\mathrm{tr}(\boldsymbol{M})$	The trace of a matrix $\boldsymbol{M}$ is expressed as shown left.
$\mathrm{diag}(x)$	This function is used twofold, if it is applied to a vector it transforms it to a diagonal matrix and if it is applied to a matrix it extracts the diagonal elements of the matrix as column vector.
$\mathrm{Diag}(x)$	This function transforms a matrix into a block-diagonal one.
$\otimes$	This operator denotes the Kronecker product.
$\circ$	This symbol is used for the Hadamard product (entry-wise matrix multiplication).
$\cong$	Linear approximations are expressed by this symbol.
$\overset{def}{=}$	If a new variable is introduced at the end of an equation it is expressed by this symbol, so that the reader is not surprised by an unknown variable.
$\overset{!}{=}$	To emphasize that the equation is valid by definition, we use this representation.
$\mathrm{E}[x]$	This function computes the expectation value of a random variable x.
x^-	A minus as successive superscript denotes the prior estimate, *e.g.*, before applying the update step of a Kalman filter.
x^+	A plus as successive superscript denotes the posterior estimate, *e.g.*, after applying the update step of a Kalman filter.

Continued on next page

Table A.2 – continued from previous page

Symbols / Examples	Description
$[\,\cdot\,]_{\text{col}}$, $[\,\cdot\,]_{\text{row}}$	For some derivations it will be necessary to reshape a matrix to a single column or row vector. For that we introduce these operators which stack all the columns or rows on top or aside of each other, respectively, *e.g.*, $\boldsymbol{a} = [\boldsymbol{A}_{3\times3}]_{\text{col}} = \begin{pmatrix} \boldsymbol{A_{1,1}} \\ \boldsymbol{A_{2,1}} \\ \boldsymbol{A_{3,1}} \\ \boldsymbol{A_{1,2}} \\ \vdots \\ \boldsymbol{A_{3,3}} \end{pmatrix}$.
$\text{col}\,[\,\cdot\,]_{i=1}^{N}$, $\text{row}\,[\,\cdot\,]_{i=1}^{N}$	If indices are required for the expression to stack, it may also be denoted as shown left.
$\underset{\text{row}\ (1,3,2,4)}{\text{perm}}$	This function permutes the rows of a matrix according to the indices in the subscript.
$\|\,\cdot\,\|$	If no matrix norm is specified the Euclidean norm $\|\,\cdot\,\|_2$ is assumed.
$\|\,\cdot\,\|_F$	This operator represents the Frobenius norm.
$\text{rk}(\boldsymbol{M})$	This function returns the rank of a matrix $\boldsymbol{M}$.
$\text{SVD}(\boldsymbol{M})$	This function returns three matrices as the result of the singular value decomposition of the matrix $\boldsymbol{M}$ ($\boldsymbol{M} = \boldsymbol{U\Sigma V}$). The singular values are in decreasing order.
$\text{EVD}(\boldsymbol{M})$	This function computes the eigenvalue decomposition of the matrix $\boldsymbol{M}$ and returns two matrices($\boldsymbol{M} = \boldsymbol{Q\Lambda Q}^{-1}$). The eigenvalues are in decreasing order.
$[\,\cdot\,]_{\times}$	This operator denotes the mapping from a vector into a skew-symmetric matrix which replaces the cross product: $[\,\cdot\,]_{\times} : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$.

Continued on next page

Table A.2 – continued from previous page

Symbols / Examples	Description
SO	This abbreviation denotes the special orthogonal group (space of rotation matrices).
$\mathcal{F}(\omega)$	This function transforms the signal into the frequency domain.
$\longleftrightarrow$	This sign denotes the same signal in time and frequency domain.
$\arg(c)$	This function computes the argument of a complex number c, which corresponds to its phase.
${}^{I}x$	A leading superscript denotes the frame of reference in which a measure is expressed.
x^{C}	A successive superscript denotes the sensor with which a measure has been gathered.
x_i, x_t	Successive subscripts are in general indices or time references.
$x_{A;i}$	If more subscripts are necessary to specify a variable, they are separated by a semicolon.
${}^{B}\boldsymbol{R}_{A}$	This expression denotes a rotation from the coordinate frame $\mathcal{A}$ to $\mathcal{B}$ represented by a DCM.
${}^{C}\boldsymbol{t}_{AB}$	This term represents the translation from coordinate frame $\mathcal{A}$ to $\mathcal{B}$ expressed in the coordinate frame $\mathcal{C}$.
DCM(...)	This function converts an arbitrary rotation representation into a DCM (see [111]).
Q(...)	This function converts an arbitrary rotation representation into a quaternion (see [111]).
$\bar{\boldsymbol{q}}$	The bar over a quaternion denotes the inverse rotation.
med($\mathcal{S}$)	Computes the median of the set $\mathcal{S}$.
$\odot$	This operator denotes the quaternion multiplication.
$\prod_{i=1\odot}^{N}$	This symbol represents the quaternion product operator which computes the product of successive rotations.

List of Abbreviations

Table A.3: List of abbreviations used within this work.

