Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

---

# Scalable Visual Navigation for Micro Aerial Vehicles using Geometric Prior Knowledge

---

**Andreas Wendel**

Graz, Austria, May 2013

*Thesis Supervisors*

Prof. Dr. Horst Bischof

Prof. Dr. Martial Hebert

If knowledge can create problems, it is not through ignorance that we can solve them.

---

*Isaac Asimov*

## STATUTORY DECLARATION

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

| | |
|---|---|
| (date) | (signature) |

# Abstract

Micro aerial vehicles (MAVs) are gaining importance as image acquisition tools in photogrammetry, but also for industrial applications such as power pylon inspection or construction site monitoring. Areas of interest for these tasks are often close to buildings, to the ground, and to other obstacles. Therefore, we see the need for systems which can navigate accurately and safely in an urban outdoor environment.

In this thesis, we present a scalable framework for visual navigation of micro aerial vehicles. The proposed algorithms facilitate autonomous flight in cluttered urban outdoor environments based on a monocular camera as the main exteroceptive sensor of the MAV. In contrast to systems which are based on monocular Simultaneous Localization and Mapping (SLAM), our approach incorporates the rich set of already available knowledge about geometric structures in the world to enhance localization quality and robustness.

Our first contribution is a toolbox of approaches for modeling geometric prior knowledge, including the reconstruction of 3D point clouds and lines, dense volumetric and mesh-based geometry, and 2.5D digital surface models. We further present quality measures which support users during the difficult task of acquiring all necessary images for reconstructing a scene.

Second, we contribute a multi-scale framework for visual localization and mapping which exploits the previously modeled geometric priors. We automatically align multiple 3D models in a global coordinate system and then partition the search space for localization using virtual views. Our method is not affected by drift and delivers positioning accuracies comparable to differential GPS. It achieves a constant localization framerate in completely scalable environments, and it supports the integration of in-flight information to improve the initial scene representation. As global localization is still too computationally heavy to run onboard micro aerial vehicles, we show how to deliberately distribute the computational load between the mobile device and a powerful server. Moreover, we present a distributed SLAM system capable of generating dense volumetric reconstructions in real-time, thus enabling fast collision avoidance and user interaction.

Finally, our third contribution is the application of computer vision methods to path planning and control. We show that digital surface models can be used as prior knowledge for high-level path and view planning, and we combine localization methods with fuzzy control techniques to build a full visual navigation system. Furthermore, we combine

state-of-the-art imitation learning techniques with visual input to demonstrate high-speed, autonomous MAV flight through a dense forest.

The proposed algorithms are quantitatively and qualitatively evaluated on a variety of datasets. Our experiments demonstrate that geometric prior knowledge helps to build a scalable, accurate, and computationally feasible visual navigation system for micro aerial vehicles.

**Keywords.** Computer vision, mobile robots, visual navigation, Micro Aerial Vehicle (MAV), geometric prior knowledge, 3D reconstruction, Structure from Motion (SfM), image-based localization, Simultaneous Localization and Mapping (SLAM), distributed system, vision-based control.

# Acknowledgments

This dissertation would have been impossible without the guidance and help of several people who in one way or another contributed their valuable time.

First of all, I would like to thank my supervisor Horst Bischof for supporting me throughout my studies. I am grateful for the confidence he has placed in me when founding the Aerial Vision Group, and I appreciate that he gave me the opportunity to develop my own ideas in a new field. I am very thankful to the former and current members of my group, Arnold Irschara, Michael Maurer, Christof Hoppe, Markus Rumpler, and Manuel Hofer, who I have closely worked with during the past years and who have significantly contributed to this thesis. I would also like to thank my Master students Thomas Kempter, Mario Katusic, Christian Mostegel, and Gert Hutter for contributing to our joint goals.

Next, I would like to express my gratitude to Martial Hebert for volunteering as my second thesis supervisor and for his efforts in hosting me at Carnegie Mellon University in 2012. My research visit has been incredibly inspiring and enlightening. I am deeply grateful for being treated as a member of the team by Martial as well as by my colleagues Debadeepta Dey, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, and Drew Bagnell.

Further, my sincere thanks go to all other colleagues I have worked with at ICG, but especially to Thomas Pock, Michael Donoser, Gottfried Graber, Gerhard Reitmayr, Axel Pinz, and Franz Leberl, as well as Martin Pfanner and Wernich de Villiers at Omicron electronics, for collaborating in research, teaching, and other projects.

Finally, I feel greatly indebted to my Mum Imelda, my brothers Klaus and Christian, and my family and friends. In particular, I would like to thank my fiancée Julia for her inspiration, love, and support. Despite the many days and nights that I have spent working on my research, she is always there when I need her and lends a helping hand. Thank you!

My research has been financially supported by the Austrian Research Promotion Agency (FFG) projects FIT-IT Pegasus (825841), Construct (830035), and Holistic (830044), as well as by a scholarship of the Austrian Marshall Plan Foundation.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

**Contents**



In the past decade Unmanned Aerial Vehicles (UAVs) have enjoyed considerable success in many civil and military applications such as search and rescue, monitoring, exploration, and mapping. Today, UAVs are deployed in many different shapes and sizes, ranging from 25 m wingspan and a payload of several tons in the case of the General Atomics Predator and Reaper series to a wingspan of 7.5 cm and negligible payload in the case of the DARPA Nano Air Vehicles program. The largest of these UAVs can fly at altitudes of 15000 meters for 30 hours, while the smallest can fly at a few meters altitude for only a couple of minutes. Additionally, the degree of autonomy varies considerably. While UAVs were historically remotely piloted, recently a trend towards autonomous control and navigation can be observed. This development is driven by increasingly powerful onboard computing possibilities as well as improved sensing technologies such as the global positioning system (GPS), radar, or optical sensors.

In this thesis we focus on Micro Aerial Vehicles (MAVs) with a mass of less than 5 kg and outdoor flight within line of sight. We discuss the design of a scalable visual navigation system which incorporates the rich set of already available knowledge about geometric structures in the world to enhance localization quality and robustness. The presented algorithms facilitate autonomous flight in cluttered urban outdoor environments based on a monocular camera as the main exteroceptive sensor of the MAV.

## 1.1 Micro Aerial Vehicle: An Emerging Robotic Platform

Micro aerial vehicles are gaining importance as image acquisition tools in photogrammetry [128] (Figure 1.1), but also for industrial applications such as power pylon inspection or construction site monitoring [54, 74, 117]. A variety of remotely piloted aerial vehicles is used for these applications, including airplanes as well as helicopters, quad-rotors, and octo-rotors. While fixed-wing vehicles offer the advantage of fast rectilinear flight to cover an extended area quickly and systematically, rotary aircraft provide the flexibility to explore a smaller area from multiple vantage points. In both cases, MAVs offer a significant advantage over traditional methods for image acquisition in terms of costs, as not only the vehicle itself is comparably cheap but also the deployment is fast and simple.

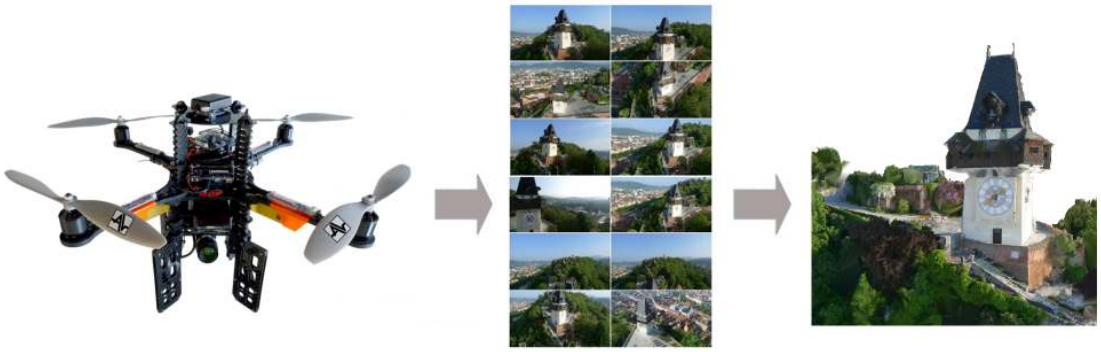


Figure 1.1: Reconstruction of the clocktower in Graz, Austria. An MAV allows to acquire imagery from new points of view which cannot be reached otherwise. The imagery can then be used for 3D reconstruction to visualize landmarks, for inspection of industrial structures, for photogrammetrical measurements, or as maps for autonomous navigation.

The usage of unmanned aerial vehicles for photogrammetric applications started as early as 1979 with the work of Przybilla and Wester-Ebbinghaus [163]. Their aircraft had a length of 3 m and a wing span of 2.6 m, and imagery was acquired from an altitude of 150 m above ground at a speed of 40 km/h. About 15 years later, Conway [30] presented a system for autonomous control of a small-scale helicopter based on an integrated GPS, which later enabled automated aerial image acquisition using UAVs [52]. With the advent of MAVs in form of very small and light aircraft or quad-rotor helicopters it became possible to fly much closer to the ground. Areas of interest are often in urban environments, close to buildings, and possibly close to human beings. While the new technology permits to obtain high–quality, close–up images of scenes, the new environment poses a problem for localization. GPS can be replaced by much more accurate differential GPS [104], if high costs are acceptable; still, both technologies are not able to detect obstacles and are prone to shadowing effects caused by neighboring buildings. Therefore, we see the need for intelligent autonomy in terms of systems which can navigate accurately and safely in an urban outdoor environment.

## 1.2 Visual Navigation: Localization, Mapping, and Control

Micro aerial vehicle navigation, or robot navigation in general, involves three key components which are highly dependent on each other. First, the robot needs to know where it is located in the world. Without proper *localization*, neither a map can be created nor a path planned. Second, to avoid obstacles the robot needs to know how the world in the vicinity of its location looks like, and for heading towards the intended target location it needs to know as much detail as possible about the rest of the world. Acquiring this knowledge and storing it in a consistent representation is called *mapping*. Due to the high interconnectivity, localization and mapping are often treated as a joint problem called Simultaneous Localization and Mapping (SLAM) [50]. Finally, given a proper localization and an accurate map, *path planning* and *control* approaches make the robot move towards the target location. However, actions might not be executed as planned, so the control strategy affects localization and mapping as well. A more detailed overview of a typical system architecture for navigation is given in Figure 1.2.

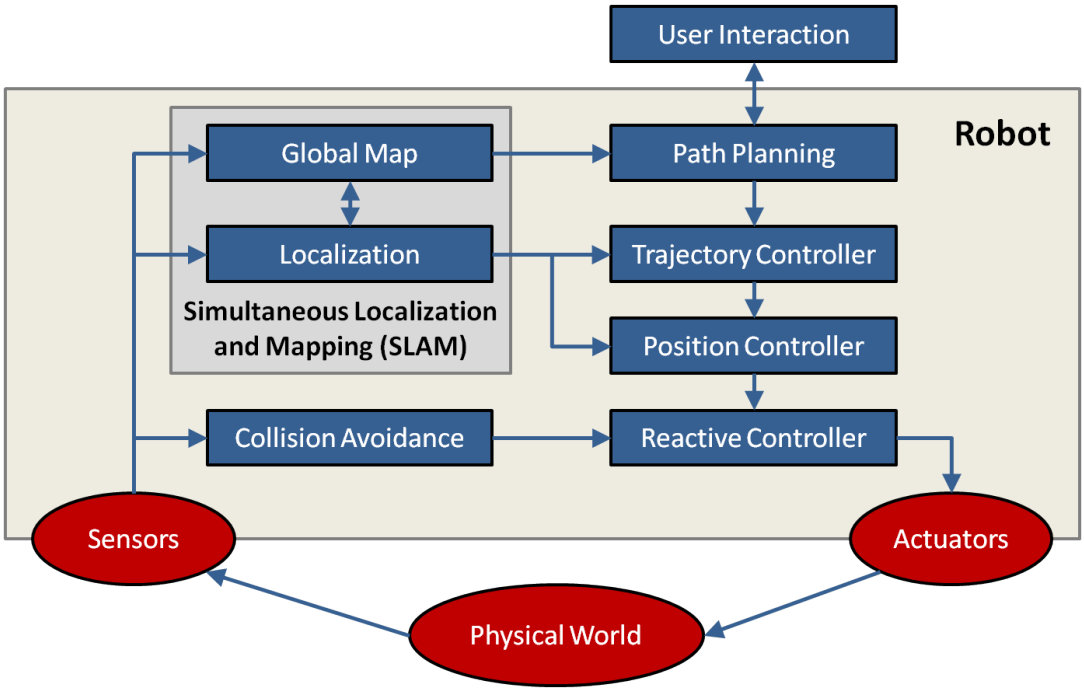



Figure 1.2: Typical system architecture for robot navigation. Maintaining a *global map* is important for defining a target where the robot should navigate to. In combination with high-level *path planning*, these two parts form the basis for *user interaction*. Once a goal is set, the robot needs to track its progress in reaching it with a *trajectory controller*. It thus needs to compare high-level commands with a sensor-based *localization* to come up with a suitable command for the *position controller*. Most systems also feature a *collision avoidance* component, which can be connected directly to a *reactive controller*.

For visual navigation of micro aerial vehicles, these key components need to be covered as well. However, the explicit implementation of the parts differs according to the sensors and actuators. For instance, when using laser range finders [8, 180] for localization and mapping, the distance between the sensor and the scene is directly obtained. Both sensor localization and measurement are noisy though, so a Bayesian probabilistic framework [200] is employed to keep track of the uncertainty. In contrast, monocular cameras as used in this thesis only provide a 2D projection of the scene at a given sensor position, and the 3D information needs to be extracted by matching and triangulating identical points in different views [80]. This process is called *Structure-from-Motion (SfM)*. When applied to image sequences, the technique is also known in robotics as *Incremental SfM* or *Monocular Visual SLAM*. Clearly, an even tighter interaction of the localization algorithms and the 2D/3D information in the map than for other sensing modalities is necessary.

For successful visual Simultaneous Localization and Mapping, the obtained images need to be processed in an online fashion. Recently, considerable progress has been made towards fast and robust monocular visual SLAM [51, 113], but tracking corresponding points in the image sequence and optimizing the map is computationally heavy. Moreover, when the map grows and an increasing amount of observations is added, solving the involved optimization problems gets infeasible. This behavior conflicts with typical applications of mobile visual localization and mapping in augmented reality and autonomous robotics, where the computational resources are often much lower than in a desktop setting, and the outdoor environments are much larger than what can be handled by current algorithms in reasonable time.

## 1.3 The Importance of Geometric Prior Knowledge

Simultaneous Localization and Mapping techniques have initially been developed for applications in completely unknown terrain. However, when navigating a robot or localizing mobile devices in urban outdoor environments, a large amount of knowledge about the geometry is readily available. Sources include mapping services such as Google Street View or Microsoft Bing Streetside, public geographic data sources such as the NASA Shuttle Radar Topography Mission [55] or local governmental providers, crowd-sourcing projects [68, 206], internet photo collections [69, 186, 187], imagery from aerial surveys [127], or simply images taken manually on ground level with a consumer camera. Generating 3D models from a set of images has become a widely studied field of research over the last few years, and good progress has been made to reconstruct and represent a variety of different environments.

In this thesis we focus on exploiting the multitude of geometric prior knowledge for improving visual navigation of mobile robots. More specifically, we observe that significant parts of urban scenes such as buildings do not alter their geometry for months, sometimes even years, and can therefore serve as natural landmarks. In our experience visual SLAM

can compete with a model-based method in the local environment it is initialized, but not on a larger scale. A direct experimental comparison such as the one depicted in Figure 1.3 shows that the visual SLAM approach misses to reconstruct large parts of the flight. As soon as the track is lost, the incremental pose update cannot be established anymore and localization fails. Especially in outdoor environments this issue occurs frequently, for instance when the MAV turns towards an open space where no textural features can be found, due to motion blur caused by fast rotations, or when direct sunlight hits the camera for a few frames. The repetitiveness of a scene causes further problems and inhibits successful relocalization within a larger space. We thus want to emphasize the importance of prior knowledge for vision-based navigation tasks and present scalable techniques for localization, mapping, and control.

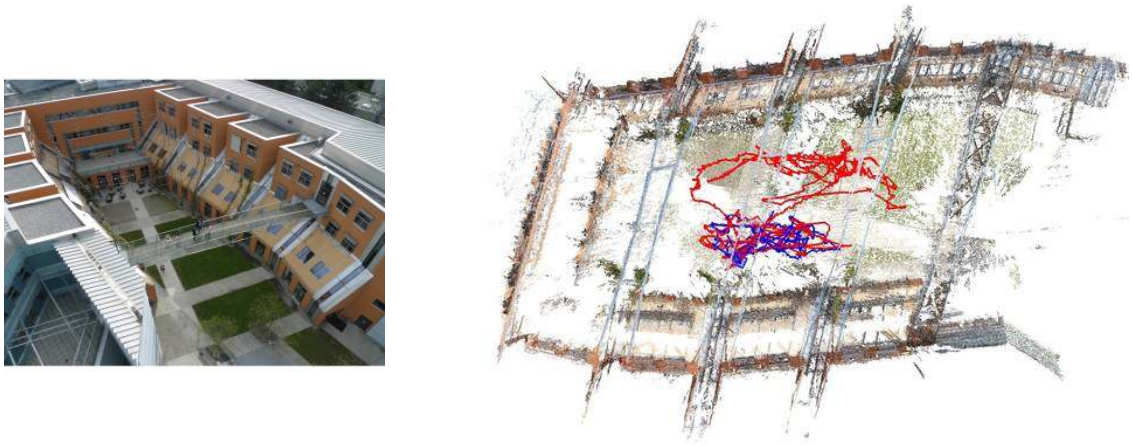


Figure 1.3: Comparison of flight trajectories in an atrium scene, overhead view. The overlay of the flight trajectories reconstructed using a visual SLAM approach (PTAM [113], blue) compared to a multi-scale, model-based method (red) as presented in this thesis shows that SLAM is not suitable for every outdoor environment. PTAM has lost track of its initial map at some point during the flight and is not able to recover. Note that this is an unfair comparison as the prior map is given in our case but needs to be obtained in case of PTAM; it demonstrates the benefits of incorporating geometric priors, though.

## 1.4 Contributions of the Thesis

Visual navigation for MAVs based on a monocular camera is a challenging task, and several problems have to be tackled to enable autonomous flight. In this thesis, we propose to exploit geometric prior knowledge to build a scalable, accurate, and computationally feasible visual navigation system. In the following, we summarize our key contributions towards this goal.

**Modeling Geometric Prior Knowledge.** Geometric prior knowledge can be represented in various forms, including 3D point clouds, 3D lines, dense volumetric or mesh-based geometry, and 2.5D digital surface models. We present a toolbox of techniques to accurately model the different priors for visual navigation. While these methods typically run offline, we also present a reconstruction approach which allows to receive online feedback about the expected quality and completeness of a reconstruction.

**Scalable Global Localization.** Monocular SLAM approaches have to create maps while localizing at the same time, which causes problems with scalability, drifting pose estimates over time, metrical scale estimation, and relocalization, as will be discussed in Section 2. In contrast, we incorporate prior knowledge and are thus able to present a multi-scale approach for monocular visual localization with an accuracy comparable to differential GPS. We first align geometric priors in a global coordinate system and then employ the concept of virtual views in 3D spaces to partition the search space. Our method is neither prone to drift nor bias, it achieves a constant localization framerate in large environments, and it supports the integration of in-flight information to improve the initial scene representation. Localization results generated by our algorithm are in a world coordinate system, which allows the MAV to incorporate additional sensors and to switch seamlessly between GPS and visual localization.