Abbreviation	Description
AGAST	**a**daptive and **g**eneric **AST**
AM	**a**ctive **m**atching
AST	**a**ccelerated **s**egment **t**est
ARM	**a**dvanced **R**ISC **c**omputer
BFGS	**B**royden-**F**letcher-**G**oldfarb-**S**hanno (method)
BRIEF	**b**inary **r**obust **i**ndependent **e**lementary **f**eatures
BRISK	**b**inary **r**obust **i**nvariant **s**calable **k**eypoints
CenSurE	**c**enter **s**urround **e**xtrema
CoTeSys	**co**gnition for **te**chnical **sys**tems - elite cluster
CPU	**c**entral **p**rocessing **u**nit
DCM	**d**irection **c**osine **m**atrix (rotation matrix)
DMA	**d**irect **m**emory **a**ccess
DOF, DOFs	**d**egree(s) **o**f **f**reedom
DoG	**d**ifference **o**f **G**aussians
DRA	**d**orsal **r**im **a**rea
EKF	**e**xtended **K**alman **f**ilter
FAST	**f**eatures from **AST**
FOC	**f**ocus **o**f **c**ontraction

Continued on next page

Table A.3 – continued from previous page

Abbreviation	Description
FOE	**f**ocus **o**f **e**xpansion
FOV	**f**ield **o**f **v**iew
FPGA	**f**ield-**p**rogrammable **g**ate **a**rrays
GPS	**g**lobal **p**ositioning **s**ystem
GPU	**g**raphics **p**rocessing **u**nit
IMU	**i**nertial **m**easurement **u**nit
IR	**i**nfra**r**ed
IQR	**i**nter**q**uartile **r**ange
KLD	**K**ullback-**L**eibler **d**ivergence
KLT	**K**anade-**L**ucas-**T**omasi (feature tracker)
LADAR	**l**aser **d**etection **a**nd **r**anging
LIDAR	**l**ight **d**etection **a**nd **r**anging
LOS	**l**ine-**o**f-**s**ight (coordinate system)
LRS	**l**aser **r**ange **s**canner
LSP	**l**aser **s**tripe **p**rofiler
MEMS	**m**icro-**e**lectro-**m**chanical **s**ystems
ML	**m**aximum **l**ikelihood
MMC	**m**ulti **m**edia **c**ard
MRID	**m**inimal **r**otation-**i**nvariant **d**escriptor
NMSM	**n**on-**m**aximum **s**uppression **m**easure
OAST	**o**ptimal **AST**
PTAM	**p**arallel **t**racking **a**nd **m**apping
RGB	**r**ed-**g**ree-**b**lue (color space)
RISC	**r**educed **i**nstraction **s**et
ROI	**r**egion **o**f **i**nterest
RVGPS	**r**obustified **VGPS**

Continued on next page

Table A.3 – continued from previous page

Abbreviation	Description
S/N	**s**ignal to **n**oise ratio
SIFT	**s**cale-**i**nvariant **f**eature **t**ransform
SLAM	**s**imultaneous **l**ocalization **a**nd **m**apping
SRAM	**s**ynchronous **r**andom-**a**ccess **m**emory
SSD	**s**um of **s**quared **d**ifferences
SURF	**s**peeded **u**p **r**obust **f**eatures
SUSAN	**s**mallest **USAN**
TOF	**t**ime **o**f **f**light
UKF	**u**nscented **K**alman **f**ilter
USAN	**u**ni-value **s**egment **a**ssimilating **n**ucleus
USB	**u**niversal **s**erial **b**us
VEM	**v**isual **e**nvironment **m**odeling
VGPS	**v**isual **GPS**

References

[15] G.K. Taylor and H.G. Krapp. **Sensory systems and flight stability: What do insects measure and why?** *Advances in insect physiology*, **34**:231–316, 2007.

[16] R.J. Greenspan. *Invertebrate neurobiology*. Number 49. Cold Spring Harbor Laboratory Pr, 2007.

[17] E. Warrant and D.E. Nilsson. *Invertebrate vision*. Cambridge Univ Pr, 2006.

[18] R. Wehner, R.D. Harkness, and P. Schmid-Hempel. *Foraging strategies in individually searching ants, Cataglyphis bicolor (Hymenoptera: Formicidae)*, **1**. G. Fischer, 1983.

[19] Y. Camlitepe and D.J. Stradling. **Wood ants orient to magnetic fields**. *Proceedings: Biological Sciences*, **261**(1360):37–41, 1995.

[20] T. Ritz, D.H. Dommer, and J.B. Phillips. **Shedding light on vertebrate magnetoreception**. *Neuron*, **34**(4):503–506, 2002.

[21] K.J. Lohmann, C.M.F. Lohmann, L.M. Ehrhart, D.A. Bagley, and T. Swing. **Geomagnetic map used in sea-turtle navigation**. *Nature: International Weekly Journal of Science*, 2004.

[22] K.J. Lohmann, C.M.F. Lohmann, and N.F. Putman. **Magnetic maps in animals: nature's GPS**. *Journal of Experimental Biology*, **210**(21):3697, 2007.

[23] P. Graham and K. Cheng. **Ants use the panoramic skyline as a visual cue during navigation**. *Current Biology*, **19**(20):R935–R937, 2009.

[24] S.F. Reid, A. Narendra, J.M. Hemmi, and J. Zeil. **Polarised skylight and the landmark panorama provide night-active bull ants with compass information during route following**. *Journal of Experimental Biology*, **214**(3):363, 2011.

[25] K. Fent and R. Wehner. **Oceili: A Celestial Compass in the Desert Ant Cataglyphis**. *Science*, **228**(4696):192, 1985.

[26] R. Wehner and M. Müller. **The significance of direct sunlight and polarized skylight in the ant's celestial system of navigation**. *Proceedings of the National Academy of Sciences*, **103**(33):12575, 2006.

[27] T. Labhart and E.P. Meyer. **Detectors for polarized skylight in insects: a survey of ommatidial specializations in the dorsal rim area of the compound eye**. *Microscopy research and technique*, **47**(6):368–379, 1999.

[28] SB Laughlin and GA Horridge. **Angular sensitivity of the retinula cells of dark-adapted worker bee**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **74**(3):329–335, 1971.

[29] R. Seidl and W. Kaiser. **Visual field size, binocular domain and the ommatidial array of the compound eyes in worker honey bees**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **143**(1):17–26, 1981.

[30] W. Stürzl, N. Boeddeker, L. Dittmar, and M. Egelhaaf. **Mimicking honeybee eyes with a 280° field of view catadioptric imaging system**. *Bioinspiration & Biomimetics*, **5**:036002, 2010.

[31] W. Livingston. **Landscape as viewed in the 320-nm ultraviolet**. *JOSA*, **73**(12):1653–1657, 1983.

[32] R. Moller. **Insects could exploit UV-green contrast for landmark navigation**. *Journal of theoretical biology*, **214**(4):619–631, 2002.

[33] L.G. Bishop and D.G. Keehn. **Neural correlates of the optomotor response in the fly**. *Biological Cybernetics*, **3**(6):288–295, 1967.

[34] M.V. Srinivasan and RL Gregory. **How Bees Exploit Optic Flow: Behavioural Experiments and Neural Models [and Discussion]**. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, **337**(1281):253–259, 1992.

[35] J. Zeil, R. Sandeman, and D. Sandeman. **Tactile localisation: the function of active antennal movements in the crayfishCherax destructor**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **157**(5):607–617, 1985.

[36] D.C. Sandeman. **Angular acceleration, compensatory head movements and the halteres of flies (Lucilia serricata)**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **136**(4):361–367, 1980.

[37] MA Gibbs and DPM Northmore. **The role of torus longitudinalis in equilibrium orientation measured with the dorsal light reflex**. *Brain, Behavior and Evolution*, **48**(3):115–120, 1996.

[38] E. Balkovsky and B.I. Shraiman. **Olfactory search at high Reynolds number**. *Proceedings of the National Academy of Sciences of the United States of America*, **99**(20):12589, 2002.

[39] H.G. Wallraff and M.O. Andreae. **Spatial gradients in ratios of atmospheric trace gases: a study stimulated by experiments on bird navigation**. *Tellus B*, **52**(4):1138–1157, 2000.

[40] G. Wallraff et al. **Zur olfaktorischen Navigation der Vögel**. *Journal für Ornithologie*, **144**(1):1–32, 2003.

[41] H.G. Wallraff. **Avian olfactory navigation: its empirical foundation and conceptual state**. *Animal Behaviour*, **67**(2):189–204, 2004.

[42] A. Stumpner and D. Von Helversen. **Evolution and function of auditory systems in insects**. *Naturwissenschaften*, **88**(4):159–170, 2001.

[43] A. Michelsen. **Biophysics of sound localization in insects**. *Springer Handbook of Auditory Research*, **10**:18–62, 1998.

[44] A. Moiseff, G.S. Pollack, and R.R. Hoy. **Steering responses of flying crickets to sound and ultrasound: mate attraction and predator avoidance**. *Proceedings of the National Academy of Sciences*, **75**(8):4052, 1978.

[45] M.E. Jensen, C.F. Moss, and A. Surlykke. **Echolocating bats can use acoustic landmarks for spatial orientation**. *Journal of experimental biology*, **208**(23):4399, 2005.

[46] S. ROSSEL. **Binocular spatial localization in the praying mantis**. *Journal of experimental biology*, **120**(1):265–281, 1986.

[47] J.M. Hemmi and J. Zeil. **Burrow surveillance in fiddler crabs. I. Description of behaviour**. *Journal of experimental biology*, **206**(22):3935–3950, 2003.

[48] J.M. Hemmi and J. Zeil. **Burrow surveillance in fiddler crabs. II. The sensory cues.** *The Journal of experimental biology*, **206**(Pt 22):3951, 2003.

[49] S. Sommer and R. Wehner. **The ant's estimation of distance travelled: experiments with desert ants, Cataglyphis fortis**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **190**(1):1–6, 2004.

[50] M. Müller and R. Wehner. **The hidden spiral: systematic search and path integration in desert ants, Cataglyphis fortis**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **175**(5):525–530, 1994.

[51] F. Papi, editor. *Animal Homing*, chapter Arthropods, pages 44–144. Chapman and Hall, 1992.

[52] G. Portelli, F. Ruffier, and N. Franceschini. **Honeybees change their height to restore their optic flow**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **196**(4):307–313, 2010.

[53] A.J. Rutkowski, M.M. Miller, R.D. Quinn, and M.A. Willis. **Egomotion estimation with optic flow and air velocity sensors**. *Biological Cybernetics*, pages 1–17, 2011.

[54] J. Zeil, M.I. Hofmann, and J.S. Chahl. **Catchment areas of panoramic snapshots in outdoor scenes**. *JOSA A*, **20**(3):450–469, 2003.

[55] R. Möller and A. Vardy. **Local visual homing by matched-filter descent in image distances**. *Biological cybernetics*, **95**(5):413–430, 2006.

[56] F. Labrosse. **Short and long-range visual navigation using warped panoramic images**. *Robotics and Autonomous Systems*, **55**(9):675–684, 2007.

[57] J. Zeil, A. Kelber, and R. Voss. **Structure and function of learning flights in bees and wasps**. *J Exp Biol*, **199**:245–252, 1996.

[58] T.S. Collett, E. Dillmann, A. Giger, and R. Wehner. **Visual landmarks and route following in desert ants**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **170**(4):435–442, 1992.

[59] R. Möller. **Do insects use templates or parameters for landmark navigation?** *Journal of Theoretical Biology*, **210**(1):33–45, 2001.

[60] T.S. Collett, P. Graham, R.A. Harris, and N. Hempel-de Ibarra. **Navigational memories in ants and bees: memory retrieval when selecting and following routes**. *Advances in the Study of Behavior*, **36**:123–172, 2006.

[61] J. Zeil, N. Boeddeker, and W. Stürzl. **Visual homing in insects and robots**. *Flying insects and robots*, pages 87–100, 2009.

[62] P. Graham, K. Fauria, and T.S. Collett. **The influence of beacon-aiming on the routes of wood ants**. *Journal of experimental biology*, **206**(3):535–541, 2003.