**Distributed SLAM and Live Dense Reconstruction.** Combined with a global localization approach, SLAM is very beneficial to handle partially or unknown parts of the scene, and to cope with changed conditions with respect to the prior. However, mobile devices such as micro aerial vehicles lack the necessary computational power to run monocular SLAM in real-time. We thus present a novel approach to visual SLAM for devices with low computational capabilities. Instead of reducing the map size and simplifying the optimization as in other approaches, we deliberately distribute the computational load between the mobile device and a powerful server. We further introduce the first system that is capable of generating dense volumetric reconstructions in real-time based on monocular input provided by a mobile platform. Creating dense models on-the-fly facilitates visual obstacle modeling for MAVs.

**Line-based Reconstruction and Tracking.** In man-made scenes, line features are often more prevalent than distinctive point features. We thus present a model-based tracking method which builds on visual SLAM but incorporates line-based priors for refinement and in case point-based localization fails. Motivated by the application of power pylon inspection using MAVs, we also present an approach for line-based 3D scene modeling without explicit appearance-based matching of the lines. Power pylons are wiry structures where the background is more dominant than the foreground, thus we align the cameras based on features in the background and exploit geometrical constraints to find suitable line segments. Both approaches work for wiry structures as well as solid objects.

**Vision for Path Planning and Control.** Navigation involves not only localization and mapping, but also path planning and control. We investigate how prior knowledge in form of digital surface models can be used for high-level path planning, as well as for view planning for optimized image acquisition. Moreover, we present a visual servoing system for MAVs based on a fuzzy logic controller which is designed to adhere the properties of visual pose estimates.

**Vision-based Reactive Control.** When encountering scenes where local features are not discriminative enough, such as a dense forest environment, it is necessary to rely on purely reactive control. We adapt a state-of-the-art imitation learning technique to train reactive heading policies based on the knowledge of a human pilot. Visual features extracted from the corresponding image are mapped to the control input provided by the expert. In contrast to straight-forward supervised learning, our policies are iteratively learned and exploit corrective input at later iterations to boost the overall performance of the predictor, especially in situations which would not be encountered by a human pilot.

## 1.5 Outline

This thesis is organized as follows. Chapter 2 introduces the state-of-the-art in visual localization and mapping, as well as in path planning and control. Further, a review of visual localization methods which incorporate prior knowledge is given. We then define four different kinds of geometric prior knowledge in Chapter 3, and we explain how the 3D model data can be obtained from imagery. We also discuss optimized strategies for image acquisition, as this task is often not straight-forward for laymen. In Chapter 4 we introduce our novel multi-scale approach to visual localization and mapping. The layers include global alignment of geometric priors, point and line-based localization within such priors, and a distributed system for incorporating real-time visual SLAM in this framework. While the thesis is focused on localization and mapping, we present several ideas and algorithms on visual input for path planning and control in Chapter 5. Again, we follow a multi-scale approach from global path planning in digital surface models to very local, high-framerate reactive control. Our framework is evaluated in Chapter 6. We show quantitative experiments for prior model accuracy and localization accuracy, as well as qualitative examples of successful flights with visual navigation. We conclude the thesis in Chapter 7 and provide an outlook to future work.

# Chapter 2

# Common Navigation Approaches for Micro Aerial Vehicles

**Contents**



An impressive body of research on navigation of micro aerial vehicles has been published within the last few years. Several state-of-the-art approaches for MAV control would be desirable for navigation in urban outdoor environments as they feature impressive aggressive maneuvers [137, 143] or can even be used for formation flight with large swarms of MAVs [122]. However, these methods still require the real-time, accurate state feedback delivered by a motion-capture system inside a laboratory and are therefore unsuitable for our purpose.

For being able to navigate in urban outdoor scenes, 3D perception directly on the robot is a necessity. Early work on autonomous navigation such as the well-known NAVLAB [199] system for road-following already incorporated 3D sensing for obstacle detection and terrain modeling. More recently, the successful implementation of large, complex robotic systems for autonomous driving during the DARPA Grand Challenge [202] and the DARPA Urban Challenge [208] has demonstrated the potential of current sensors and algorithms. Ever since a combination of multiple high-quality active and passive 3D sensors have been successfully applied, a major research focus is to simplify such systems and to reduce the costs [40].

In this thesis, we focus on low-cost but highly informative sensors for navigation: Passive, monocular cameras. Visual sensors deliver the richest representation of a scene, are light-weight, require little power, and can be used for indoor as well as for outdoor

applications. As a result, cameras are perfectly suited for the autonomous navigation of an MAV. A single visual sensor is enough to map large-scale environments and to estimate motion in 6 degrees of freedom with a higher precision than low-cost GPS sensors [221, 222]. The only two drawbacks are that visual input requires the most algorithmic intelligence and computational power to process the data, and that it might not work in scenes with very little textural features.

In the following, we will review the key components of a navigation system which have been introduced in the previous section. We start with an overview of Simultaneous Localization and Mapping approaches, then present related algorithms for localization in a known model, and finally give an introduction to path planning methods and control strategies.

## 2.1 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) addresses the problem of exploring a previously unknown environment. This requires to determine the sensor pose given an estimated map, and to optimize the map given the sensor information at the estimated pose; two tasks which highly depend on each other and thus have to be solved simultaneously.

The SLAM problem can be treated in a Bayesian framework where sensor poses and map points are connected by sensor measurements [49, 50, 201]. As these measurements are affected by noise, typically modeled as Gaussian distribution, a maximum likelihood solution to the pose and structure problem is desired. Early SLAM methods were based on successive measurements of geometric beacons [131], ultra-sonic scans [32], or laser range scans [136], and they updated probability distributions over map points and sensor poses accordingly. These approaches first established the group of *recursive filtering techniques*, which lead to the seminal work of Dissanayake et al. [44] on radar-based SLAM using a Kalman Filter, and the FastSLAM algorithm of Montemerlo et al. [146] for robot pose estimation with a Particle Filter.

While recursive filtering techniques are not restricted to certain sensors, the most popular sensors to carry on-board MAVs are laser range finders (LiDARs) and RGB-D sensors. Both deliver quite accurate depth estimates at a high framerate without any additional processing effort. He et al. [81] as well as Bachrach et al. [8] demonstrated using scanning LiDARs for SLAM in unknown indoor environments, and Bry et al. [24] showed how to use the same sensor for fast flight in indoor environments. The trend in indoor active sensing goes towards RGB-D sensors [9] that allow more detailed and faster scans. Real-time 3D reconstruction using a Microsoft Kinect system has already been demonstrated [152]. Unluckily, the structured light pattern projected by Kinect is only suitable for indoor usage and in general, all currently available RGB-D sensors suffer from very limited range in outdoor environments. Therefore, Vandapel et al. [210] proposed outdoor planning approaches in three dimensions for UAV navigation using lidar data, and Scherer et al. [180] achieved fast obstacle avoidance using a Yamaha RMax helicopter

and a 2-axis scanning LiDAR. For carrying outdoor LiDAR systems and the corresponding power supplies, larger and more expensive aerial vehicles than what we aim for are required. A suitable alternative is passive vision as presented in the next paragraph.

### 2.1.1 Visual SLAM

Accurate depth estimation and localization is also possible purely based on passive visual sensors, and several state-of-the-art approaches exist. An important technique is stereo imaging, which has been successfully used in real–time, outdoor SLAM systems [61, 142]. Stereo systems use two cameras with a fixed, known distance (*baseline*) between them. This has the essential benefit that the depth measurements are retrieved in metrical scale. When using only one camera, the relative transform between individual camera poses can be estimated as well, but the overall scale factor remains unobservable unless at least one metrical distance between poses or map points is known. However, on the one hand the baseline available to most micro aerial vehicles is quite small and thus there is little benefit for outdoor usage, and on the other hand the required processing power for stereo image acquisition and processing is typically higher than for monocular vision.

Early visual SLAM systems using a single camera as the main perceptual sensor were inspired by traditional SLAM methods using recursive filtering approaches. The Bayesian network allows handling of uncertainties in position and scale in a way that robust localization is possible. Davison [37] fused measurements from a sequence of images by updating probability distributions over features and extrinsic camera parameters using an Extended Kalman Filter (EKF). This requires tracking and mapping to be closely linked, as the camera pose and feature locations are updated together at every single frame. Based on this approach Davison et al. [38] proposed *MonoSLAM*. They extend the previously proposed method of monocular feature initialization by a feature orientation estimation. This successfully increased the range in which landmarks are detected and hence improved tracking. However, *MonoSLAM* only worked for slow camera motions or in combination with additional sensors.

In contrast to the incremental processing pipelines in robotics, the photogrammetry community focused on *batch optimization techniques* [203] to solve for extrinsic camera parameters and 3D points. The key idea is again to formulate the problem as a Bayesian network, but to minimize an image-based error. Additionally, obtaining the measurements is a more complicated process: One needs to match points in a given collection of unordered images, estimate pairwise relative camera configurations (again up to scale), find the position of the 3D points using triangulation, group the set of measurements and 3D points, and finally optimize the resulting Bayesian network by minimizing the distance between reprojected 3D points and the corresponding points in the image. As a result, the camera poses as well as the 3D points are obtained.

For a long time, batch optimization techniques were computationally too intensive for real-time operation. However, advances in computing hardware and a deliberate imple-

mentation of Klein and Murray [113, 114] made it possible to develop a real-time visual SLAM system based on keyframes named Parallel Tracking and Mapping. PTAM employs two separate threads, one for tracking the camera pose in every single frame, and another for mapping the environment by applying bundle adjustment [203] to a set of spatially distributed keyframes. Keyframes thus correspond to the collection of images used for bundle adjustment as mentioned before. The major drawback of filtering approaches, namely the limited number of features that can be updated in between two frames, is circumvented in PTAM by leaving more time to the parallel mapping process. To keep the computation time roughly constant even when the number of keyframes gets large, local bundle adjustment for a subset of frames is performed [149].

More recently, Newcombe et al. [153] introduced a system which is capable of dense tracking and mapping, meaning that their system does not rely on sparse feature tracking anymore. Their approach is highly sophisticated and delivers accurate depth maps based on all captured frames. However, it also requires a considerable amount of processing power to be available on-site, which in our case means that it would be needed on-board the micro aerial vehicle. In fact, all currently available live dense reconstruction systems exploit general-purpose graphics processing units (GPGPUs) to achieve real-time performance and are therefore not directly applicable on an MAV. The only exception is our own work [223] as presented in Section 4.3.

### 2.1.2 Visual SLAM for Devices with Low Computational Power

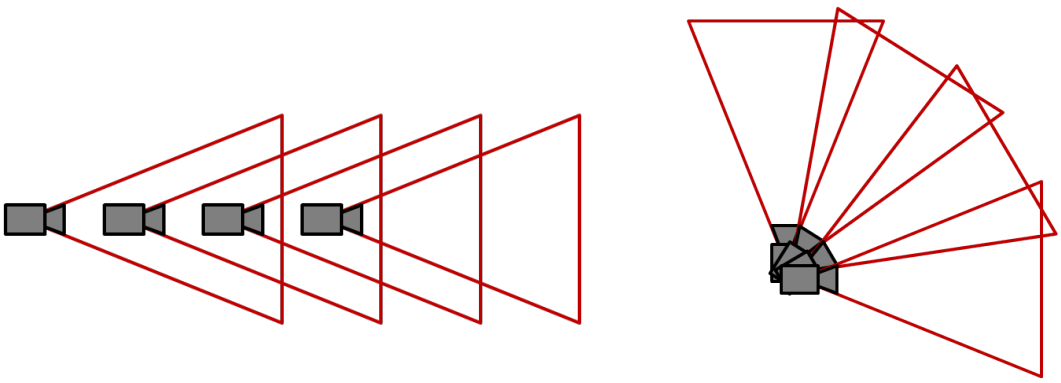
When the computational resources are less than in a desktop setting, only few suitable visual SLAM approaches remain, and the typical research areas are augmented reality and robotics. Klein and Murray [115] extended the original PTAM approach to work with mobile phone cameras. To compensate for the lack of processing power they had to introduce several simplifications: First, they require corners to be detected over multiple scales; this leads to less but more stable features. Second, measurements are thinned rather than densified as in original PTAM to limit the map size. As a result, the pose estimate is far less robust and a considerable amount of texture is required in the scene for the tracking to work at all. In robotics, the problem of visual navigation for MAVs in unstructured environments has been addressed only recently. PTAM has been used for providing position estimates for visual control in indoor and outdoor settings [2, 21, 218, 224], but again simplifications in tracking and bundle adjustment were necessary for the approach to run on a quad–rotor helicopter.

### 2.1.3 Challenges in Visual SLAM

Strasdat et al. [194] experimentally showed that monocular visual localization using a large amount of features measured at low temporal frequency is in general superior to a small amount of features measured at every frame, thus keyframe-based methods typically outperform filtering approaches. Still, in both cases the mapping process is affected by ac-

cumulated noise of sensor measurements, called *drift*, which leads to a significant difference between the pose estimate and the true pose after some time. When the SLAM approach is built on a monocular camera system where pose and point estimates are available only up to scale, the resulting maps are even less constrained than when using a sensor which directly delivers metric measurements, and thus drift is more likely to occur.

While small measurement uncertainties are in general unavoidable when tracking in the discretized image space, severe drift occurs for two major reasons. First, the camera motion might be degenerate (Figure 2.1). Pure forward motion leads to an ill-posed problem when triangulating 3D points, because the triangulation angle is very small and thus the position uncertainty is very large. The same problem can be observed when rotating around the camera's center as no baseline between individual views is created. Second, the mapped structures can be degenerate. Outdoor scenes typically contain lots of features, but they might not be very distinctive as in Figure 2.2(a). As soon as tracking is lost the incremental pose update cannot be established anymore, and localization as well as mapping fails. Especially in outdoor environments this issue occurs frequently, for instance when the MAV turns towards an open space where no textural features can be found, due to motion blur caused by fast rotations, or when direct sunlight hits the camera for a few frames. Also, occluders might suddenly cover the already created map (Figure 2.2(b)). While most SLAM systems feature recovery strategies based on image retrieval approaches [156], those still just work when an overlap to the obtained map is found. Even then, repetitive scenes as in Figure 2.2(c) often inhibit successful relocalization and can even cause misleading map updates.



(a) (b)

Figure 2.1: Degenerate motions in visual SLAM. (a) Pure forward motion leads to an ill-posed problem when triangulating 3D points because the triangulation angle is very small and thus the position uncertainty is very large. (b) The same problem occurs when rotating around the camera's center as no baseline between individual views is created. Especially fast rotations are critical when using tracking-based pose estimation.


(a) (b) (c)

Figure 2.2: Degenerate structures in visual SLAM. (a) Indiscriminative features, (b) loss of tracking due to sudden occluders, (c) and repetitive scenes are just some examples for situations which cause problems when creating visual maps online.

The drift problem can only be tackled to a certain degree on a small scale using measurements which are as accurate as possible. On a larger scale, a *loop closure* is applied to correct for the drift when revisiting previously mapped areas. Similar to relocalization, appearance-based methods [33] are used to detect already visited areas. Then, the loop consisting of camera poses and relative transforms in between is closed and the error is distributed over all relative constraints [51, 193]. While such an approach theoretically ensures consistent maps, in practice three issues need to be discussed. First, the loop optimization can of course only be performed if the robot returns to a previously visited 3D position with a similar 3D camera orientation, which is not necessarily the case in many MAV applications. Second, the graph of poses and relative constraints can get very large in outdoor environments. In this case, most current systems revert to approximate or simplified optimization problems [149] to guarantee computational feasibility. Finally, the localization can be significantly wrong before a loop closure, so a direct relation to global coordinates cannot be established. This prevents an MAV to switch seamlessly between GPS-based and visual navigation.

We thus want to emphasize the importance of prior knowledge for vision–based navigation tasks. In this thesis, several scalable algorithms for incorporating geometric priors into online localization and mapping are proposed, and a multi-scale framework which combines the techniques is presented. We tackle the drift and scaling problems by localizing selected keyframes in a global coordinate system relative to previously reconstructed, optimized, and aligned visual maps.

## 2.2 Visual Localization in a Known Model

The problem of determining the position and orientation of a camera relative to an already known model is called *pose estimation*. When the camera intrinsics have been calibrated beforehand [80, 100], six degrees of freedom (DoF) remain; three DoF for position and another three DoF for the orientation of the camera. In the following, we distinguish between three different methods for pose estimation.

Typically, methods for pose estimation rely on local image descriptors [12, 84, 135] to establish correspondences between a model view and a query view. Initially, a large number of images of the model from different, known points of view are created, and descriptors are extracted at salient image locations. During the query phase, descriptors are extracted in the query view as well, and then matched against the set of model views to find the best fit. If the 2D to 3D correspondences for the points in the model view are known, an absolute 6 DoF pose can be computed [118]. While descriptor-based methods have proven to be efficient in many scenarios, they have problems with weakly textured scenes and changes in illumination. Unluckily, such changes often occur in outdoor environments due to weather and seasonal variations. Moreover, establishing correspondences might fail if the viewing angle between model and query image is too large. State-of-the-art keypoint detectors and image descriptors are robust to a certain degree of variation in appearance [145], but this is still a major issue which needs to be tackled by the localization framework.

In contrast, correspondence-based approaches [20, 35, 39, 58, 148] use point configurations and therefore circumvent the appearance-based matching problem. While being ideal in weakly textured regions, they have to deal with the large search space resulting from establishing all possible point correspondences if no pose prior is given. The search space can get even bigger when additional features are introduced by clutter. This can be avoided by choosing more complex features such as lines or other geometric primitives, which reduces the search space on the one hand but makes the algorithm less robust, for instance to partial occlusions of the model. RANdom SAmple Consensus (RANSAC) [58] style approaches are successfully applied to problems where correspondences have to be hypothesized, but they are not suitable for time-critical applications as the number of iterations needed to obtain satisfactory results is quickly increasing with the number of 3D model points and detected 2D image points. Due to the fact that we need to estimate the pose not only for single images but for a continuous sequence, tracking the correspondences is more suitable.

Tracking-based visual pose estimation approaches [20, 46, 112, 155, 168] can rely on the fact that the images are only subject to small changes from one frame to the next. In turn, the framerate delivered by the camera needs to be high enough with respect to its motion and the distance to the scene. Also, the pose estimation needs to be done in real-time as the algorithm relies on this prior when coping with the next frame. Such approaches are very similar, and in most cases even identical, to what is used in visual SLAM systems; thus they suffer from the same drawbacks.

In our work, we focus on tracking-based methods for continuous pose estimation in point-based and wireframe model-based SLAM, but correct for the resulting drift using descriptor-based methods during global localization. To cope with severe appearance changes, we propose to create several prior models which are acquired in different conditions, and align them in the same coordinate system (Section 4.1.3).

## 2.3 Path Planning

Path and mission planning have a long history in robotics. Classical algorithms such as Dijkstra's algorithm [43] and the A* algorithm [78] apply global graph search to find the least-cost path from a given starting point to a target. Other algorithms [121, 126] sample the local environment to find the least-cost path. The best path is often considered to be the shortest path, but it could also be the path with maximum coverage of an area, minimal energy use, or optimized predicted perception quality. In some cases it is even most beneficial to choose from a given set of trajectories which can be followed by the robot's controller rather than planning a specific and maybe impossible path [42]. For this reason, different path planning algorithms are used based on the situation. In the following, we introduce two major groups of techniques, namely heuristic and probabilistic methods.

### 2.3.1 Heuristic Path Planning Methods

Heuristic methods use an estimated cost function for target-oriented path searching, which considerably reduces the computation time. The heuristic function includes the cost from the starting node to the current location, $c(s, s')$, and the estimated cost from the current location to the goal, $h(s')$. Path planning employs *consistent* heuristics which are based on two conditions: First, the estimate must never overestimate the costs of any vertex s. Second, it needs to satisfy $h(s) \leq c(s, s') + h(s')$ for all vertices $s \in S$ and $s' \in Succ(s)$, where $h(s)$ is the estimated cost of a vertex s, $c(s, s')$ is the true cost between two vertices and $Succ(s)$ denotes a set of successors of vertex s. If only the first condition is satisfied the heuristic is said to be *admissible*. The advantage of heuristic methods is that they are complete, i.e., they always find a solution if one exists.

A well-known heuristic path planner is the A* algorithm [78]. A* follows the path with the lowest expected total cost $c(s, s') + h(s')$ as it traverses the graph towards the goal node. To keep track of alternate paths, a priority queue is maintained. A* is similar to Dijkstra's algorithm [43] except that it guides its search towards the most promising states, which can save a significant amount of computation effort.

The limitation of the above approaches is that they need a full map of the area under exploration. However, when operating in real-world scenarios, new information might be added to the map so replanning is essential. While A* could be used to plan from scratch for every update, this is computationally very expensive. Instead, Focussed Dynamic

A* (D*) [191] and D* Lite [119] search for a path from the goal towards the start, and nodes are only updated when changes occur. An updated path is calculated based on the previous path, which is much more effective.

Finally, Field D* is an interpolation based path planning and re-planning algorithm [57]. In contrast to other methods where nodes are defined as the centers of grids, field D* defines nodes on corners of grids. Linear interpolation is then used to create waypoints which are located along the edges of grids. This allows to plan direct, low-cost, and smooth paths in non-uniform environments. D* and its variants are widely used for autonomous robots, including Mars rovers and autonomous cars [192, 208].

### 2.3.2 Probabilistic Path Planning Methods

Heuristic methods are not efficient in high-dimensional spaces as there are too many possible states. Probabilistic approaches avoid these problems by randomly sampling the configuration space. While this helps to decrease planning time and memory usage, the main disadvantage is that an optimal path cannot be guaranteed anymore.

One of the most popular methods is the Probabilistic Roadmap (PRM) algorithm [105, 106]. PRM planning generally consists of two phases: First, the roadmap graph is built by randomly sampling points in the configuration space. Then, the sampled configurations are connected to their neighbors. Different strategies to connect the nodes exist, but typically the $k$-nearest neighbors are added to the graph. Second, in the query phase, the start node and the goal node are connected to their nearest neighbors in the graph, and the path is calculated using a heuristic method. PRMs are provably probabilistically complete [27], meaning that with an increasing number of samples any existent path will be found. However, situations such as narrow corridors in large environments can increase the path planning time rapidly and thus deliberate sampling strategies are necessary.

PRMs allow multiple queries to be executed on the same graph but need some pre-processing. Typically, obstacles are defined during this preprocessing phase. When using visual input for path planning, it is important to close the gaps between individual points in a point cloud so that no path can be planned in between. In other words, dense models are required for proper collision avoidance. In this thesis, we present algorithms to create dense digital surface models as prior knowledge which can be used for large-scale path planning, but also methods to estimate watertight surfaces on-the-fly based on input from visual SLAM.

## 2.4 Control Strategies

Control theory tackles the problem of influencing the dynamic behavior of a system to reach a certain target state. In general, one can distinguish between open-loop control systems which excite the system without measuring the output, and feedback control systems which use the error between measured output and expected output for control.

A large body of research regarding stability, observability, and system identification is available [45, 89, 189]. However, in this thesis we will only focus on control to an extent that is necessary for implementing our visual navigation strategies on micro aerial vehicles.

### 2.4.1 Cascaded Control

In general, feedback controllers are used for autonomous robots. The goal is to minimize the error between a desired system state and the noisy measured system state by controlling the system's input variables. Because MAVs are in general highly unstable and nonlinear systems, typically a cascaded control structure is used [15, 21]. In such systems the outer control loops view inner loops as quasi-static systems, thus inner loops need to operate much faster. The individual parts, from inner to outer loops, are introduced in the following.

For MAVs such as quad-rotor helicopters, it is feasible to control the speed of four rotors independently, or on a higher level to control the three system angles roll $\Phi$, pitch $\Theta$, and yaw $\Psi$, as well as the thrust $T$. Many commercially available MAVs, such as the Ascending Technologies Pelican, Falcon, and Parrot AR.Drone quad-rotors used in our work, already come with an attitude controller which translates angular and thrust commands into rotational speeds [73] at control cycles up to 1000 Hz. Attitude control is based on IMU and accelerometer readings and provides a good basis for robust and stable flight. However, due to the lack of exteroceptive sensors it is prone to drift, and depends on the accuracy of the system model.

Visual servoing techniques use visual input to control the motion of a robot. Given a 6 degrees of freedom (DOF) visual pose estimate, it is possible to control the position of the MAV in space using Position-Based Visual Servoing (PBVS). A position controller has four inputs, namely the desired position $(x^*, y^*, z^*)$ and the desired yaw angle $\Psi^*$. Roll $\Phi$ and pitch $\Theta$ depend on the position commands, as the MAV has to nick or roll to obtain a certain position. To cope with the drift of pure attitude control, control commands should be sent at a rate of approximately 20-100 Hz. By changing the desired values accordingly, the position controller is used to control an autonomous MAV during take-off, hovering, and landing; in other words, a desired trajectory could be followed. The trajectory is typically defined by the path planning algorithm as discussed in the previous section. Typical trajectory control rates depend on the inner loops and the path replanning speed, but should be around 0.2-5 Hz.

Cascaded control is a concept which works with various kinds of controllers on each nesting level. Often, variants of the well-known PID controller are employed; for instance, the realization of the attitude controller mentioned above is based on a PD controller [73]. We therefore briefly introduce this control design next.

### 2.4.2 PID Control

The proportional-integral-derivative (PID) controller is the most popular and widely used controller in industrial applications. For a PID controller, the relationship between the error term $e(t)$ at the input of the controller and the output $u(t)$ (corresponding to the system input) is

$$u(t) = k_P\, e(t) + k_I \int_0^t e(\tau)\,\mathrm{d}\tau + k_D \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \tag{2.1}$$

where $k_P$ is the proportional gain, $k_I$ is the integral gain, and $k_D$ is the differential gain.

The proportional term $k_P e(t)$ represents the present error. While too high values of $k_P$ cause a system to turn unstable, low values affect the responsiveness of the system. The integral term $k_I \int_0^t e(\tau)\,\mathrm{d}\tau$ accumulates the past errors and thus eliminates steady-state errors of purely proportional controllers. However, setting $k_I$ too large causes overshoots. Finally, the differential term $k_D$ predicts future errors and thus can improve settling time and stability. The differential term is sensitive to noise, so large amounts of noise and a high $k_D$ value can turn the controller unstable as well. Many theoretical methods for estimating the three parameters exist [45, 89], but in practice the parameters are often found experimentally.

The PID control design appears in many MAV controllers [111, 217, 225] due to its relatively simple implementation procedure. The controller is well suited to follow pre-planned trajectories [14], especially when GPS is used for sensing. However, when trying to fly based on visual input the pose estimation noise causes problems in the differential term [108], and high-accuracy flights are hard to achieve. Most recent approaches to MAV control thus use more sophisticated but strictly modeled systems such as the Linear Quadratic Gaussian controller with Loop Transfer Recovery (LQG/LTR) [2, 21], or a more flexible control strategy such as fuzzy control [224, 236] which simplifies parameter tuning.

## 2.5 Summary

In this chapter, we have introduced state-of-the-art approaches for micro aerial vehicle navigation, as well as for the major components of such a system including localization, mapping, path planning, and control. A large amount of previous work relies on Simultaneous Localization and Mapping (SLAM) techniques to tackle the perception part of the navigation problem. However, especially monocular visual SLAM techniques require considerable amounts of computational power to operate in real-time. The few available methods suffer from error accumulation during tracking and from missing robustness to cope with large-scale outdoor environments. In contrast, suitable algorithms for path planning and control are readily available, but need to be chosen wisely depending on the application at hand.

In the next chapters we thus present tools for modeling geometric prior knowledge, a robust localization framework, and path planning as well as control approaches which

are adapted to visual input. We aim to tackle the major challenges presented in this chapter including missing scale information in monocular SLAM, drift, global relocalization when losing track, appearance changes, repetitive structures, weakly textured models, and scalability given the low computational power onboard MAVs.

# Chapter 3

# Modeling Geometric Prior Knowledge

**Contents**



Generating 3D models from a set of images has become a widely studied field of research over the last few years, and good progress has been made to reconstruct and represent a variety of different environments [160]. In this thesis we focus on exploiting the multitude of prior knowledge in form of such models that is often readily available for improving visual navigation of mobile robots. More specifically, we observe that significant parts of urban scenes such as buildings do not alter their geometry for months, sometimes even years, and can therefore serve as natural landmarks. Imagery for our approach can be collected from many sources such as Microsoft Bing Streetside or Google Street View, or by taking pictures of the area with a consumer camera. Additionally, one can exploit novel image acquisition tools such as micro aerial vehicles to take pictures from completely different viewpoints with ground sampling distances below 1 cm. In the following, we survey the different classes of algorithms which can be used to create geometric prior knowledge.

**3D Point Clouds.** The majority of available algorithms for 3D reconstruction in the context of mobile robots is based on point correspondences between multiple views using various local descriptors such as the Scale-Invariant Feature Transform (SIFT) [135] in order to obtain a 3D point cloud while simultaneously solving for camera orientations. This process is called Structure-from-Motion (SfM). It is a fundamental and well studied

problem in both computer vision and photogrammetry [80]. In contrast to approaches which directly estimate a dense depth map for every acquired frame, for instance by using stereo imaging [142] or active sensing [152], SfM systems extract depth information from a set of images which are acquired one after another. SfM has reached maturity over the past years, and meanwhile fully automatic pipelines for ordered [161] and unordered [186, 187] image collections exist. Processing is typically done offline, which allows to quickly capture images in the field and then do the processing in the lab with powerful hardware. An overview of a typical SfM pipeline can be found in Figure 3.1.

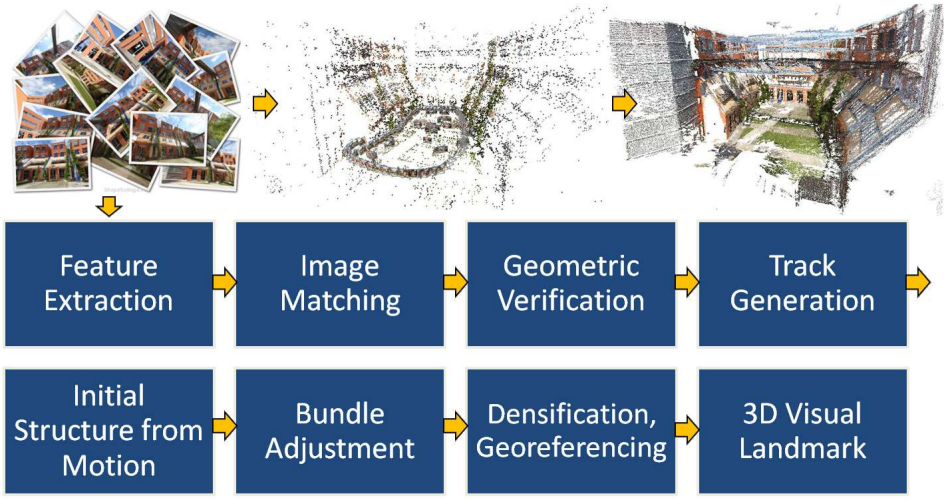



Figure 3.1: Overview of a point-based Structure-from-Motion pipeline. After extracting and matching interest points, hypothesis for pairwise epipolar geometries are verified. The resulting camera poses are greedily added to the reconstruction and bundle adjustment is used to optimize the result. After densification and georeferencing, a suitable representation of geometric prior knowledge is available.

**3D Lines.** The density of the resulting point clouds highly depends on the amount of texture available in the images. Therefore, point-based SfM may deliver poor results in environments with a low amount of distinctive interest points. To tackle this issue, many line-based reconstruction approaches have been presented over the years. Especially buildings and indoor environments can often be represented by a set of 3D line segments. Similar to traditional SfM it is necessary to match 2D line segments from various views to triangulate a 3D line segment. This can be done using appearance-based similarity measures, for instance normalized cross-correlation (NCC) or line descriptors [110, 241], possibly combined with additional geometric constraints [19]. Since the endpoints of matched line segments usually do not correspond to each other due to inexact line segment detection or occlusions, creating 3D line segments from matched 2D lines is much more difficult than traditional point-to-point matching.

**Dense Multi-view Stereo Geometry.** Another method of acquiring geometric knowledge is dense scene reconstruction. Classical point-based or line-based Structure-from-Motion yields camera orientations and a sparse set of triangulated primitives, and this is sufficient for a localization task. However for geo-referencing, occlusion handling, or simply user-friendly presentation a denser model is required. Dense models are typically obtained using two-view or multi-view stereo estimation based on given camera orientations, where one camera serves as the so-called key view and the other(s) as the sensor view(s). As a result one obtains a 2.5D representation of the scene where every pixel in the key view holds an assigned depth. The individual depth maps finally have to be fused into a single 3D representation.

**Digital Surface Models.** A digital surface model (DSM) is a 2.5D height representation of the Earth's surface and all objects on it as seen from an orthogonal aerial viewpoint. In contrast to a full 3D model, just the maximum height over ground is considered, rather than looking at all structures that might be below. Digital surface models are typically created from large-format aerial images with up to 250 Mpx with a ground sampling distance of 8-10 cm/pixel. While DSMs can also be obtained using modern tools such as fixed-wing MAVs, classical aerial images provide the advantage that they cover a very large area on the ground and thus the resulting reconstructions are inherently geometrically consistent. The aerial images are typically geo-referenced, either by using a high-quality GPS receiver in the plane or by using ground control points on the surface. DSMs therefore provide a good basis for the global localization of point clouds and for high-level path planning.

**Other Priors.** Several other forms of prior knowledge are often readily available. Mobile robots typically carry additional sensors such as an inertial measurement unit (IMU), a 3D magnetic compass, and a global positioning system (GPS) receiver. Also, a robot has several motion constraints which allow to reason about the current position given the previous location estimate. Finally, visual semantics and behavioral data can be exploited for visual navigation. The experience of a human expert, as well as the experience of other robots traversing a certain path previously, could be shared amongst robots as geometric prior knowledge.

In the following sections we will review algorithms for successful modeling of these different geometric priors. In contrast to visual navigation algorithms, where low computational costs and real-time capability are required, the modeling approaches presented in this chapter focus on accuracy and are meant to run offline. However, receiving online feedback about the expected quality and completeness of a reconstruction is important when processing imagery offline. We thus conclude the chapter with a section on optimized strategies for manual and automated image acquisition.

## 3.1 Point-based Scene Reconstruction

For sparse 3D model reconstruction we rely on a Structure from Motion (SfM) approach that is able to reconstruct a scene from unorganized image sets. Our solution to the 3D reconstruction problem is based on the work of [98] and [221]. It is widely applicable since no prior knowledge about the scene is necessary (i.e., no sequential ordering of the input images has to be provided) and can therefore be used to create 3D models from terrestrial as well as aerial imagery. To accelerate the computations we take advantage of graphic processing units (GPUs) for efficient parallelized computing.

### 3.1.1 Related Work

The recent success of Structure from Motion techniques can be traced back to significant advances in wide-baseline feature matching [135] and multiple-view geometry [154, 203]. Also, the advent of much more powerful computational hardware, and especially general-purpose graphic processing units (GPGPU) [59], supported the applicability of 3D reconstruction from image sequences. Still, exhaustive pairwise feature matching to find correspondences in the individual views is the most time demanding task in today's reconstruction pipelines. To circumvent this issue, Agarwal et al. [3] introduced a two-stage matching approach which is now considered best practice. First, every image is described by a single feature vector and coarse matches between images are detected. This process is based on image retrieval and bag-of-words concepts [185]. Then, only those image pairs which achieve a high similarity score are used for detailed feature matching. Our pipeline incorporates all of these ideas to create robust and accurate 3D models.

### 3.1.2 Establishing the Epipolar Graph

A typical SfM framework consists of three processing steps, namely feature extraction, matching, and geometry estimation. These steps are summarized in the following paragraphs.

First, salient features are extracted from each frame. Our method utilizes the very effective combination of DoG keypoint detector and SIFT descriptor [135] which achieves excellent repeatability performance for wide baseline image matching [145]. In particular we rely on the publicly available SiftGPU [232] software.

We then match keypoint descriptors between each pair of images. A variety of approaches has been proposed to speedup nearest neighbor matching in high-dimensional spaces (like the 128-dimensional SIFT descriptor space). Among the most promising methods are randomized kd-trees [5] with priority search and hierarchical k-means trees [63]. These algorithms are in general designed to run on a single CPU and are known to provide speedups of about one or two orders of magnitude over linear search, but the speedup comes with the cost of a potential loss in accuracy [150]. On the other hand, given that the number of features is limited to some thousands, nearest neighbor search, implemented

as a dense matrix multiplication on recent graphics hardware, can achieve an equivalent speedup but delivers the exact solution. Hence, we employ a GPU-accelerated feature matching approach based on the CUBLAS library with subsequent instructions to apply the distance ratio test and to report the established correspondences.

After matching relevant images to each query view, geometric verification based on the Five-Point algorithm [154] is performed. Given five corresponding points in two calibrated viewpoints with unknown extrinsic parameters, the Five-Point algorithm recovers the relative 3D positions of the points, as well as the pose of the second camera $P_1$ relative to the first camera $P_0$.

Since matches that arise from descriptor comparisons are often highly contaminated by outliers, we employ the random sample consensus (RANSAC) [58] algorithm for robust estimation. In its basic implementation, RANSAC acts as a hypothesize-and-verify approach. From a minimal set of samples several hypotheses are generated and the consensus of the model is evaluated on the full set of observations. The RANSAC termination confidence,

$$p = 1 - \exp(N \log(1 - (1 - \epsilon)^s)) \tag{3.1}$$

is used to decide whether two images satisfy the epipolar geometry. Here, $N$ is the total number of evaluated models, $w = 1 - \epsilon$ the probability that any selected data point is an inlier, and $s = 5$ is the cardinality of the sample point set used to compute a minimal model for the Five-Point algorithm. We require $p > 0.999$ in order to accept an epipolar geometric relation. In our experiments, we use up to $N = 2000$ models which corresponds to a maximal outlier fraction of $\epsilon = 0.67$.