[63] P. Graham, V. Durier, and T.S. Collett. **The binding and recall of snapshot memories in wood ants (Formica rufa L.)**. *Journal of experimental biology*, **207**(3):393–398, 2004.

[64] W. Junger. **Waterstriders (Gerris paludum F.) compensate for drift with a discontinuously working visual position servo**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **169**(5):633–639, 1991.

[65] P. Graham and T.S. Collett. **View-based navigation in insects: how wood ants (Formica rufa L.) look at and are guided by extended landmarks**. *Journal of experimental Biology*, **205**(16):2499–2509, 2002.

[66] N. Boeddeker and J.M. Hemmi. **Visual gaze control during peering flight manoeuvres in honeybees**. *Proceedings of the Royal Society B: Biological Sciences*, **277**(1685):1209, 2010.

[67] N. Boeddeker, L. Dittmar, W. Stürzl, and M. Egelhaaf. **The fine structure of honeybee head and body yaw movements in a homing task**. *Proceedings of the Royal Society B: Biological Sciences*, **277**(1689):1899, 2010.

[68] L. Dittmar, W. Stürzl, E. Baird, N. Boeddeker, and M. Egelhaaf. **Goal seeking in honeybees: matching of optic flow snapshots?** *Journal of Experimental Biology*, **213**(17):2913, 2010.

[69] N. Hempel de Ibarra, M. Giurfa, and M. Vorobyev. **Discrimination of coloured patterns by honeybees through chromatic and achromatic cues**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **188**(7):503–512, 2002.

[70] M. Lehrer. **Dorsoventral asymmetry of colour discrimination in bees**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **184**(2):195–206, 1999.

[71] R. Menzel and E. Lieke. **Antagonistic color effects in spatial vision of honeybees**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **151**(4):441–448, 1983.

[72] MV Srinivasan, SW Zhang, and K. Witney. **Visual discrimination of pattern orientation by honeybees: Performance and implications forcortical'processing**. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, **343**(1304):199, 1994.

[73] G.A. Horridge. **Spatial coincidence of cues in visual learning by the honeybee (Apis mellifera)**. *Journal of Insect Physiology*, **44**(3-4):343–350, 1998.

[74] V. Durier, P. Graham, and T.S. Collett. **Snapshot memories and landmark guidance in wood ants**. *Current biology*, **13**(18):1614–1618, 2003.

[75] T.S. Collett, K. Fauria, K. Dale, and J. Baron. **Places and patterns–a study of context learning in honeybees**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **181**(4):343–353, 1997.

[76] M. Collett, T.S. Collett, S. Bisch, and R. Wehner. **Local and global vectors in desert ant navigation**. *Nature*, pages 269–271, 1998.

[77] M. Collett, TS Collett, S. Chameron, and R. Wehner. **Do familiar landmarks reset the global path integration system of desert ants?** *Journal of experimental biology*, **206**(5):877–882, 2003.

[78] M. Knaden and R. Wehner. **Nest mark orientation in desert ants Cataglyphis: what does it do to the path integrator?** *Animal behaviour*, **70**(6):1349–1354, 2005.

[79] A. Narendra, K. Cheng, D. Sulikowski, and R. Wehner. **Search strategies of ants in landmark-rich habitats**. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **194**(11):929–938, 2008.

[80] R. Wehner, M. Boyer, F. Loertscher, S. Sommer, and U. Menzi. **Ant navigation: one-way routes rather than maps**. *Current Biology*, **16**(1):75–79, 2006.

[81] M. Collett and T.S. Collett. **Insect navigation: no map at the end of the trail?** *Current biology*, **16**(2):R48–R51, 2006.

[82] K. Geiger. *Long-Distance Orientation in Honeybees*. PhD thesis, Free University of Berlin, 1994.

[83] S. van der Zwaan and J. Santos-Victor. **An insect inspired visual sensor for the autonomous navigation of a mobile robot**. *Proc. of the Seventh International Sysposium on Intelligent Robotic Systems (SIRS)*, 1999.

[84] R. Hornsey, P. Thomas, W. Wong, S. Pepic, K. Yip, and R. Krishnasamy. **Electronic compound eye image sensor: construction and calibration**. In *Proc. SPIE*, **5301**, pages 13–24, 2004.

[85] A.A. Stocker. **Analog integrated 2-D optical flow sensor**. *Analog Integrated Circuits and Signal Processing*, **46**(2):121–138, 2006.

[86] W. Reichardt. **Autocorrelation, a principle for the evaluation of sensory information by the central nervous system**. *Sensory communication*, pages 303–317, 1961.

[87] J. Díaz, E. Ros, F. Pelayo, E.M. Ortigosa, and S. Mota. **FPGA-based real-time optical-flow system**. *Circuits and Systems for Video Technology, IEEE Transactions on*, **16**(2):274–279, 2006.

[88] Z. Wei, D.J. Lee, and B.E. Nelson. **FPGA-based real-time optical flow algorithm design and implementation**. *Journal of Multimedia*, **2**(5):38–45, 2007.

[89] T. Zhang, H. Wu, A. Borst, K. Kuhnlenz, and M. Buss. **An fpga implementation of insect-inspired motion detector for high-speed vision systems**. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 335–340. IEEE, 2008.

[90] F. Ruffier and N. Franceschini. **OCTAVE, a bioinspired visuo-motor control system for the guidance of Micro-Air-Vehicles**. In *Proceedings of SPIE*, **5119**, pages 1–12, 2003.

[91] W. Stürzl, M. Suppa, and D. Burschka. **Light-weight panoramic mirror design for visual navigation**. *Omnidirectional Robot Vision*, pages 218–229, 2008.