The matching output is a graph structure denoted as epipolar graph, that consists of the set of vertices $\mathcal{V} = \{I_1 \dots I_N\}$ corresponding to the images and a set of edges $\mathcal{E} = \{e_{ij} | i, j \in \mathcal{V}\}$. The edges are pairwise reconstructions, i.e. relative orientations between view $i$ and $j$, $e_{ij} =< P_i, P_j >$, and a set of triangulated points with respective image measurements. Since the cameras are calibrated, a linear triangulation method [79] is sufficient to accurately estimate the 3D point location. This procedure is followed by a pruning step that discards ill-conditioned 3D points (points that do not satisfy the cheirality criterion [80]) and points that have a high depth uncertainty (points where the roundness of the confidence ellipsoid [13] is larger than a given threshold).

### 3.1.3 Structure Initialization

Our SfM method follows an incremental approach [186] based on the epipolar graph. In order to reconstruct a consistent 3D model, a robust and reliable start configuration is required. When the initial structure is prone to errors, a subsequent iterative optimization procedure will eventually end up in a wrong local minimum, hence good initialization is critical. As proposed in [116] we initialize the geometry in the most connected parts of the graph, therefore the vertex $I^*$ with highest degree, i.e. the node having the largest number of edges, is determined. We start from the vertex $I^*$ and determine all connected

neighbors. Next, all pair-wise measurements are linked into point tracks and the global scale factor of the initial structure is estimated. Then, bundle adjustment [203] is used to optimize camera orientations $P_i$ and 3D points $\mathbf{X}_j$ by minimizing the reprojection error,

$$\mathcal{C}(P, \mathbf{X}) = \sum_i \sum_j v_{ij} d(P_i \mathbf{X}_j, \mathbf{x}_{ij})^2 \tag{3.2}$$

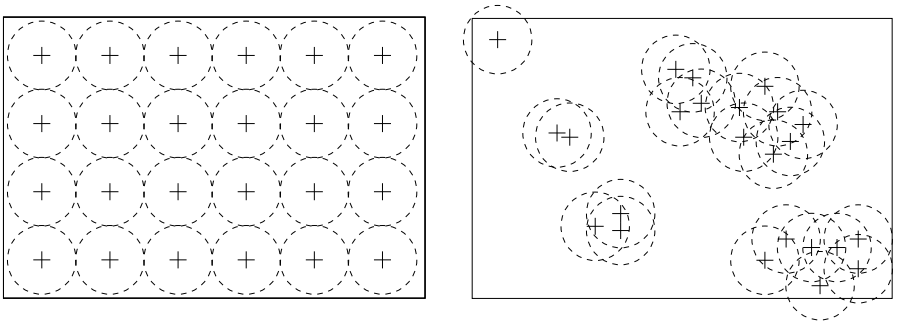
where $\mathbf{x}_{ij}$ are 2D point measurements of observed 3D points and $v_{ij}$ is an indicator variable that is 1 if the point $\mathbf{X}_j$ is visible in camera $P_i$ and 0 otherwise. To limit the impact of outliers, we use a robust Huber M-Estimator [95] to compute the costs $d$.

Given the initial optimized structure, each 3D point is back-projected and searched for in every image. We utilize a 2D kd-tree for efficient search and restrict the search radius to a constant factor $r_t$. Again, given the new measurements, bundle adjustment is used to optimize 3D points and camera parameters. This method ensures strong connections within the current reconstruction.

### 3.1.4 Incremental Reconstruction

Next, for every image $I$ that is not reconstructed and has a potential overlap to the current 3D scene (estimated from the epipolar graph), 2D–to–3D correspondences are established. A three-point pose estimation algorithm [76, 118] inside a RANSAC loop is used to insert the position of a new image. When a pose can be determined (i.e., a sufficient inlier confidence is achieved), the structure is updated with the new camera and all measurements visible therein. A subsequent procedure expands the current 3D structure by triangulation of new correspondences. We follow the approach of Snavely et al. [186] and use a priority queue to guide the insertion order. Our insertion order is based on a saliency measure that favors early insertion of images that have a strong overlap with the given 3D structure. Rather than using the raw number of potential 2D–to–3D matches, we compute an effective matching score that further takes the spatial match distribution into account. This idea of Irschara et al. [101] is depicted in Figure 3.2. While the number of features is equal in 3.2(a) and 3.2(b), the uniform spatial distribution of point features in 3.2(a) can be regarded as more reliable than the one shown in 3.2(b). Hence, we weight the raw number of features by an estimate for the covered image fraction yielding the effective inlier count.

Whenever a number of $N$ images is added (we use $N = 10$), bundle adjustment is used to simultaneously optimize the structure and all camera poses. The iterative view insertion procedure is repeated until the priority queue is empty. The sparse reconstruction result can be seen in Figure 3.3(b).



(a) Many effective inliers (b) Little effective inliers

Figure 3.2: Effective inlier measure. (a) A uniform spatial distribution of image measurements is regarded as more reliable than (b) a non-uniform distribution.

### 3.1.5 Online Structure-from-Motion

Classically, batch-based SfM approaches assume spatially unordered images as input and therefore require several minutes or hours to determine the spatial ordering by constructing the epipolar graph [187]. While the resulting reconstruction is optimized, some applications require to report the current reconstruction quality to a user, so the SfM problem has to be solved in an online fashion. Visual SLAM systems like PTAM [113] are able to recover camera positions and 3D structure in real-time, but they rely on high framerates and strong motion assumptions and are therefore not suitable for generic modeling of prior knowledge. Additional to our offline pipeline, we have thus proposed an online Structure-from-Motion method [87].

The construction of the epipolar graph requires to estimate the relative orientations between all image pairs, which is the most time-consuming task in batch SfM pipelines. In practice, we can assume that a user does not acquire images in a totally random order. If we assume that a new input image $I$ has an overlap to an already reconstructed scene part, we can skip the epipolar graph construction and the SfM problem can be split into two tasks that are easier to solve: A localization and a structure expansion part. More formally, given a freshly acquired input image $I$ and a reconstructed scene $M$, we find the position of $I$ within $M$ using robust three-point pose estimation [76, 118], and subsequently expand the map $M$ by triangulation.

For bootstrapping the scene $M$, we rely on the same initialization schemes that are also used in batch-based SfM methods but only use the first two suitable images. To prevent the triangulation of degenerated 3D points lying at infinity, we require that the triangulation angle exceeds a minimum threshold of $\delta_{min} = 2°$. To prevent scene drift caused by incremental map building, we use a global optimization scheme to obtain a consistent map. Hence, we perform iterative bundle adjustment [203] in a parallel thread.

(a)

(b)

Figure 3.3: Atrium scene reconstruction from 157 views. (a) Original view of the scene. (b) Sparse model and source cameras obtained by Structure from Motion reconstruction.

The presented method is similar to visual SLAM, but it matches wide-baseline features instead of tracking interest points. It is designed to process high-quality images in a user-acceptable time, but does not impose real-time constraints. An important advantage over offline SfM pipelines is the ability to provide feedback to the user: If the system cannot localize $I$ within $M$, this is reported to the user instantly and he is asked to take a new picture. The drawback however is that not the entire imagery is available in the beginning, thus the reconstruction might not be optimal.

### 3.1.6 Summary

We have presented methods for offline and online Structure-from-Motion reconstruction of sparse point clouds [87, 98, 221]. Our generic reconstruction pipeline is widely applicable since no prior knowledge about the scene is necessary. It can therefore be used to create 3D models from terrestrial as well as aerial imagery. Moreover, when operating in an online manner it is possible to obtain a low-resolution reconstruction preview while still acquiring images. This is especially helpful when using micro aerial vehicles for photogrammetry or when recording at distant locations, because the costs for acquiring additional imagery to complete the reconstruction is considerably high. The accuracy and usability of our pipeline is demonstrated in Section 6.1.1.

## 3.2 Line-based Scene Reconstruction

Line-based scene reconstruction approaches have been developed to overcome situations where point-based modeling has problems, namely in man-made environments with a low amount of distinctive interest points. Most of the previous approaches rely on an accurate line matching process between the various views using appearance-based similarity measures [10, 11, 97, 182]. This usually works fine if the lines are located on a planar surface with constant background, for instance when matching window frames. However, when dealing with wiry structures such as power pylons, bridges or scaffolds, appearance-based matching is hardly possible due to a visually different background of the line segments in different views. Figure 3.4 visualizes this problem. We present a hypothesize-and-verify approach which is especially designed to handle such cases but also performs well on solid objects [86].

### 3.2.1 Related Work

Similar to traditional SfM it is necessary to match 2D line segments from various views to triangulate a 3D line segment. This can be done using appearance-based similarity measures, for instance normalized cross-correlation (NCC) [10] or line descriptors [110, 241], possibly combined with additional geometric constraints [19]. Motivated by the application of autonomous power pylon inspection using MAVs, we aim for reconstructing wiry objects as well. Unfortunately, appearance-based approaches do not perform well in these cases, thus we need to rely on geometric constraints only.

In order to create 3D models without the need for explicit line matching, Jain et al. [102] developed a sweeping based approach which defines the unknown 3D locations of the endpoints of 2D line segments as random variables. They estimate 3D line hypotheses by generating all possible endpoint locations in a certain depth interval (assuming known camera intrinsics and extrinsics) and keep the one with the highest score based on the gradient images of many neighboring views. Hence, they create a 3D line for every 2D line in every view. In order to delete outliers and cluster corresponding line segments together,

(a) (b)

Figure 3.4: Examples for wiry structures. (a) Objects such as a power pylon or a scaffold are hard to model using point features. (b) Appearance-based line matching is not possible either because corresponding line segments have a visually different background in the two views (yellow lines).

they group 3D line segments which lie close in space and discard all segments which do not have at least one such neighbor. They also perform an optimization based on 2D line connections using loopy belief propagation to enforce connected 3D lines. Even though their approach delivers accurate results and is robust against noise and partial occlusions, it is very slow compared to appearance-based approaches.

In our approach as presented in [86] we build upon these principles but employ a different matching strategy. Instead of using a time consuming sweeping approach we generate hypothetical 3D line segments using epipolar constraints, which drastically limits the number of possible 3D locations for each 2D line segment. Our algorithm consists of a hypotheses creation and a verification step, which are summarized in the following.

### 3.2.2 Creation of Line Segment Hypotheses

Given an unordered set $I = \{I^1, ..., I^n\}$ of $n$ images and the corresponding cameras $C = \{C^1, ..., C^n\}$ we want to generate a set of 3D line segments $S = \{S^1, ..., S^k\}$. Since we do not perform explicit line matching and line-based relative pose estimation, the cameras have to be known beforehand. For this purpose we use a point-based SfM pipeline as introduced in the previous section, which results in relative poses of all cameras $C$ in a common coordinate frame. This limits the application to scenes where interest points can

be found, but we have seen that we can usually find enough correct correspondences for an accurate relative pose estimation directly on solid objects, but also in the background of wiry structures.

In order to generate triangulated 3D line segments from a set of images, we first apply the Line Segment Detector (LSD) [214] algorithm to extract all relevant line segments with as few incorrect detections as possible. Assuming no false detections, every 2D line segment from image $I^i$ corresponds to a 3D line segment in world space. As we know the projection matrix $P^i$ of the camera, we are able to compute the epipolar geometry [80] between $I^i$ and some other view $I^j$. Using the epipolar lines $e_p$ and $e_q$ defined by the two endpoints $p$ and $q$ of a certain line segment $l$ in view $i$, we can limit the possible matches for $l$ to those line segments whose endpoints lie on $e_p$ and $e_q$ respectively. This is visualized in Figure 3.5. For every hypothesis we create a 3D line segment $L$ by triangulating the two corresponding endpoint pairs from the two views $I^i$ and $I^j$.

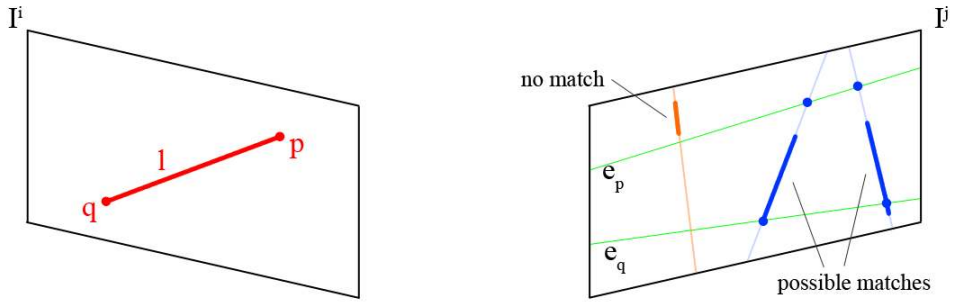



Figure 3.5: Epipolar geometry-guided line matching. We match the line segment $l$ in view $I^i$ with line segments from view $I^j$ using its epipolar lines $e_p$ and $e_q$. The blue line segments are possible candidate matches because of their overlap with the region between the two epipolar lines. The endpoints of the hypothetical line segments used for triangulation are shown as blue dots. The orange line segment does not overlap with the epipolar lines and is therefore not considered as a possible match.

### 3.2.3 Verification and Clustering of Line Segments

Epipolar lines do not provide enough information to perform exact matching, so we usually have more than one hypothesis for each 2D line segment and have to determine which one is correct. Therefore we adopt a scoring approach similar to [102, 109] which weights backprojected lines according to the perpendicular distance to the closest image gradient. Assuming that line segments correspond to high gradient areas in images, this method ensures that we choose the hypothesis which fits best to the image data. The 3D line segment with maximum score is thus added to the hypotheses set $H$. Since we generate 3D line segments for all views individually, we end up with a quite large set of 3D lines which has to be pruned. Figure 3.6(b) shows an example for a 3D line model before grouping and outlier removal.

(a) (b) (c)

Figure 3.6: Line-based reconstruction results. (a) Original view of the scene. (b) Before grouping and outlier removal, 71538 segments from 106 views are generated. Note that there is a large number of outliers due to incorrect matches, but the power pylon which appears in the imagery is clearly recognizable. (c) After the grouping procedure most of the outliers have been successfully removed, resulting in an accurate 3D model with 1381 line segments.

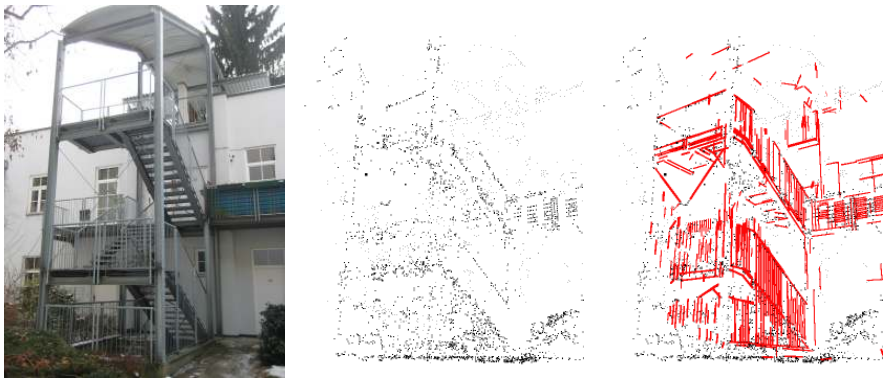
Since we match and triangulate the 2D line segments individually for every view, the same 3D line might be generated in multiple views. Assuming a correct matching procedure, all hypotheses in $H$ which correspond to the same 3D line should be located close to each other in space. Hence, we greedily group lines based on their spatial proximity in 3D space [102]. For each line $L_m \in H$ we define a cylinder of a fixed radius $r$ around the line and then search for all line segments $L_n, n \neq m$ where both endpoints are located within the cylinder. If the final line group has at least $h_{min}$ members we consider it to be valid and exclude all line segments in the group from further grouping, otherwise $L_m$ is removed from $H$ and we continue with the next best hypothesis. After the clustering step, each group is replaced with one single 3D line segment for our final set $S$. Figure 3.6(c) illustrates the result of the grouping procedure. Note that compared to Figure 3.6(b) most of the outliers have been successfully removed.

Even for solid objects, it is often the case that the point cloud obtained using traditional SfM techniques is sparsely distributed due to the lack of distinctive features in man-made environments. Many of the keypoints may be rather located on the background instead of the object. Nevertheless, background features can be used for relative pose estimation and therefore our line matching algorithm can be applied in order to densify the 3D model. Figure 3.7 shows an example 3D point cloud of a house surrounded by a scaffold, and Figure 3.8 shows a model of a staircase. As we can see, the point clouds are rather sparsely

occupied and the visible objects are difficult to determine for the viewer. Adding 3D line segments clearly improves the result and allows the viewer to identify the underlying structure.

(a) (b) (c)

Figure 3.7: Line-based reconstruction of the *House* sequence (93 images). (a) Example view, (b) SfM point cloud, (c) 3D model with the reconstructed line segments.



(a) (b) (c)

Figure 3.8: Line-based reconstruction of the *Stairs* sequence (14 images). (a) Example view, (b) SfM point cloud, (c) 3D model with the reconstructed line segments.

In case of wiry objects a specific volume (e.g. the cylinder used for grouping) can be assigned to the object rather than representing it as a pure line model. As visualized in Figure 3.9, the appearance then varies depending on the viewing distance from the object, which is a more natural representation of real-world geometry. Such a model is perfectly suitable for model-based localization as presented in Section 4.4.

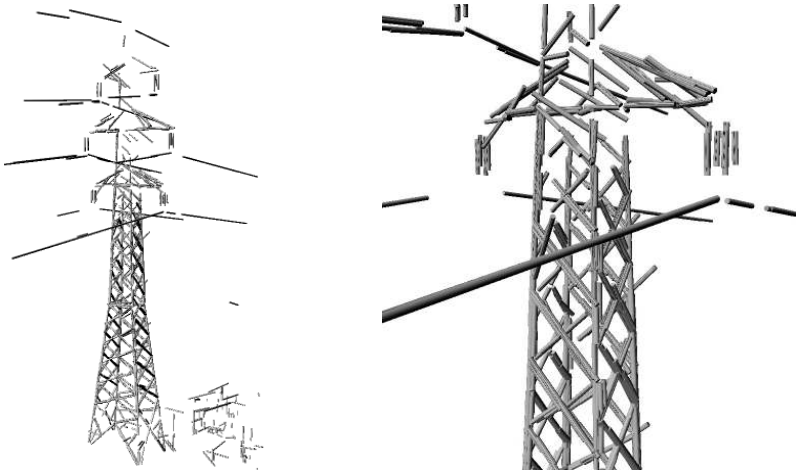


Figure 3.9: Cylinder approximation for line models. For model-based tracking, realistic rendering of the object is crucial. Thus, cylinders are used to approximate the volume of individual line segments. A change of the viewing distance thus effects the object size as in a real-world setting.

### 3.2.4 Summary

In summary, our approach for 3D line-based scene reconstruction [86] without explicit appearance-based matching works for wiry structures as well as solid objects. In contrast to a previous approach by Jain et al. [102] we exploit epipolar constraints to speed-up the computation while still creating accurate results. An evaluation can be found in Section 6.1.2. We have shown that 3D line segments can be used to densify the result in scenes with few distinctive feature points. This is of particular importance for localization in urban scenes with man-made or wiry structures.

## 3.3 Dense Multi-View Scene Reconstruction

Dense models contain a lot more information than sparse models, but it is also much harder and computationally more expensive to obtain accurate dense models. Pixel-wise depth estimation is often not possible due to untextured regions or occlusions, so depth map smoothing, interpolation, and meshing algorithms need to be applied. Additionally, modification and optimization of dense point clouds is not straight-forward, for instance when loop closing is necessary. We thus rely on sparse point clouds as basic representation of the environment and refer to densification and surface reconstruction as refinement steps for certain applications. In the following, we first introduce representations of dense geometry, then review two methods to densify point clouds, namely patch-based multi-view stereo [65] and planesweep stereo [28], and finally demonstrate two methods which we use to extract 3D surfaces.

### 3.3.1 Related Work

Dense geometry can be defined in multiple ways. We distinguish between densified or semi-dense point clouds, and watertight surfaces. Even for very dense point clouds it is not entirely clear which points belong together and thus form a common surface. In contrast, our understanding of dense geometry incorporates a neighborhood for every point, thus spanning surfaces between individual points. Several different representations for such kinds of geometry exist.

Volumetric representations are the 3D equivalent of occupancy grids [147], as every voxel is assigned to be filled (i.e. part of an object) or empty. Often, a probability is assigned instead of a binary value. Such representations are especially popular in robotics as classical algorithms like histogram filters [25] can be directly extended to 3D. While these representations can handle arbitrary topology, the biggest drawback is the large memory consumption which scales cubically. Hornung et al. [90] overcome this issue in their Octomap framework, which uses a hierarchical data structure based on octrees [141, 230] for 3D space subdivision. Octomap allows fast multi-resolution queries and insertions, and it scales very well to outdoor environments. However, a major drawback is the lack of a fixed size neighborhood for each cell, which is necessary for efficient filtering and is usually a core advantage of volumetric representations.

Several methods use volumetric representations to extract a surface by computing a signed distance field separately on all voxels induced by the range images [34, 83, 229]. Surface regularization can be integrated to allow for spatial coherence and smoothness within a global energy minimization framework, which can then be optimized using variational techniques [70, 223, 235] or graph cuts [213]. The latter method tries to reconstruct objects from the captured input images directly by incorporating a photo-consistency measure.

Another very popular and flexible representation of dense 3D geometry are meshes. The Poisson surface reconstruction method [107] tries to fit a surface based on the gradient field imposed by samples and their normals. In contrast to other approaches, new vertices are created in the process of meshing and surfaces need to be of closed form. Thus, often undesirable bubbles are created and extensive postprocessing is necessary. In contrast, Delaunay triangulation [22] creates meshes which consist only of the initial sample points and intersection-free triangles in between. It can be shown that the memory usage for such a representation is linear to the number of points [7], and that the Delaunay triangulation contains a good approximation of the true surface if enough samples are provided [4]. However, a large number of inner triangles need to be removed if only the surface should be extracted.

In the following we will thus discuss state-of-the-art methods to densify sparse, keypoint-based geometry. While the Delaunay triangulation can even extract a surface from such a sparse point cloud, a densified set of points is clearly beneficial.

### 3.3.2 Densification using Patch-based Multi-View Stereo

The publicly available patch–based multiview stereo (PMVS) software of Furukawa and Ponce [65] is one of the most popular approaches to automatically reconstruct a scene when the camera positions are known. One of the main properties is that this approach densifies a point cloud directly in 3D space rather than estimating the depth of every pixel. PMVS initially estimates the position and orientation of small rectangular patches on the surface of the object based on sparse feature matching. Then, an expansion procedure iteratively adds patches in the neighborhood of already known regions. The approach successfully copes with outliers and thus is very robust, but the computational effort is also considerable. Since our structure from motion pipeline delivers sub-pixel accurate 3D camera orientations, PMVS can be run at full image resolution. However, even PMVS requires at least weakly textured surfaces for densification, thus a constant ground sampling distance cannot be guaranteed. The PMVS dense reconstruction result of an atrium scene is depicted in Figure 3.10(b).

(a) (b)

Figure 3.10: Densified atrium scene reconstruction. (a) Original view of the scene. (b) Semi-dense point model obtained by refining the sparse atrium model with PMVS [65].

### 3.3.3 Densification using Planesweep Stereo

In contrast to PMVS, multi-view planesweep stereo [28] densifies a point cloud in image space. While this might introduce noise in untextured regions, it is easier to apply smoothing and interpolation algorithms to a 2.5D representation than to a full 3D representation. The planesweep approach enables an elegant way for image based multi-view reconstruction since image rectification is not required. The method allows the reconstruction of depth maps from arbitrary collections of images and implicit aggregation of multiple view's matching costs. Furthermore, the method is well suited to run on current graphics processing units (GPUs) which makes processing of large datasets consisting of many high resolution images feasible [99].

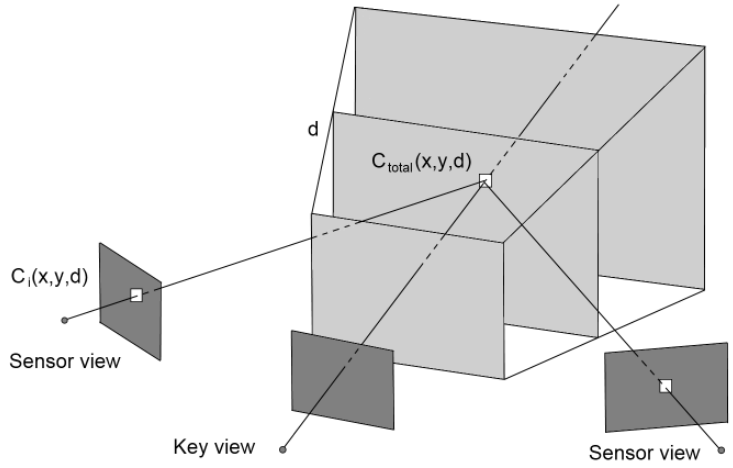



Figure 3.11: Principle of planesweep stereo. Given the camera orientations as a result of the SfM reconstruction, the multi-view planesweep approach [28] is used to generate dense 3D reconstructions.

In more detail, the scene is iteratively traversed by parallel planes aligned with a reference view and positioned at an arbitrary number of discrete depths, as shown in Figure 3.11. Each sensor view $i$ is then projected onto the current 3D key plane $\Pi_d = (n^\top, d)$ at depth $d$ according to the epipolar geometry. The mapping is given by a homography $H(d)$ which can be computed directly from the camera matrices $K$ and $K'$, the relative rotation $R$ and translation $t$ between key and sensor camera, the normal vector $n^\top$ of the plane and the current depth $d$ as

$$H_i(d) = K_i' \left( R_i - \frac{t_i n^\top}{d} \right) K. \tag{3.3}$$

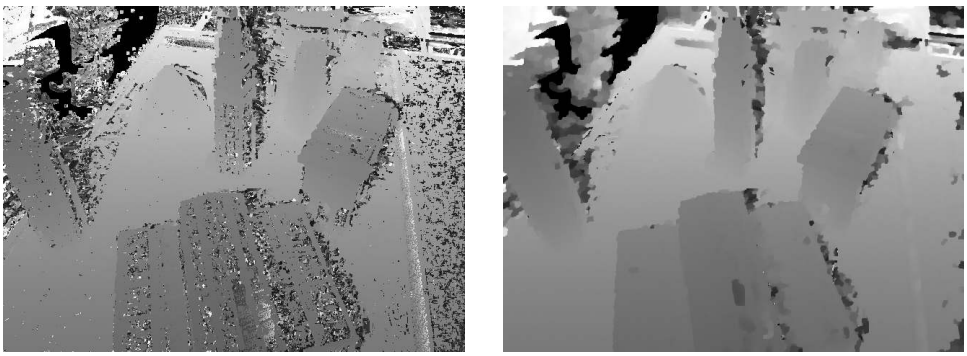
The plane sweep technique is based on the idea that if the plane at a certain depth passes exactly through the object's surface we want to reconstruct, then the appearance of the image points in the key view should match those of the projected sensor image under constant brightness conditions. For each discrete depth $d$, the sensor images are projected onto the plane and the similarity $C_i(x, y, d)$ between pixels of the key view and the sensor view $i$ are calculated as the zero-mean normalized cross correlation (ZNCC). Once the cost volume $C_{total}$ is filled with the accumulated matching scores, we extract final depth values from the volume by a simple winner-takes-all (WTA) [179] strategy in order to achieve high performance and low memory requirements. We select the depth where the accumulated cost has its minimum along the line of sight through the volume. Additionally, we discard depth values with a correlation value below a threshold $C_{min}$ in order to get the most reliable depth hypotheses only. The raw depth map can be seen in Figure 3.12(a).

The output of the planesweep algorithm is typically noisy and contains outliers as well

as areas with missing data. To improve the result, raw depth-maps are smoothed using a total variation (TV-L1) based image denoising model denoted by

$$\min_{u_d} \left\{ \int_{\Omega_d} |\nabla u_d| dx + \int_{\Omega_d} \lambda_d |u_d - f_d| dx \right\}. \tag{3.4}$$

Here, $u_d$ is the unknown denoised image, $f_d$ is the input and $\Omega_d \subseteq \mathbb{R}^2$ is the image domain. Using total variation as regularizer has the desired property of preserving edges while smoothing the flat regions. The parameter $\lambda_d$ controls the amount of smoothing. The resulting depth map can be seen in Figure 3.12(b).



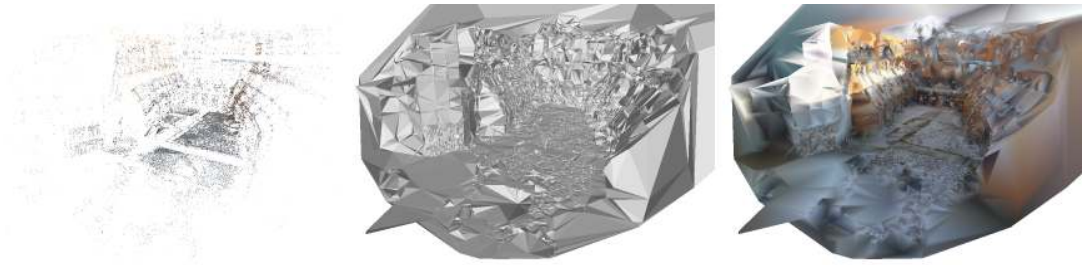
(a) (b)

Figure 3.12: Densification using Planesweep Stereo. (a) Raw depth–map before denoising. (b) Depth–map after TV-L1 denoising ($\lambda_d = 0.5$). Flat depth regions are smoothed, discontinuities are preserved.

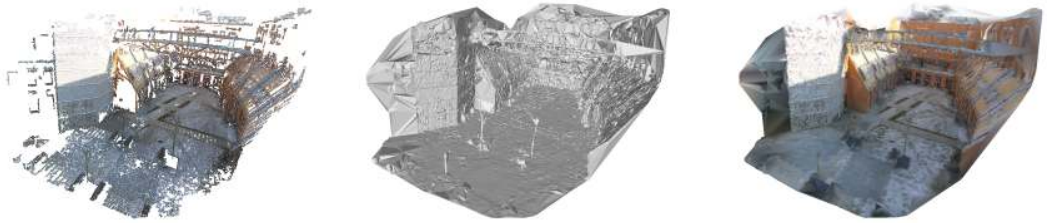
### 3.3.4 Mesh-based Surface Extraction

As discussed before, watertight 3D models can be obtained by performing a 3D Delaunay triangulation [22] followed by probabilistic space carving based on visibility information as initially proposed by Pan et al. [158]. Specifically, we employ the approach of Labatut et al. [124] who minimize an energy consisting of two terms to model the surface of objects: First, the visibility-based energy term votes for tetrahedra as being part of the object or part of free space, and it penalizes the number of conflicts of the line of sight with triangles crossing the surface. The second term takes care of the smoothness of the surface. It geometrically constrains the shape of a tetrahedra by penalizing the surface area, and by constraining the size of the circumsphere of the two adjacent tetrahedra of a triangle. This removes for example tiny elongated facets which are not desired in the resulting mesh. Finally, this energy functional is optimized via graph cuts [23].

The approach is very robust against outliers, and it preserves edges and fine structures



(a)



(b)

Figure 3.13: Mesh-based surface extraction based on the approach of [124]. From left to right, the point cloud, the surface mesh, and a colored mesh are visualized, based on (a) the SfM point cloud (36.948 points), and (b) the densified point cloud (318.540 points). The densification procedure clearly supports the extraction of the true surface and leads to visually appealing results.

while the result is at the same time free of self-intersections. An advantage over volumetric methods is that the level-of-detail problem is solved inherently. Parts of the surface that are sampled more densely by 3D points are represented with a higher resolution than weakly sampled parts, whereas the voxel representation has a uniform resolution. However, the accuracy of the surface estimation depends on the number of 3D points in the mesh, as triangles between points can only linearly interpolate the surface. Therefore, prior densification using PMVS or planesweep stereo as mentioned above is still very beneficial to increase the level of detail in the model. The meshing results given a sparse point cloud versus a densified point cloud are compared in Figure 3.13.

### 3.3.5 Volumetric Surface Extraction

While mesh-based representations can handle individual noisy measurements in an offline reconstruction very well, they are currently not suitable to accumulate knowledge regarding a certain space over time in an online fashion. We thus employ a volumetric

representation of geometry using a truncated signed distance function (TSDF) in such cases. Volumetric approaches allow solving for arbitrary 3D geometry, i.e. there is no constraint concerning the genus of the surface topology.

Our reconstruction algorithm is based on the method of [234, 235]. Using the signed distance formulation, the 3D surface is represented implicitly as the zero level-set of a function $u : \Omega \rightarrow [-1, 1]$, where $\Omega$ is a subset of $\mathbb{R}^3$. However, instead of storing the volumetric representation of $f$ directly, the truncated signed distance function $[-1, 1]$ is sampled at evenly spaced discrete positions $d_i$, $i = 1 \dots N$ and the exact value of every voxel of $f$ is replaced by the nearest $d_i$. Thus, we can store an arbitrary number of distance fields $f$ with constant memory footprint using a *volume histogram* consisting of $N$ bins. Every voxel $x$ has an associated histogram $h(x)$ which approximates the probability density function of the truncated signed distance function, as depicted in Figure 3.14. $h(x, i)$ denotes the histogram count of bin $i$, thus listing how often the value $d_i$ occurred in the distance fields $f$ at voxel $x$.

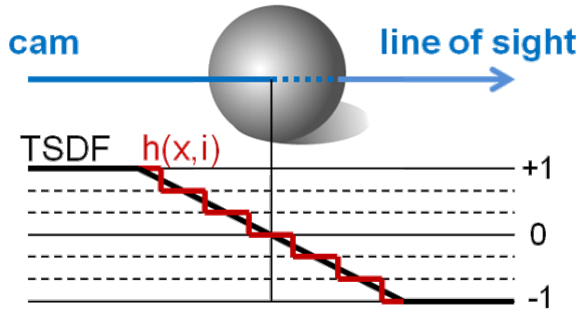
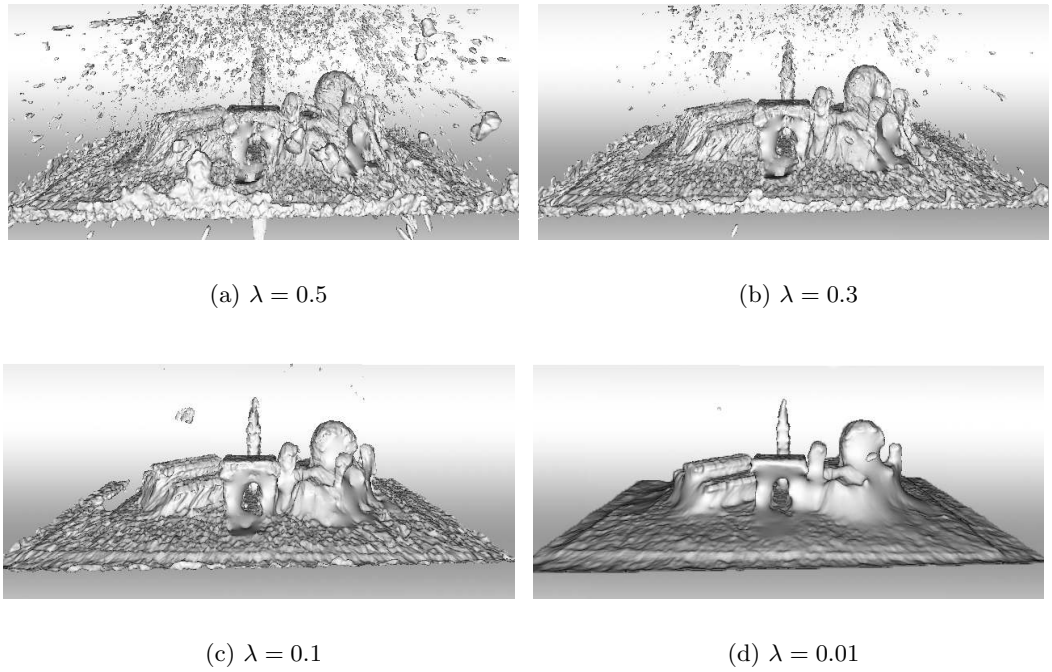



Figure 3.14: Quantized Truncated Signed Distance Function. Instead of storing the probability of being part of an object, every voxel $x$ stores the signed distance to the surface. Positive values indicate free space, negative values occupied space. The distance is truncated and then quantized in a histogram, which allows accumulation of several depth estimates. After smoothing in 3D, the surface can be extracted as the zero level-set.

Individual depth maps are fused together by minimizing the convex energy functional

$$\min_u \left\{ \int_\Omega |\nabla u| dx + \lambda \sum_{i=1}^{N} \int_\Omega h(x, i) |u(x) - d_i| dx \right\}. \tag{3.5}$$

The regularization term $\int_\Omega |\nabla u| dx$ measures the total variation of the function $u$. It minimizes the surface area of the level sets and hence effectively removes noise caused by the outliers of the depth maps while simultaneously computing a minimal surface approximation in areas with missing depth data (see Figure 3.15). Note that the total variation independently minimizes the surface area of each level set of the function $u$, that is, the surface of the final 3D model. The data term measures the $\ell_1$ distance of the



(a) $\lambda = 0.5$ (b) $\lambda = 0.3$

(c) $\lambda = 0.1$ (d) $\lambda = 0.01$

Figure 3.15: Variational noise removal on the dense, watertight surface helps to cope with inaccurate or incomplete depth-maps. From (a) to (d), smoothing is increased. The images show a scene where considerable amounts of data are missing; however, the fusion algorithm can successfully gap the holes in the reconstruction.

solution to the individual distance fields. Using the $\ell_1$ norm is more robust than a squared $\ell_2$ norm, but on the other hand it is more difficult to minimize. For optimization, we thus employ a globally optimal primal–dual approach as proposed by Graber et al. [70].

### 3.3.6 Summary

We have reviewed methods for the densification of sparse point clouds and for watertight surface extraction. On the one hand, we employ a state-of-the-art patch-based multi-view stereo method and a space carving-based meshing approach for offline operation. On the other hand, we propose to fuse depthmaps obtained with planesweep stereo using a volumetric representation and a variational optimization framework when updating dense surfaces in an online manner.