[92] D.J. Heeger and A.D. Jepson. **Subspace methods for recovering rigid motion I: Algorithm and implementation**. *International Journal of Computer Vision*, **7**(2):95–117, 1992.

[93] H. Dahmen, RM Wüst, and J. Zeil. **Extracting egomotion parameters from optic flow: principal limits for animals and machines**. *From Living Eyes to Seeing Machines*, pages 174–98, 1997.

[94] A. Albu-Schäffer, C. Ott, U. Hagn, and T. Ortmaier. **Method for controlling a robot arm, and robot for implementing the method**, January 12 2010. US Patent 7,646,161.

[95] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Stemmer, A. Albu-Schäffer, O. Eiberger, A. Beyer, G. Grunwald, and G. Hirzinger. **Aus der Forschung zum Industrieprodukt: Die Entwicklung des KUKA Leichtbauroboters**. *at-Automatisierungstechnik*, **58**(12):670–680, 2010.

[96] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger. **The DLR lightweight robot: design and control concepts for robots in human environments**. *Industrial Robot: An International Journal*, **34**(5):376–385, 2007.

[97] J. Ulmen and M. Cutkosky. **A robust, low-cost and low-noise artificial skin for human-friendly robots**. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4836–4841. IEEE, 2010.

[98] A. Schmitz, M. Maggiali, L. Natale, and G. Metta. **Touch sensors for humanoid hands**. In *RO-MAN, 2010 IEEE*, pages 691–697. IEEE, 2010.

[99] G. Kowadlo and R.A. Russell. **Robot odor localization: a taxonomy and survey**. *The International Journal of Robotics Research*, **27**(8):869, 2008.

[100] M.A. Willis. **Chemical Plume Tracking Behavior in Animals and Mobile Robots**. *Navigation*, **55**(2), 2008.

[101] M.M. Hunt, W.M. Marquet, D.A. Moller, K.R. Peal, W.K. Smith, and R.C. Spindel. **An acoustic navigation system**. Technical report, Woods Hole Oceanographic Institution, 1974.

[102] S.T. Birchfield. **A unifying framework for acoustic localization**. In *Proceedings of the 12th European Signal Processing Conference (EUSIPCO)*, pages 1127–1130. Citeseer, 2004.

[103] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool. **Omnidirectional vision based topological navigation**. *International Journal of Computer Vision*, **74**(3):219–236, 2007.

[104] Y. Ma. *An invitation to 3-d vision: from images to geometric models*, **26**. Springer Verlag, 2004.

[105] J.J. Craig. *Introduction to robotics: mechanics and control*. Addison-Wesley New York, 3 edition, 2006.

[106] R.Y. Tsai and R.K. Lenz. **A new technique for fully autonomous and efficient 3D robotics hand/eye calibration**. *Robotics and Automation, IEEE Transactions on*, **5**(3):345–358, 1989.

[107] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision*, **201**. Prentice Hall New Jersey, 1998.

[108] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[109] K.H. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. **DLR CalLab and DLR CalDe - http://www.robotic.dlr.de/callab/**.

[110] 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, 1981.

[111] J. Wendel. *Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation*. Oldenbourg Wissenschaftsverlag, 2007.

[112] R.E. Kalman. **A new approach to linear filtering and prediction problems**. *Journal of basic Engineering*, **82**(Series D):35–45, 1960.

[113] M.S. Grewal and A.P. Andrews. *Kalman filtering: theory and practice using MATLAB*, **2**. Wiley Online Library, 2001.

[114] P. Kovesi. **Image features from phase congruency**. *Videre: Journal of Computer Vision Research*, **1**(3):1–26, 1999.

[115] P. Kovesi. **Phase congruency detects corners and edges**. In *The Australian Pattern Recognition Society Conference*, pages 309–318. Citeseer, 2003.

[116] T. Tuytelaars and K. Mikolajczyk. **Local invariant feature detectors: a survey**. *Foundations and Trends® in Computer Graphics and Vision*, **3**(3):177–280, 2008.

[117] C. Harris and M. Stephens. **A combined corner and edge detector**. In *Alvey vision conference*, **15**, page 50. Manchester, UK, 1988.

[118] J. Shi and C. Tomasi. **Good features to track**. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1993.

[119] J.A. Noble. *Descriptions of image surfaces*. University of Oxford, 1989.

[120] D.G. Lowe. **Object recognition from local scale-invariant features**. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, **2**, pages 1150–1157. Ieee, 1999.

[121] D.G. Lowe. **Distinctive image features from scale-invariant keypoints**. *International journal of computer vision*, **60**(2):91–110, 2004.

[122] H. Bay, T. Tuytelaars, and L. Van Gool. **Surf: Speeded up robust features**. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[123] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. **Speeded-up robust features (SURF)**. *Computer Vision and Image Understanding*, **110**(3):346–359, 2008.

[124] M. Agrawal, K. Konolige, and M. Blas. **Censure: Center surround extremas for realtime feature detection and matching**. *Computer Vision–ECCV 2008*, pages 102–115, 2008.

[125] S.M. Smith and J.M. Brady. **SUSAN–A new approach to low level image processing**. *International journal of computer vision*, **23**(1):45–78, 1997.

[126] E. Rosten and T. Drummond. **Fusing points and lines for high performance tracking**. *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[127] E. Rosten, R. Porter, and T. Drummond. **Faster and better: a machine learning approach to corner detection**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **32**(1):105–119, 2010.

[128] E. Rosten and T. Drummond. **Machine learning for high-speed corner detection**. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[129] J.R. Quinlan. **Induction of decision trees**. *Machine learning*, **1**(1):81–106, 1986.

[130] B.K.P. Horn and B.G. Schunck. **Determining optical flow**. *Artificial intelligence*, **17**(1-3):185–203, 1981.

[131] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. **An improved algorithm for TV-L 1 optical flow**. *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45, 2009.

[132] S. Leutenegger, M. Chli, and R.Y. Siegwart. **BRISK: Binary robust invariant scalable keypoints**. In *Proc. IEEE Int. Conf. on Computer Vision*, 2011.

[133] B.D. Lucas, T. Kanade, et al. **An iterative image registration technique with an application to stereo vision**. In *International joint conference on artificial intelligence*, **3**, pages 674–679. Citeseer, 1981.

[134] G.D. Hager and P.N. Belhumeur. **Real-time tracking of image regions with changes in geometry and illumination**. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 403–410. IEEE, 1996.

[135] G. Klein and D. Murray. **Parallel tracking and mapping for small AR workspaces**. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

[136] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. **Brief: Binary robust independent elementary features**. *Computer Vision–ECCV 2010*, pages 778–792, 2010.

[137] E. Tola, V. Lepetit, and P. Fua. **Daisy: An efficient dense descriptor applied to wide-baseline stereo**. *IEEE transactions on pattern analysis and machine intelligence*, pages 815–830, 2009.

[138] T. Ojala, M. Pietikäinen, and T. Mäenpää. **Multiresolution gray-scale and rotation invariant texture classification with local binary patterns**. *IEEE Transactions on pattern analysis and machine intelligence*, pages 971–987, 2002.

[139] S. Birchfield. **Derivation of kanade-lucas-tomasi tracking equation**. *Unpublished*, 1997.

[140] J.Y. Bouguet et al. **Pyramidal implementation of the lucas kanade feature tracker description of the algorithm**. *Intel Corporation, Microprocessor Research Labs, OpenCV Documents*, **3**, 1999.

[141] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. **MonoSLAM: Real-time single camera SLAM**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1052–1067, 2007.

[142] M.V. Srinivasan. **An image-interpolation technique for the computation of optic flow and egomotion**. *Biological Cybernetics*, **71**(5):401–415, 1994.

[143] JJ Koenderink and A.J. Doorn. **Facts on optic flow**. *Biological Cybernetics*, **56**(4):247–254, 1987.

[144] HC Longuet-Higgins. **A computer algorithm for reconstructing a scene from two projections**. *Readings in computer vision: issues, problems, principles, and paradigms*, page 61, 1987.

[145] P.H.S. Torr and D.W. Murray. **Outlier detection and motion segmentation**. *Sensor Fusion VI*, **2059**:432–443, 1993.

[146] L. Quan. **Invariants of six points and projective reconstruction from three uncalibrated images**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **17**(1):34–46, 1995.

[147] D. Nistér. **An efficient solution to the five-point relative pose problem**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **26**(6):756–770, 2004.

[148] S. Šegvić, G. Schweighofer, and A. Pinz. **Performance evaluation of the five-point relative pose with emphasis on planar scenes**. Technical report, EMT, TU Graz, 2009.

[149] R.I. Hartley. **In defense of the eight-point algorithm**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **19**(6):580–593, 1997.

[150] J. Weng, T.S. Huang, and N. Ahuja. **Motion and structure from two perspective views: Algorithms, error analysis, and error estimation**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **11**(5):451–476, 1989.

[151] G. Strang. **Linear algebra and its applications, 1988**. *Hartcourt Brace Jovanovich College Publishers*, 1988.

[152] M. Mühlich and R. Mester. **The role of total least squares in motion analysis**. *Computer Vision-ECCV*, page 305, 1998.

[153] G.H. Golub, A. Hoffman, and G.W. Stewart. **A generalization of the Eckart-Young-Mirsky matrix approximation theorem**. *Linear Algebra and Its Applications*, **88**:317–327, 1987.

[154] J.W. Demmel. **The smallest perturbation of a submatrix which lowers the rank and constrained total least squares problems**. *SIAM Journal on Numerical Analysis*, **24**(1):199–206, 1987.

[155] D. Burschka and G.D. Hager. **V-GPS-image-based control for 3d guidance systems**. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, **2**, pages 1789–1795. IEEE, 2003.

[156] D. Burschka and G.D. Hager. **V-GPS (SLAM): Vision-based inertial system for mobile robots**. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, **1**, pages 409–415. IEEE, 2004.

[157] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour. **Closed-form solution of absolute orientation using orthonormal matrices**. *JOSA A*, **5**(7):1127–1135, 1988.

[158] G. Klein and D. Murray. **Improving the agility of keyframe-based SLAM**. *Computer Vision–ECCV 2008*, pages 802–815, 2008.

[159] R.O. Castle, G. Klein, and D.W. Murray. **Wide-area Augmented Reality using Camera Tracking and Mapping in Multiple Regions**. *Computer Vision and Image Understanding*, **In Press,**(6):854–867, 2011.

[160] H. Strasdat, JMM Montiel, and A.J. Davison. **Real-time monocular SLAM: Why filter?** In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE, 2010.

[161] R.A. Newcombe and A.J. Davison. **Live dense reconstruction with a single moving camera**. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE, 2010.

[162] M. Chli and A. Davison. **Active matching**. *Computer Vision–ECCV 2008*, pages 72–85, 2008.

[163] M. Chli and A.J. Davison. **Active Matching for visual tracking**. *Robotics and Autonomous Systems*, **57**(12):1173–1187, 2009.

[164] B. Simons. **An overview of clock synchronization**. *Fault-Tolerant Distributed Computing*, pages 84–96, 1990.

[165] F. Sivrikaya and B. Yener. **Time synchronization in sensor networks: a survey**. *Network, IEEE*, **18**(4):45–50, 2004.

[166] S. Bruder, M. Farooq, and M. Bayoumi. **Robotic heterogeneous multisensor fusion with spatial and temporal alignment**. In *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pages 506–511. IEEE, 1991.