Dense models contain a lot more information than sparse models, but it is also much harder and computationally more expensive to obtain accurate dense models. We thus rely on sparse point clouds as basic representation of the environment and use dense geometry only for certain applications, including alignment, occlusion handling, collision avoidance for path planning, and visualization. Additionally, dense matching techniques are required to build large-scale digital surface models as presented in the next section.

## 3.4 Digital Surface Model Reconstruction

A digital surface model (DSM) is a 2.5D height representation of the Earth's surface and all objects on it as seen from an orthogonal aerial viewpoint. In contrast to a full 3D model, just the maximum height over ground is considered, rather than looking at all structures that might be below. The creation of a DSM typically requires a set of aerial images acquired by a plane or a micro aerial vehicle. After recovering the sparse geometry and camera orientations using Structure from Motion techniques, dense 3D reconstructions are generated and fused into a large-scale DSM. In this work, digital surface models are used for the global localization of point clouds and for high-level path planning, thus our approach for reconstructing detailed and accurate DSMs [175] is described in the following.

### 3.4.1 Related Work

Aerial images produced by modern large format camera systems comprise up to 250 Mpx at high radiometric resolution of more than 12 bits per pixel. Typical airborne photogrammetric surveys are flown with at least 80% forward overlap and 60% side-lap. Flying at an altitude of 1000 m, a ground sampling distance (GSD) of 8-10 cm/pixel is obtained. While DSMs can also be obtained using modern tools such as fixed-wing Micro Aerial Vehicles as shown below, classical aerial images provide the advantage that they cover a very large area on the ground and thus the resulting reconstructions are inherently geometrically consistent. Further, the large distance between camera and objects allows certain assumptions such as piecewise planarity.

Many large-scale 3D reconstruction techniques compute a digital surface model by merging depth hypotheses from multiple, pairwise estimated range images *a posteriori*, meaning that one depth map is estimated from all possible stereo pairs [85, 207]. The fusion step is important to create a reliable fused 3D point cloud and a dense, watertight surface mesh, but it is influenced by accumulated registration errors of the cameras and noise from false depth estimates. These erroneous depth estimates and outliers usually lead to a poor integration and consequently, need to be handled robustly.

Methods for range image integration in related work operate on different representations. They can be roughly categorized into mesh-based [205], volumetric [34, 83, 229] and point-based methods [66, 207]. We have presented a variety of such generic methods for surface extraction in the previous section. While these methods are suitable as well, the constraint of a common ground plane and a 2.5D representation allows to decrease memory consumption and increase runtime performance for large scenes.

### 3.4.2 Probabilistic Range Image Fusion

For range image estimation, we first employ the plane sweep approach [28] as presented in the previous section. While this methods exploits the approximately ortho-parallel nature of aerial images, the quality of the individual range images still depends heavily on the

input data. Hence, the resulting depth maps may contain noise and gross outliers due to erroneous depth values. These outliers can originate from image noise, illumination changes and shadows, as well as from occlusions, repetitive structures or uniform image regions with insufficient texture, which usually lead to a poor integration and consequently, need to be handled robustly.

Often, the majority of outliers can be removed by considering neighborhood information during depth extraction. For this purpose, state-of-the-art global optimization methods such as the variational optimization problem in 3D voxel space as proposed by Irschara et al. [99] can be applied. The individual range images created by such methods are in general smooth and visually pleasing. However, there is a trade-off between putting large effort in generating clean, outlier-free range maps using global optimization methods, and fast depth estimation using local methods which permit a certain amount of outliers in the data but preserve detail. Once the input data is smoothed it may be fused using a simple method like median fusion, but with the drawback of not being able to recover fine detail.

DSMs are meant to serve as the global layer of our pipeline, thus we focus on large-scale processing of range images and use a much faster local cost volume filtering approach [169]. Concerning runtime, this approach for depth extraction is 25 times faster than the global method. Although the depth maps from local cost volume filtering contain outliers, more details are preserved in the final DSMs, which is important for our purpose of MAV collision avoidance.

Outliers can finally be removed by exploiting the redundancy in aerial imagery. For fusing individual depth maps, we use our probabilistically motivated 3D filtering scheme for range image integration as proposed in [175]. The approach works in object-space, thus we do not need to select a reference view and the fusion scales very well to large sets of images. Furthermore, the method has the advantage that it avoids a volumetric voxel representation and is able to select a final depth from continuous values in object-space instead of a discrete voxel sampling (Figure 3.16). The method is memory efficient and can easily be parallelized for each individual DSM grid cell. Moreover, it is able to integrate point clouds from arbitrary sources into one consistent DSM; this allows to exploit existing airborne LiDAR data which has traditionally been used for surface modeling [127].

An example for a resulting high-quality DSM is visualized in Figure 3.17, and quantitative and qualitative comparisons to other methods can be found in Section 6.1.4.

### 3.4.3 Exploiting Publicly Available Geographic Data Sources

Neither high-quality aerial images nor the final digital surface models may not be accessible for everyone, and they may not be available for some locations on Earth. However, web–based map providers and community projects [68, 206] provide fairly accurate information not only regarding roads and directions, but also concerning buildings and vegetation. This kind of publicly available GIS data can be used to estimate a digital surface model

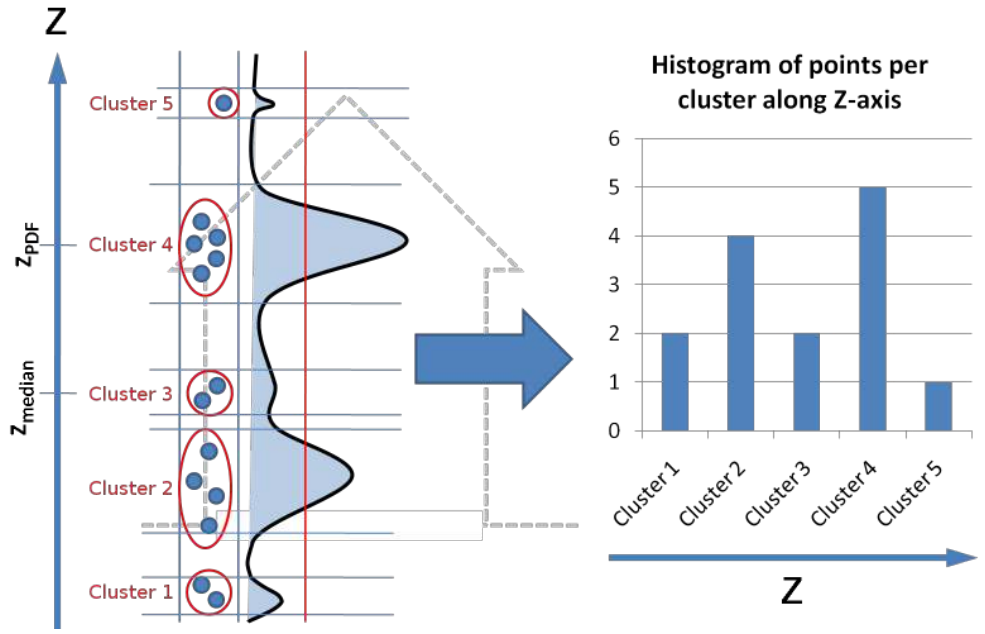


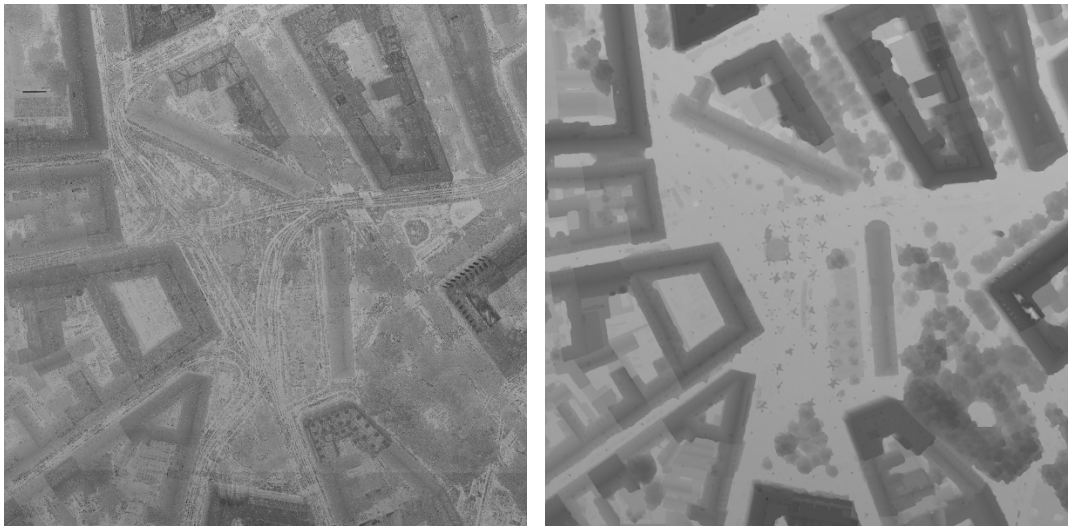

Figure 3.16: Probabilistic range image fusion. Depth hypotheses are considered as samples from an underlying probability density function, where local maxima of the density function correspond to dense regions in feature space [175]. The local maxima correspond to point clusters with higher density. Median fusion of all depth hypotheses would lead to a wrong height value whereas the probabilistic approach is able to select a height value correctly.

which is of low-quality, but still provides important prior information about the world. The process and a typical result are depicted in Figure 3.18.

First, we use terrain elevation data of the NASA Shuttle Radar Topography Mission (SRTM) [55] to generate a digital terrain model (DTM). SRTM obtained elevation data of the Earth at near-global scale, covering about 80% of the Earth's total landmass, and producing the most complete and high-resolution digital topographic database of the Earth. The dataset is available at 1 arc-second resolution (SRTM-1, approximately 30 meters) over the United States and its territories and at 3 arc-second resolution (SRTM-3, approximately 90 meters) for the rest of the world. To construct a triangle mesh from the unconnected sample points, we perform a 2D Delaunay triangulation and afterwards assign the corresponding height to each inserted vertex. Data voids present in the raw SRTM data due to shadowing, topographic variations or other radar-specific causes need no further processing, as the missing areas are implicitly interpolated through the mesh triangles of adjacent valid data points.

Then, we employ map data obtained from OpenStreetMaps[1] for extruding buildings. Semantical meanings in the map are connected to certain tags in the data vectors, so

[1] http://www.openstreetmap.org



(a) (b)

Figure 3.17: DSM reconstruction results. (a) The planesweep method combined with a winner-takes-all depth estimation strategy produces detailed but noisy results. (b) Combined with our probabilistic fusion approach [175] clean and accurate digital surface models are created.

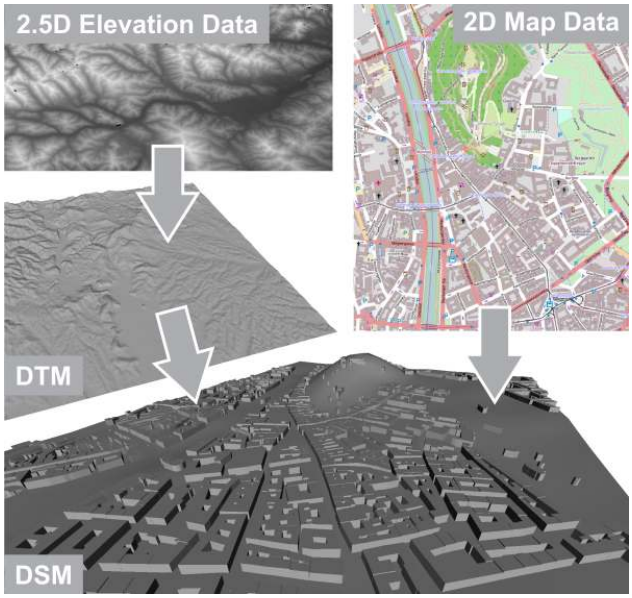



Figure 3.18: DSM estimation from publicly available geographic data. Terrain elevations and buildings extracted from map data are used to approximate a geo–referenced digital surface model.

it is straightforward to approximate a rough DSM by extracting the building polygon, projecting it onto the terrain model, and assigning a single, fixed height over ground to all buildings. While the resulting building heights are obviously not correct, they are good enough for several applications including occlusion handling during localization, large-scale rough path planning, or giving context to certain models for visualization.

### 3.4.4 Summary

We have shown a process to create digital surface models based on plane sweep stereo [28], cost volume filtering [169], and a probabilistic method for range image integration [175]. In contrast to existing methods, our approach scales very well because it avoids a volumetric voxel representation for fusion, and it is able to select a final depth from continuous values instead of the discrete voxel values delivered by the input range images. Additionally, we demonstrated how a rough DSM can be estimated from publicly available data [139, 174]. In Section 4.1 we will show how DSMs can be used to geo-reference and fuse SfM point clouds, which is the first step of our localization pipeline.

## 3.5 Optimized Acquisition Strategies for Prior Modeling

Geometric prior knowledge sometimes needs to be acquired under certain illumination conditions, in remote locations, or from aerial viewpoints. Especially in these situations it would be cumbersome if reconstruction fails, hence it is important to guarantee that the images taken on site are sufficient for the application of Structure from Motion and the respective densification steps. As the quality and completeness of a reconstructed model largely depends on the image quality and on the chosen acquisition strategy, we further investigate this topic.

To recover a detailed and complete 3D model, the SfM process has several requirements on the input images: The viewing angle between two images may not be too large to allow feature matching, the view cones must overlap, the images have to be textured but the texture should not be repetitive, and lighting should not change too much between images. For a typical user it is impossible to estimate if the acquired images fulfill all demands. Another difficult question is scene completeness, i.e. the coverage of the scene. Parts of the scene that are not captured with sufficient image overlap cannot be reconstructed. Since completeness depends on the required reconstruction resolution and on the surface itself, it is not possible to quantify the degree of completeness without prior information. In contrast, for a human it is relatively easy to answer this question by comparing a 3D model to the real world. In this section, we thus describe quality measures for planning suitable views automatically [88], as well as quality measures designed for guiding users during the image acquisition process [87].

### 3.5.1 Related Work

The problem of determining good imaging positions is known in different areas of computer vision [64, 188], photogrammetry [26, 60, 157], and computer graphics [212]. In robotics, the sensor placement problem is known as Next-Best View (NBV) problem [204]. Current methods aim to perform scene reconstruction and sensor placement simultaneously, hence they do not need prior knowledge about the scene. However, these algorithms are iterative and rely on high-framerate input so they require powerful processing units for real-time processing.

Only few methods exploit NBV strategies and prior knowledge to calculate a path offline. Dunn et al. [48] propose an algorithm that combines accuracy-driven NBV and path planning methods on given textured 3D models. Their objective function takes texture properties, uncertainty minimization, and the number of triangles visible from a certain viewpoint into account. As the texture is often not available when planning, the work of Wenhardt et al. [226] proposes a purely accuracy-driven NBV planning. In their work, the uncertainty of 3D points is evaluated based on the point covariance matrix [13]. We use their uncertainty measure, but in our approach we do not need to perform exhaustive search over all sampled camera positions.

### 3.5.2 Quality Measures for Planning Suitable Views

Automatic planning of "ideal" viewpoints for 3D reconstruction can be seen as a chicken-and-egg problem. To find good viewpoints, the 3D geometry has to be known in advance. On the other hand, a reconstruction is necessary to estimate the underlying 3D geometry. To circumvent this problem, our goal is to find a small set of cameras that covers the whole object and allows an accurate 3D reconstruction. In practice, however, it is more important to guarantee a sufficient reconstruction of the scene after a single flight session than to reach the smallest possible number of images.

Our approach requires a rough surface description of the desired object given as a triangular mesh, which allows us to perform visibility tests and to find self-occlusions. For example, this mesh can be obtained from a DSM or from previous reconstructions of the object. If both are not available, a few images from a nadir view acquired by an MAV can be used to construct a rough model of the object. Based on this prior knowledge, our algorithm computes viewpoints while taking three constraints into account. First, we define an upper bound of a reconstructed point's uncertainty. Given a large number of cameras, we want to find a subset that triangulates a 3D point with a given accuracy. Second, we want to exceed a minimum overlap percentage between image pairs. Third, all parts of the surface have to be covered by the acquired images. Our solution consists of three main steps, which are described in detail in [88] and summarized below.

First, during *camera placement*, we initialize our search space with a discretized surface given as a triangular mesh and create a very large number of candidate cameras observing the mesh from a fronto-parallel viewpoint. The distance to the mesh depends on the desired

ground sampling distance $s$ and the camera's focal length $f$. Then, *camera clustering* reduces this search space by grouping candidate cameras based on their viewing angle on certain surface triangles. As shown by Beder et al. [13], the depth uncertainty of a reconstructed point depends mainly on the triangulation angle between the intersecting rays, thus cameras with similar viewing angles can be removed. This constraint reduces the search space drastically, while it still guarantees a certain triangulation angle which can be seen as the lower bound of the reconstruction accuracy. Clustering also allows us to define the importance of a camera for the reconstruction process, as cameras which cover many scene points are usually more important for the overall reconstruction. Finally, during *view selection* we incrementally select camera positions according to our quality criterion [88, 226] while maintaining side constraints like image overlap and maximum angle between view points. Due to the low computational complexity, our approach is able to handle large, complex models that require several hundred camera positions to be reconstructed accurately.

The output of our photogrammetric network design algorithm is a small set of views which satisfies the quality measures for reconstruction uncertainty, overlap, and coverage of the scene. When all of these constraints are fulfilled, a successful 3D reconstruction can be expected. In Section 5.1 we will further tackle this topic by adding a path planner to the approach, making it suitable for autonomous image acquisition.

### 3.5.3 Quality Measures for Guiding Users during Acquisition

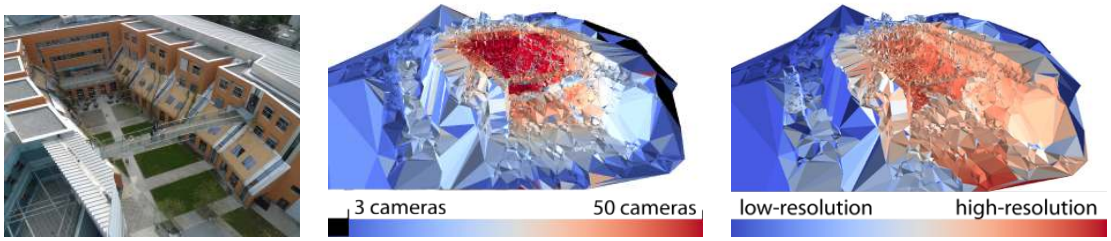
When prior knowledge is not available or when automated methods cannot be applied, the user has to manually acquire the images. Without feedback it is even for a very experienced user difficult to determine if a large-scale scene is sampled sufficiently, i.e. if all relevant parts have been captured. Although the SfM point cloud already indicates the scene completeness, it is difficult for a user to derive this information from the point cloud. Therefore, we employ the approach of Labatut et al. [124], as presented previously, to extract a surface mesh given the sparse SfM point cloud. Based on the extracted surface model, we derive two measures to support the user during the acquisition process: The ground sampling distance (GSD) and the degree of redundancy.