[167] T. Lei and CA Stelios. **Decentralized filtering with random sampling and delay**. *Information Sciences*, **81**(1-2):117–131, 1994.

[168] T.D. Larsen, N.A. Andersen, O. Ravn, and N.K. Poulsen. **Incorporation of time delayed measurements in a discrete-time Kalman filter**. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, **4**, pages 3972–3977. IEEE, 1998.

[169] W. Li and H. Leung. **Simultaneous registration and fusion of multiple dissimilar sensors for cooperative driving**. *Intelligent Transportation Systems*, 2004.

[170] S.J. Julier and J.K. Uhlmann. **Fusion of time delayed measurements with uncertain time delays**. In *American Control Conference, 2005. Proceedings of the 2005*, pages 4028–4033. IEEE, 2005.

[171] F. Tungadi and L. Kleeman. **Time Synchronisation and Calibration of Odometry and Range Sensors for High-Speed Mobile Robot Mapping**. In *ACRA'08*, 2008.

[172] J. Kelly and G.S. Sukhatme. **A General Framework for Temporal Calibration of Multiple Proprioceptive and Exteroceptive Sensors**. In *IFRR ISER'10*, 2010.

[173] R.Y. Tsai and R.K. Lenz. **Real time versatile robotics hand/eye calibration using 3d machine vision**. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 554–561. IEEE, 1988.

[174] Y.C. Shiu and S. Ahmad. **Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB**. *Robotics and Automation, IEEE Transactions on*, **5**(1):16–29, 1989.

[175] J.C.K. Chou and M. Kamel. **Finding the position and orientation of a sensor on a robot manipulator using quaternions**. *The international journal of robotics research*, **10**(3):240, 1991.

[176] F.C. Park and B.J. Martin. **Robot sensor calibration: solving AX=XB on the Euclidean group**. *Robotics and Automation, IEEE Transactions on*, **10**(5):717–721, 1994.

[177] R. Horaud and F. Dornaika. **Hand-eye calibration**. *The international journal of robotics research*, **14**(3):195, 1995.

[178] K. Daniilidis and E. Bayro-Corrochano. **The dual quaternion approach to hand-eye calibration**. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, **1**, pages 318–322. IEEE, 1996.

[179] K. Daniilidis. **Hand-eye calibration using dual quaternions**. *The International Journal of Robotics Research*, **18**(3):286, 1999.

[180] K.H. Strobl and G. Hirzinger. **Optimal hand-eye calibration**. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4647–4653. IEEE, 2006.

[181] J. Lobo and J. Dias. **Relative Pose Calibration Between Visual and Inertial Sensors**. In *ICRA Workshop on Integration of Vision and Inertial Sensors - 2nd InerVis*. IEEE, 2005.

[182] J. Lobo and J. Dias. **Relative pose calibration between visual and inertial sensors**. *The International Journal of Robotics Research*, **26**(6):561, 2007.

[183] J. Kelly and G. Sukhatme. **Fast relative pose calibration for visual and inertial sensors**. In *Experimental Robotics*, pages 515–524. Springer, 2009.

[184] J. Kelly and G.S. Sukhatme. **Visual-inertial simultaneous localization, mapping and sensor-to-sensor self-calibration**. In *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*, pages 360–368. IEEE, 2009.

[185] F.M. Mirzaei and S.I. Roumeliotis. **A Kalman filter-based algorithm for IMU-camera calibration**. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2427–2434. IEEE, 2007.

[186] F.M. Mirzaei and S.I. Roumeliotis. **A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation**. *Robotics, IEEE Transactions on*, **24**(5):1143–1156, 2008.

[187] F.M. Mirzaei and S.I. Roumeliotis. **IMU-Camera Calibration: Bundle Adjustment Implementation**. Technical report, Department of Computer Science, University of Minnesota, 2007.

[188] J.D. Hol, T.B. Schön, and F. Gustafsson. **Modeling and Calibration of Inertial and Vision Sensors**. *The International Journal of Robotics Research*, **29**(2-3):231, 2010.

[189] P. Lang and A. Pinz. **Calibration of hybrid vision/inertial tracking systems**. In *Proceedings of the 2nd InerVis: Workshop on Integration of Vision and Inertial Sensors*. Citeseer, 2005.

[190] D. Strelow and S. Singh. **Optimal motion estimation from visual and inertial measurements**. In *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, pages 314–319. IEEE, 2002.

[191] L. Armesto, J. Tornero, and M. Vincze. **Fast ego-motion estimation with multi-rate fusion of inertial and vision**. *The International Journal of Robotics Research*, **26**(6):577, 2007.

[192] P. Gemeiner, P. Einramhof, and M. Vincze. **Simultaneous motion and structure estimation by fusion of inertial and vision data**. *The International Journal of Robotics Research*, **26**(6):591–605, 2007.

[193] D. Strelow and S. Singh. **Online motion estimation from image and inertial measurements**. In *Workshop on Integration of Vision and Inertial Sensors (INERVIS)*. Citeseer, 2003.

[194] A.I. Mourikis and S.I. Roumeliotis. **A multi-state constraint Kalman filter for vision-aided inertial navigation**. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3565–3572. IEEE, 2007.

[195] S. Weiss and R. Siegwart. **Real-time metric state estimation for modular vision-inertial systems**. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4531–4537. IEEE, 2011.

[196] A. Chilian, H. Hirschmüller, and M. Görner. **Multisensor data fusion for robust pose estimation of a six-legged walking robot**. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2497–2504. IEEE, 2011.

[197] P. Piniés, T. Lupton, S. Sukkarieh, and J.D. Tardós. **Inertial aiding of inverse depth SLAM using a monocular camera**. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2797–2802. IEEE, 2007.

[198] K. Konolige, M. Agrawal, and J. Sola. **Large-scale visual odometry for rough terrain**. *Robotics Research*, pages 201–212, 2011.

[199] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart. **Fusion of IMU and vision for absolute scale estimation in monocular SLAM**. *Journal of intelligent & robotic systems*, pages 1–13, 2010.

[200] S.I. Roumeliotis and J.W. Burdick. **Stochastic cloning: A generalized framework for processing relative state measurements**. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, **2**, pages 1788–1795. IEEE, 2002.

[201] X.R. Li and V.P. Jilkov. **A survey of maneuvering target tracking-Part III: Measurement models**. In *Proc. 2001 SPIE Conf. on Signal and Data Processing of Small Targets*, **4473**, pages 423–446, 2001.

[202] M.S. Schlosser and K. Kroschel. **Limits in tracking with extended Kalman filters**. *Aerospace and Electronic Systems, IEEE Transactions on*, **40**(4):1351–1359, 2004.

[203] T.L. Song, J.Y. Ahn, and C. Park. **Suboptimal filter design with pseudomeasurements for target tracking**. *Aerospace and Electronic Systems, IEEE Transactions on*, **24**(1):28–39, 1988.

[204] V.Y. Liepin'sh. **An algorithm for evaluation a discrete Fourier transform for incomplete data**. *Automatic control and computer sciences*, **30**(3):27–40, 1996.

[205] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*. Springer Verlag, 2002.

[206] C. De Boor. *A practical guide to splines*, **27**. Springer Verlag, 2001.

[207] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2000.

[208] A. Blake and A. Zisserman. **Visual reconstruction**. 1987.

[209] S.J. Julier and J.K. Uhlmann. **Unscented filtering and nonlinear estimation**. *Proceedings of the IEEE*, **92**(3):401–422, 2004.

[210] N. Trawny and S.I. Roumeliotis. **Indirect Kalman filter for 3D attitude estimation**. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, **2**, 2005.

[211] K. Schmid. **Kalman filter framework for multi-sensor fusion with measurement time consideration**. Technical report, German Aerospace Center (DLR), 2011. to be submitted.

[212] B.K.P. Horn. **Closed-form solution of absolute orientation using unit quaternions**. *JOSA A*, **4**(4):629–642, 1987.

[213] K.S. Arun, T.S. Huang, and S.D. Blostein. **Least-squares fitting of two 3-D point sets**. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700, 1987.

[214] S. Umeyama. **Least-squares estimation of transformation parameters between two point patterns**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 376–380, 1991.

[215] P.J. Huber. *Robust statistical procedures*. Number 68. Society for Industrial Mathematics, 1996.

[216] L. Dorst. **First order error propagation of the procrustes method for 3D attitude estimation**. *IEEE transactions on pattern analysis and machine intelligence*, pages 221–229, 2005.

[217] M.R. Garey and R.L. Graham. **Performance bounds on the splitting algorithm for binary testing**. *Acta Informatica*, **3**(4):347–355, 1974.

[218] L. Hyafil and R.L. Rivest. **Constructing optimal binary decision trees is NP-complete**. *Information Processing Letters*, **5**(1):15–17, 1976.

[219] MR Garey. **Optimal binary identification procedures**. *SIAM Journal on Applied Mathematics*, pages 173–186, 1972.

[220] D. Geman and B. Jedynak. **Model-based classification trees**. *Information Theory, IEEE Transactions on*, **47**(3):1075–1082, 2001.

[221] SS Kislitsyn. **On discrete search problems**. *Cybernetics and Systems Analysis*, **2**(4):52–57, 1966.

[222] M. Suppa, S. Kielhofer, J. Langwald, F. Hacker, K.H. Strobl, and G. Hirzinger. **The 3d-modeller: A multi-purpose vision platform**. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 781–787. IEEE, 2007.

[223] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P.R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, et al. **Rollin'Justin-Mobile platform with variable base**. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1597–1598. IEEE, 2009.

[224] F. Hacker, J. Dietrich, and G. Hirzinger. **A laser-triangulation based miniaturized 2-d range-scanner as integral part of a multisensory robot-gripper**. In *EOS Topical Meeting on Optoelectronic Distance/Displacement Measurements and Applications, Nantes France*, 1997.

[225] K.H. Strobl, W. Sepp, E. Wahl, T. Bodenmuller, M. Suppa, JF Seara, and G. Hirzinger. **The DLR multisensory hand-guided device: The laser stripe profiler**. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, **2**, pages 1927–1932. IEEE, 2004.

[226] H. Hirschmüller. **Accurate and efficient stereo processing by semi-global matching and mutual information**. *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[227] H. Hirschmüller. **Stereo processing by semiglobal matching and mutual information**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 328–341, 2008.

[228] F. Chen, G.M. Brown, and M. Song. **Overview of three-dimensional shape measurement using optical methods**. *Optical Engineering*, **39**:10, 2000.

[229] F. Blais. **Review of 20 years of range sensor development**. *Journal of Electronic Imaging*, **13**(1), 2004.

[230] T. Bodenmüller and G. Hirzinger. **Online surface reconstruction from unorganized 3D-points for the DLR hand-guided scanner system**. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 285–292. IEEE, 2004.

[231] B. Jähne. *Digitale Bildverarbeitung*. Springer Verlag, 2005.

[232] K. Werner. **Development of an FPGA-Based MMC Interface for Analog Cameras**. Term Paper – Advisors: Mair, E. and Burschka, D., 2011.

[233] F. Wilde. **Development of an IMU-Board, Efficient Strapdown Computation and Fusion of the Gravitation Vector**. Bachelor Thesis, ongoing – Advisors: Mair, E. and Burschka, D., 2011.

[234] J. Zaddach. **Linux Link-Up of an IMU and an MMC-Framegrabber on an Embedded Platform**. Interdisciplinary Project – Advisors: Mair, E. and Burschka, D., 2011.