The ground sampling distance measures the resolution of the mapping between the 3D physical surface to 2D image space. Therefore, the GSD determines the (theoretical) maximum resolution of a dense 3D model. To compute the quality measure, we record the maximum resolution a mesh triangle is mapped to in image space. We reproject each triangle $T_i$ of the mesh $S$ to each aligned camera $C_t$. We then calculate the maximum resolution

$$R(T_i) = \min_{C_t} \sqrt{\frac{A(T_i)}{P(T_i, C_t)}} \tag{3.6}$$

which corresponds to the minimum value of the GSD, where $P(\cdot, \cdot)$ is the number of pixels that triangle $T_i$ covers in camera $C_t$ and $A(\cdot)$ is the area of the triangle in 3D space.

Our second measure, the degree of redundancy, describes how often a physical surface area is captured by images. Redundancy is required to achieve accurate results but it also increases the computational costs and lengthens the acquisition. Especially in large-scale and geometrically complex scenes, the visualization of the redundancy supports the user to obtain complete scene sampling. The degree of redundancy can be computed at the same time as the GSD by counting the number of cameras $T_i$ is visible in. We define that $T_i$ is visible in $C_t$ if less than 50% of $T_i$'s area is occluded. This prevents largely occluded triangles from being counted as visible.




(a) (b) (c)

Figure 3.19: Quality measures for guiding users during acquisition. (a) Sample image from the scene acquired by an MAV. (b) Mesh surface based on the online SfM reconstruction, overlaid by redundancy information. (c) The visualization of the GSD supports the user to recognize which parts of the scene have been sampled at which resolution.

To visualize both measures, we overlay the mesh by a color map according to the measure's value. The user interactively selects which information he requires for the decision in the next step. Since the scale of the SfM result is arbitrary, $A(\cdot)$ is typically not in metric scale and we determine the range of the color map by $\alpha$-trimming all values of $R(T_i)$, where $\alpha = 10\%$. If the scale of the reconstruction can be determined, for example by roughly aligning the reconstruction using GPS data, we can choose the color map according to predefined resolutions. Figure 3.19 visualizes the two quality measures for the atrium scene.

### 3.5.4 Summary

We have presented quality measures which support users during the difficult task of proper image acquisition for Structure from Motion reconstruction. We discussed the visualization of the expected redundancy and resolution of a reconstruction [87], and we have shown how to select suitable views automatically based on prior knowledge about the geometry [88]. Especially when taking pictures in remote locations, under certain illumination conditions, or from aerial viewpoints, it is cumbersome to repeat image acquisition once the model

has been reconstructed, thus optimized acquisition strategies are crucial for successful 3D reconstruction.

In the next chapter, we will demonstrate how the reconstructed priors can be aligned in a global coordinate system based on geometry only, thus tolerating different ground sampling distances and different appearance. The priors can then be used for robust localization under various environment conditions, even if the individual models cannot be directly fused by Structure from Motion techniques.

# Chapter 4

# Multi-Scale Distributed Visual Localization and Mapping

**Contents**



A large number of interesting applications for micro aerial vehicles such as surveillance, agriculture, inspection, or simply 3D reconstruction for visualization share one common feature: The need for highly accurate localization of the vehicle with respect to the scene. According to the work of Konolige and Bowman [120] on lifelong maps, long–term visual localization requires an accurate map with good features, but should also allow incremental mapping and recovery from localization failures. An additional requirement for MAVs is the possibility to switch between GPS and visual localization, as visual maps are unnecessary when flying high above buildings or in wide open spaces.

In the previous chapter we have shown how to model geometric prior knowledge by taking pictures with a consumer camera from eye–level above ground, by using micro aerial vehicles as acquisition tools in low altitudes, or by using large-format digital cameras in higher altitudes. In this chapter, we present a multi-scale approach for monocular localization with an accuracy comparable to differential GPS in a global coordinate system (Figure 4.1). Further, we introduce an incremental feature update step which allows to fuse in–flight information back into the original scene model, and we show how the complex computations can be executed in real-time by distributing the load between a mobile robot and a computationally more capable server. We conclude with a model-based tracking method which builds on visual SLAM but incorporates line-based priors; an approach that is especially useful when localizing relative to known objects.

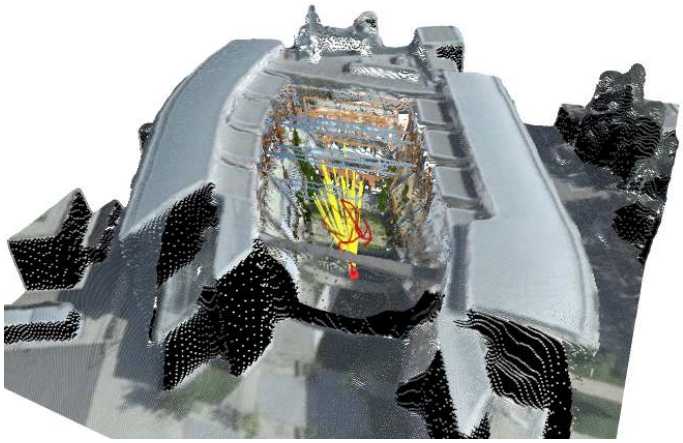


Figure 4.1: Localization in a geo–referenced, metric visual landmark allows to convert the resulting local ECEF coordinates back to a GPS position. It is therefore possible to seamlessly switch between GPS and visual navigation, as well as to exploit onboard GPS and IMU data in the localization process.

## 4.1 Global Alignment of Geometric Priors

Alignment of visual maps in a global coordinate system is important for several reasons: First, it facilitates the incorporation of additional sensors at flight time and for evaluation, as coordinates obtained by visual localization are directly comparable to IMU and GPS readings. This also allows an MAV to switch seamlessly between GPS waypoint mode and visual localization, which is very important when connecting different visual maps, and it allows to represent point clouds with metric scaling. Second, proper alignment is beneficial to applications where the model should be set into context, for instance in industrial applications such as construction site monitoring [117], or whenever further algorithmic steps depend on it as in automatic view planning [88, 183].

However, two major challenges have to be tackled. On the one hand, geo-referencing of 3D reconstructions is often difficult because of shadowed GPS signals in urban areas, so accurate alignment purely based on GPS information is not possible. On the other hand, flight space might be restricted in urban areas, which leads to missing views for accurate 3D reconstruction and causes fracturing of large models. This could also happen due to vegetation or simply a change of illumination during image acquisition. For the resulting fractured models, not enough visual feature correspondences for fusing the models can be established.

In this section, we therefore address the automatic alignment of such partial Structure from Motion (SfM) reconstructions to an overhead digital surface model (DSM) [219, 220] as depicted in Figure 4.2. Our approach is to get an initial rough estimate of the model's pose using noisy GPS locations, and subsequently refine the pose based on a 2D correlation of the DSM height map with a projected model height map.

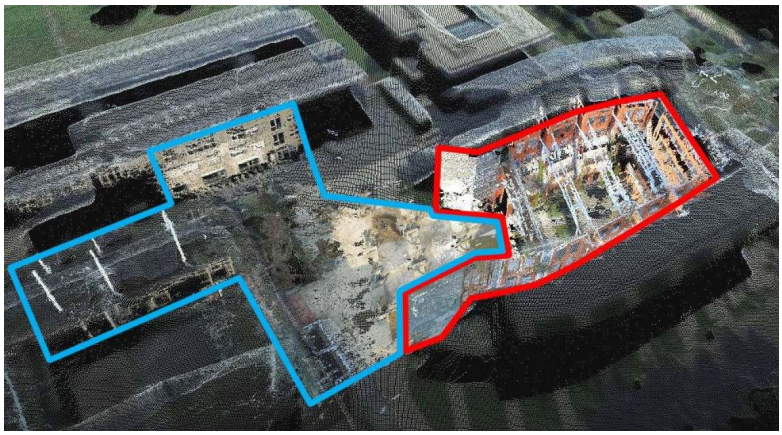


Figure 4.2: Automatic alignment of two semi-dense 3D point clouds to an overhead, textured Digital Surface Model (DSM). The atrium model (red) has been reconstructed from ground-level, whereas the images for the campus model (blue) originate from a micro aerial vehicle.

### 4.1.1 Related Work

The problem of aligning 2D images or 3D models to a 3D structure is well studied, especially in the context of large-scale city modeling. Früh and Zakhor [62] present an algorithm to fuse close–range facade models acquired at ground level with a far–range DSM recorded by a plane. The models are created using both ground-based and airborne laser scanners, as well as digital cameras for texturing. Their approach is based on registering the edges of the DSM image to the horizontal scans of a ground model using Monte-Carlo-Localization. Similarly, Strecha et al. [195] register facades segmented from a 3D point cloud to building footprints. Their approach combines various visual and geographical cues in a generative model, which allows robust treatment of outliers. However, both approaches are focused on large-scale city models with flat facades to both sides, resulting in fairly clean edges. In contrast, our approach takes the height over ground into account and therefore even benefits from complex structures.

A popular approach to aligning and fusing SfM point clouds is to use random sample consensus (RANSAC)-based geometric verification [58]. A typical issue is the estimation of a reasonable inlier threshold, however this has been resolved in recent work [164]. Still, such an approach is not feasible for our purpose as on the one hand feature correspondences cannot be established and the algorithm would have to solve a huge combinatoric problem. On the other hand, we want to align data with significant variations of the ground sampling distance which would not be possible either.

Another well known method of aligning two point clouds is the Iterative Closest Points (ICP) algorithm [238]. ICP estimates a transformation to minimize the overall distance between points by iteratively assigning closest points as correspondences and solving for the best rigid transform. While ICP is mainly used for registering 3D laser scans, Zhao et

al. [240] use it to align dense motion stereo from videos to laser scan data. However, 3D ICP can take very long and suffers from getting stuck in local minima due to its typically small convergence radius. In other words, a very good initialization is necessary for ICP to converge. However, using a point-to-plane variant [134] the ICP method can still be exploited on top of our method to improve the results.

Kaminsky et al. [103] use 2D ICP to compute the optimal alignment of a sparse SfM point cloud to an overhead image using an objective function that matches 3D points to image edges. Additionally, the objective function contains free space constraints which avoid an alignment to extraneous edges in the overhead image. While their approach is suitable to align many 3D models obtained from ground level, it has problems with points on the ground and would therefore fail to align the models acquired using our micro aerial vehicle. Typical models presented in their paper show vertical walls which can be easily projected to the ground and fitted to respective edges. However, when many extraneous edges are visible in the overhead image their method fails. In contrast, given a sufficient point density in the reconstruction, our approach is less prone to errors caused by objects on the ground, it implicitly follows a free-space constraint and it works with models covering a small area.

### 4.1.2 Automatic Alignment Pipeline

Our alignment pipeline is based on the registration of 2D depth maps showing the scene in a nadir view, rather than 3D point clouds. We can thus handle geometric configurations where ICP fails due to unknown correspondences or local minima, as well as significant differences in appearance because there is no need to establish sparse feature correspondences. We follow a three step approach: First, we convert all GPS and pixel coordinates into a local, metric coordinate system. Second, we roughly align the acquired 3D point cloud to the DSM by exploiting the available GPS information. Finally, we search for the best height map alignment between the DSM and the semi-dense model point cloud using normalized cross-correlation.

**Conversion to Local ECEF Coordinates.** Geo-referenced model coordinates are represented in a local Earth-centered, Earth-fixed (local ECEF) coordinate system. While the global ECEF coordinate system has its origin at the center of the Earth, with the $x$ axis passing through the equator at the prime meridian and the $z$ axis passing through the north pole, local ECEF employs a tangent plane to the Earth's surface at a reference point (Figure 4.3). By definition, the $x$ axis heads East, the $y$ axis North, and the $z$ axis up into the sky, thus it is also called *ENU* coordinate system. While the individual reference point needs to be known using this representation, the benefits include a more intuitive Cartesian system which allows to obtain metric measurements in the transformed point cloud, and a better numerical stability of the transformations. We choose the reference point to be the arithmetic center of the DSM's known corner points.

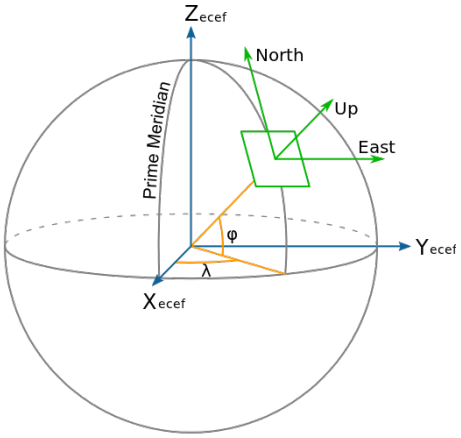
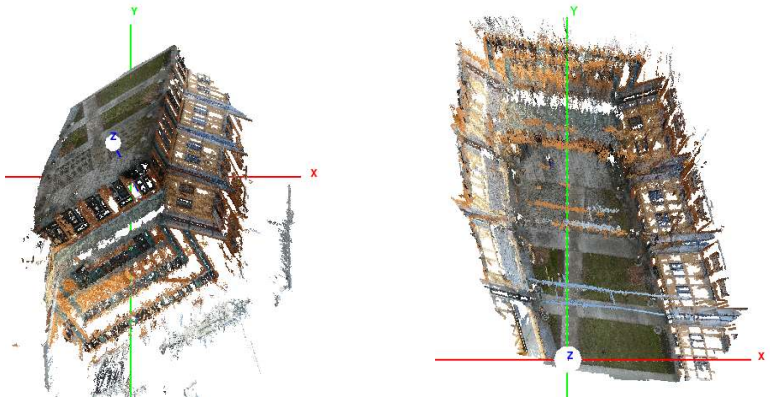



Figure 4.3: Relationship between GPS coordinates ($\lambda$, $\varphi$) and the Cartesian local ECEF coordinate system (east, north, up). A tangent plane to the Earth's surface at a reference point is established for a more intuitive and numerically stable representation.

**Rough Alignment using GPS Information.** We exploit the GPS information available for every camera in our 3D models for rough alignment to the DSM. If the coordinates measured by the GPS system were correct, it would be straightforward to estimate a 3D similarity transform. However, longitude and latitude are noisy, and especially the altitude estimate can be very inaccurate.

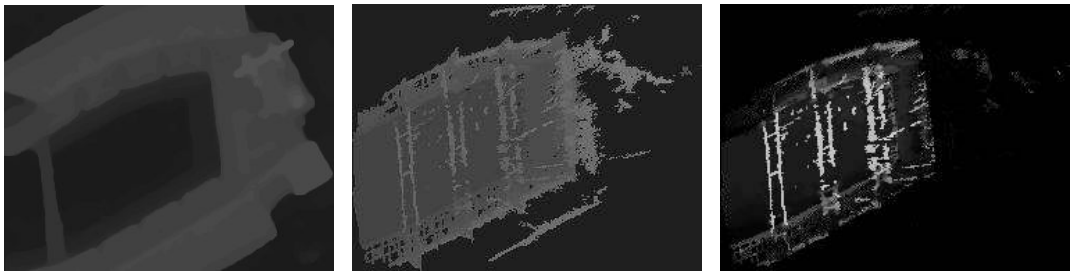
Therefore, our approach is to solve for a 2D similarity transform between the camera positions $(x_{sfm,i}, y_{sfm,i})$ in the SfM model and the GPS coordinates $(x_{gps,i}, y_{gps,i})$, both in local ECEF coordinates. Neglecting the altitude is just possible when the ground planes are aligned. However, SfM pipelines typically store resulting models in the coordinate system of the first camera. As a result, the axes of partial reconstructions do not align and have to be rotated to a common ground plane. We employ a reasonable assumption to approximate this plane, namely that the horizontal axis in every image coordinate system is approximately parallel to the ground plane, which is the case when taking upright photographs from the ground but also when taking nadir and oblique pictures on a micro aerial vehicle. The approach of Szeliski [198] can then be used to compute the ground plane normal and the corresponding rotation, as visualized in Figure 4.4. Finally, a robust 2D similarity transformation can be found using RANSAC [58].

**Precise Alignment using Correlation.** Given the rough alignment, we use correlation for precise alignment. We project the semi-dense 3D point cloud into the pixel grid of the DSM image, storing only the maximum height value per pixel. Pixel clusters with a radius $r \leq 2px$ are removed using morphological operations to get rid of reconstruction outliers. The model template $M_0$ is finally created by cropping an axis-aligned box containing the defined values. The template is visualized in Figure 4.5(b), with undefined values in black color.



(a) (b)

Figure 4.4: Rotation for ground plane alignment based on the method of Szeliski [198]. (a) SfM model in the coordinate system of the first model camera. (b) SfM model after performing the estimated rotation.



(a) Ground Height (b) Model Height (c) Difference

Figure 4.5: Precise Alignment using Correlation. The normalized height values $G$ of the ground (a) are compared to those of the projected SfM model $M_t$ (b), with undefined points marked in black. (c) Differences occur not only when a building height does not fit, but also when the ground height does not fit.

As the uncertainty of the rough alignment can introduce rotation and scale errors next to the translational uncertainty $\Delta T$, we rotate the model by the angles $\Delta\phi$, $\Delta\theta$, and $\Delta\psi$ (roll, pitch, yaw) and scale it with a factor $s = 1.0 \pm \Delta s$ to generate the model templates $M_t$. We cover the search space using the coarse–to–fine approach by [103] to speed up computation, and crop the ground template $G_{gsd}$ from the DSM image according to the pyramid level $gsd$ (Figure 4.5(a)).

The score of template $t$ is finally computed by normalized cross-correlation of ground

and model templates,

$$d(t) = \frac{1}{n_t - 1} \sum_{x,y} \frac{(G_{gsd}(x,y) - \overline{G_{gsd}})(M_t(x,y) - \overline{M_t})}{\sigma_{G_{gsd}} \sigma_{M_t}}, \tag{4.1}$$

where $n_t$ is the number of *defined* pixels for every template $t$, and $\overline{G_{gsd}}$, $\sigma_{G_{gsd}}$, $\overline{M_t}$ as well as $\sigma_{M_t}$ are computed only for *defined* pixels. Additionally, we introduce a term for penalizing alignments which contain a large amount of *undefined* pixels,

$$r(t) = \frac{n_t}{N_t}, \tag{4.2}$$

where $N_t$ is the number of all pixels in template $t$. The best height map alignment is then associated with the best model template

$$t_{best} = \arg\max_t \;\; d(t) + \lambda r(t). \tag{4.3}$$

The mode of the difference between the ground template and the best model template, and the ratio of the respective variances, can finally be used to estimate the translation and scaling on the vertical axis.

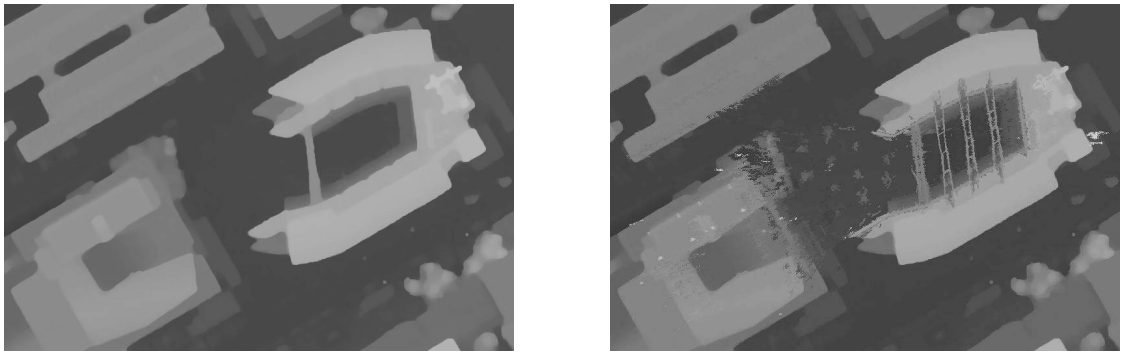
Using height maps for alignment can be seen as an implicit free space constraint, if compared to the work of Kaminsky et al. [103]. Differences between ground and model templates as in Figure 4.5(c) occur not only when a building height does not fit, but also when the ground height does not fit.

### 4.1.3 Benefits of Global Alignment

The alignment of 3D point clouds acquired at ground level or using airborne sensors such as a micro aerial vehicle has several benefits other than automatic georeferencing. In the following paragraphs we give an overview of how the gained information can be used for DSM refinement, for season-invariant matching, and for providing context in visualization.

**DSM Refinement.** One of the major benefits of alignment is the refinement of existing high-quality DSMs. In contrast to the original DSM in Figure 4.6(a), the refined DSM in Figure 4.6(b) shows details such as several antennas (left), small trees (middle), or steel wires (right). While these structures might not be important for building models, they are very important and dangerous for autonomous MAV flights. In Section 5.1.2 we demonstrate how to use the refined DSM for path planning and collision avoidance on our MAV.

**Fusing Reconstructions from Different Viewpoints.** Our method also bridges the gap between reconstructions computed from airborne and ground-level imagery. Figure 4.7 shows the partial reconstruction of a building that is obtained from nadir images taken by



(a) Original DSM (b) Refined DSM

Figure 4.6: DSM refinement based on alignment. In contrast to the original DSM (a), the refined DSM (b) shows details such as several antennas (left), small trees (middle), or steel wires (right).

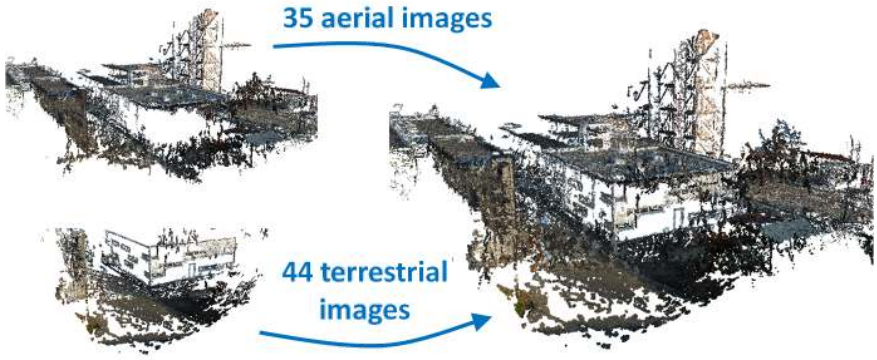



Figure 4.7: Fusing aerial and terrestrial reconstructions based on geometry only. The resulting model demonstrates that our method can merge reconstructions obtained from very different viewpoints and with highly varying GSD to an accurate 3D model.

our MAV at a height of 60 m, and thus does not show facade details. The reconstruction of the same building from ground-level images is shown next to it. Due to the weakly textured surfaces, the utilized densification algorithm does not perform very well and the GSD of the model varies. Even using such challenging data, our algorithm fuses both parts to a comprehensive model.

**Season-Invariant Matching.** Seasonal changes are a challenge for every appearance-based outdoor localization algorithm. Recent work of Valgren and Lilienthal [209] investigates the performance of local features such as SIFT [135] and SURF [12] in scenes which undergo such changes. In contrast, we propose to accurately align several models of

(a) (b)

Figure 4.8: Season-invariant matching. (a) Accurate alignment allows to collect local features in different seasons (blue, red) and fuse the overlapping point clouds. (b) The aligned models have been acquired in different seasons (summer and winter), as can be seen when looking at the vegetation. While the appearance differs, the alignment in a global coordinate system is successful.

the same locations, acquired at different days or in different seasons. The geo-registered feature clouds can then be used to perform accurate, season-invariant matching. The concept of virtual cameras, as presented in the next section, fully integrates this increased amount of descriptors during a localization task. A visualization of two overlapping and two adjacent models, taken in different seasons, can be found in Figure 4.8.

**Providing Context in Visualization.** Fitting the model into an environment can also be important for visualization. Figure 4.9(a) shows a construction site, aligned to a construction plan. The model has been aligned to the existing building next to it, using the proposed algorithm, and based on a DSM generated automatically from OpenStreetMap. However, it was necessary to manually align the construction plan to the DSM and to manually shift the model along the vertical axis due to the missing accurate building heights in the DSM. Figure 4.9(b) shows the alignment result with the estimated DSM. While the result looks good, more stable alignment in the environment of construction sites could be achieved by using multiple evolving models with less changes, or by including a significant part of the surrounding buildings into the reconstruction.

### 4.1.4 Summary

In this section, we have presented an algorithm for the automatic alignment of partial 3D reconstructions. Given an overhead Digital Surface Model (DSM) and approximate GPS tags for the individual camera positions, we refine the alignment based on the correlation of orthographic depth maps. We can handle complex cases where previous methods had

(a) (b)

Figure 4.9: Providing context in visualization. (a) A construction site aligned to a construction plan using an estimated DSM. (b) The estimated DSM can be enhanced by fusing the aligned model into it.

problems, including models which do not share any appearance features and models with considerably different ground sampling distances. This allows not only to fuse data acquired by an MAV, but also combining aerial and terrestrial data sources. The accuracy of our approach mainly depends on the ground sampling distance of the overhead digital surface model, which is in the range of 5-30 cm per pixel for modern cameras.

## 4.2 Point-based Localization using Geometric Priors

In the previous sections we have shown how to create geometric prior knowledge in form of 3D reconstructions, and how localization of such models on a global level can be achieved. In this section we introduce the concept of virtual views and demonstrate how it can be used to localize within metric, geo–referenced 3D priors. We present a method for monocular localization with an accuracy comparable to differential GPS in a global coordinate system. Further, we introduce an incremental feature update step which allows to fuse in–flight information back into the original scene model, and we exploit additional sensor information to ensure localization robustness.

### 4.2.1 Related Work

Model–based localization approaches have been extensively investigated over the past decade [133, 181, 237, 242]. Most closely related to our approach is the urban localization approach of Irschara et al. [101] which uses image retrieval-based methods for location recognition. Compared to this class of methods, considerable improvements have

recently been achieved [132, 176] using prioritized, direct matching of 3D points to 2D features. However, Sattler et al. [177] have investigated the performance gap between image retrieval-based methods and direct matching. It turns out that it is caused by a large number of incorrect votes cast by standard re-ranking schemes, and can be fixed using their proposed method. Instead, we employ a motion prior to select a small, feasible set of virtual views which is then ranked. The search space is thus much smaller and we are hardly affected by ranking issues. Apart from that, our approach using virtual views has many benefits over direct matching as described in the next section.

### 4.2.2 Scalable Localization using Virtual Cameras

Our framework provides a visual landmark in terms of a large geo–registered 3D point cloud with associated visual features. However, the information in the landmark is based on a a low number of fixed viewpoints and thus does not provide views which are sufficiently close to the images taken by an MAV. Additionally, matching descriptors of a query image against thousands of descriptors in the model is infeasible as it would result in a high number of non–unique matches, and would not scale in terms of computational effort. In contrast, the partitioning of the search space using virtual cameras [101, 221] enables efficient visual localization in outdoor environments. Combined with an incremental feature update step [222] to cope with missing viewing angles in the original model, a robust and scalable localization system is proposed.

**Placement of Virtual Cameras.** Virtual cameras hold the correspondences of only those 3D points and features which are visible given the respective camera center, camera orientation, and occlusion surfaces. Motivated by [101], who use synthetic views for fast location recognition in a flat environment, we have extended the concept to 3D space in [221]. We sample camera centers on a regular grid $G$, and camera vectors in uniform angular steps of $\gamma$ on a unit sphere. Additionally, we use the intrinsics of the camera mounted on the MAV to retrieve the correct scale of the features.

After setting up the necessary views, we project all 3D points into the respective cameras. A projected feature has to be in front of the camera and within the extent of the image, and it needs to have a scale larger than one pixel after projection. Additionally, we use the aligned digital surface model for view culling (Figure 4.10). We restrict the maximum distance of projected points to the distance of the closest surface in viewing direction; this approach avoids adding occluded features. For every 3D point $X$, we use the SIFT descriptor [135] with the smallest extraction angle

$$\delta_{min}(X,i) = \min_j \arccos\left(\frac{\boldsymbol{XV_i}}{\|XV_i\|} \cdot \frac{\boldsymbol{XE_j}}{\|XE_j\|}\right) \tag{4.4}$$

between the virtual camera $V_i$ and the model camera $E_j$. Consequently, $\delta_{min}(X,i)$ increases with any deviation from the camera positions used for acquisition, but especially

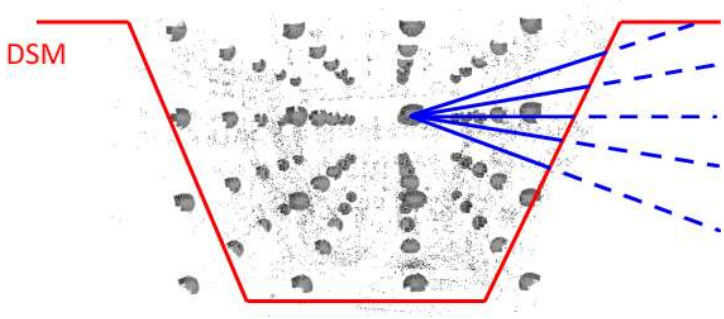


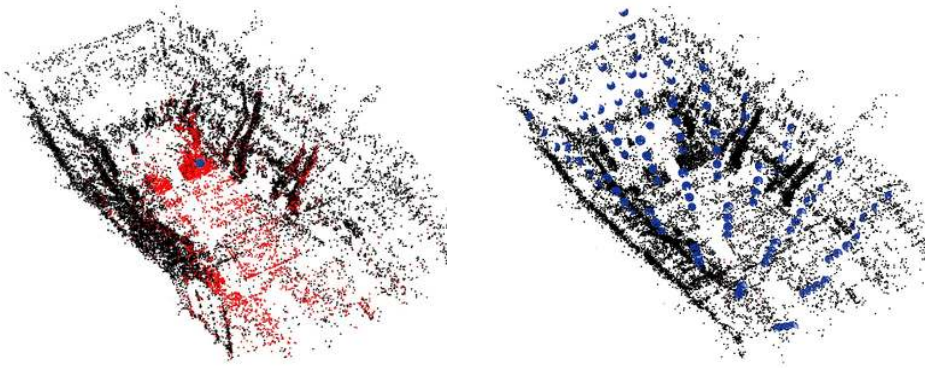

Figure 4.10: DSM view culling. When projecting 3D points into the respective virtual camera representations (gray view frustums), the aligned digital surface model is used to remove features which are likely to be occluded.

in higher altitudes when using images taken at ground level to create the model. This increase in extraction angle is visualized in Figure 4.13(a). However, according to Mikolajczyk et al. [145] SIFT is only robust to off–image–plane rotations up to $\delta_{min} < 30°$. Given for instance a distance between camera and point of 10 m, this would mean that the localization only works up to a height of 5 m above the acquisition height, because for higher altitudes matching fails and the pose cannot be estimated robustly. This problem can be circumvented by adding in–flight information to the model; our approach called *incremental feature updates* is described in detail in Section 4.2.3.

We determine the number of necessary virtual cameras based upon the 3D reconstruction of the scene. The extent and the alignment of the grid $G$ are defined by the 3D point cloud. To ensure overlapping coverage of the entire model, we increase the number of grid points along each axis until the number of feature points which are visible in less than $N_{coverage}$ cameras converges. During this process all views with less than $|F|_{min}$ features are removed. The placement of virtual cameras is visualized in Figure 4.11.

**Localization using Virtual Cameras.** The resulting virtual cameras $V$ provide a highly redundant representation of the scene. At first sight, the usage of synthetic views seems to cause an enormous need for feature matching. However, the approach scales very well for image sequences, as MAVs do not exceed a predetermined translational and rotational speed in normal operation. The feasible set of virtual cameras $V_t$ at a certain time $t$ is defined by the spatial distance to the previously estimated position and orientation of the camera. We define the translational and rotational search radii $t_{max}$ and $R_{max}$ around that position, which are enlarged in case the MAV cannot localize its position for a number of frames, for instance due to close–up views of an obstacle or unusual movements.

Virtual cameras within that search area are ranked based on appearance using a vocabulary tree [156] and corresponding inverted files. Restricting the search space by cascading prior knowledge and appearance features to the feasible set $V_t$ allows our approach to han-



(a) (b)

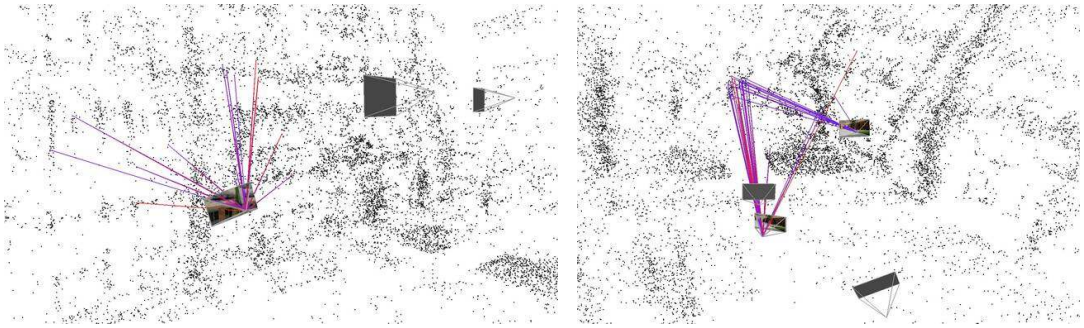
Figure 4.11: Placement of virtual cameras. (a) A placement with $G = \{1, 1, 1\}$ and $\gamma = 30°$ does not satisfy the required coverage of $N_{coverage} = 3$. Uncovered points are marked in red, virtual cameras in blue. (b) The coverage converges for $G = \{7, 4, 4\}$ and $\gamma = 30°$.

dle scenes with unlimited size, and successfully copes with repetitive structures. We sort $V_t$ according to the scores obtained by the vocabulary tree query, and subsequently use the top–scoring $k$ views $V_t^{1:k}$ to geometrically verify the pose of the MAV.

For geometric verification, SIFT features extracted in the query view are matched to those in the virtual view using the GPU and the distance ratio test (see Section 3.1). We then exploit the knowledge about the 2D to 3D correspondences of points in the virtual view, and estimate a robust absolute pose from three point correspondences (P3P) [118] using fast RANSAC methods [58]. Finally, only poses with a minimal number of effective inliers, $I_{eff} \geq th_{eff}$, are accepted. The measure of effective inliers [101] reports the spatial distribution of inliers within the image (Figure 3.2); a high value indicates that the inliers are well distributed and therefore more reliable. Figure 4.12 visualizes the top three virtual cameras $V_t^{1:3}$ and the corresponding estimated absolute poses $P_t$ for selected query images.

In summary, the main idea is to fill a scene with a large number of virtual cameras which each hold $N_{V_i}$ correspondences between visible 3D points and features, given the respective camera center and orientation. This overcomes the problem of matching all $N_{query}$ features from a query image to all features in the model. Instead, we match only $N_{query} \times N_{V_i} \times k$ features for a small, feasible subset of $k$ virtual cameras. As a result, the approach scales very well and it can cope with ambiguous environments.

Due to the previously discussed process of creating and aligning visual maps, the result of the localization process is a metrically correct position which can be converted back to GPS coordinates. The concept of virtual cameras supports the storage of feature descriptors from different acquisition days, thus it is possible to localize the camera in different seasons if the necessary data is available. It is also possible to update virtual cameras, as discussed in the next section.



(a) (b)

Figure 4.12: Localization initiated by virtual cameras. (a) In most cases, pose estimation based on the virtual cameras $V_t^{1:3}$ (visualized without texture) results in the same final absolute pose $P_t$ (textured with the query image). (b) If the results do not coincide, the pose $P_t$ with most effective inliers $I_{eff}$ (red matches) is preferred to other poses (blue matches).

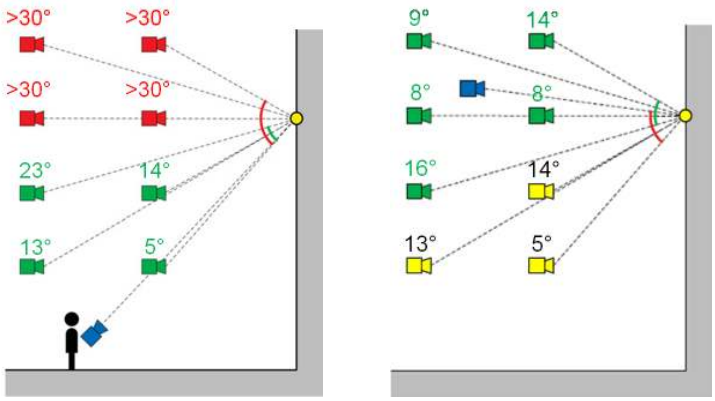
### 4.2.3 Updating Geometric Priors with In-flight Information

The previously presented approach has a major drawback, which is the lack of good features whenever the angle to the closest view used for modeling the scene is large. As images are typically taken at ground level, this especially applies when the MAV flies in altitudes higher than 5 m, but may also occur in many other scenarios.

**Incremental Feature Update.** In [222] we have proposed to tackle this issue by adding in–flight information to the model, which we call incremental feature update. The visual localization of every single frame results in a set of inlier features which each correspond to a 3D point $X$. We evaluate the new extraction angle

$$\delta_{update}(X,i) = \delta_{min}(X,i) - \arccos\left(\frac{\boldsymbol{XL}}{\|XL\|} \cdot \frac{\boldsymbol{XV_i}}{\|XV_i\|}\right) \tag{4.5}$$

between the localized camera $L$ and the virtual camera $V_i$. Respective inlier features in virtual camera $i$ are updated, if $\delta_{update}(X,i) > th_{update}$, i.e., if the new extraction angle is at least $th_{update}$ smaller than the previous one. Note that just the descriptor has to be exchanged; 3D points and 2D feature positions in the virtual camera remain constant. This update step is very fast and does not reduce the localization rate of 4 fps, as we exploit a precomputed list of relevant virtual cameras for every 3D point. By computing the extraction angle for every virtual view individually, we ensure that the average extraction angle is optimized. Figure 4.13(b) visualizes the update of cameras which initially had a high extraction angle $\delta_{min}(X,i)$. Virtual cameras in high altitudes are updated, while the features for lower altitudes remain the same.




(a) (b)

Figure 4.13: The concept of incremental feature updates. (a) Depending on the acquisition strategy, some virtual views (red cameras) might have an extraction angle larger than 30° compared to the model view (blue camera) which contains the real visual information. (b) The problem can be circumvented by incrementally updating virtual cameras with in–flight information (blue camera). The extraction angle for green cameras is reduced and features are updated; the extraction angle for yellow cameras would be increased so the features stay the same.

Incremental feature updates are generally applied in an *online* fashion during the flight of the MAV. This allows a large boost in performance, but early frames do not profit from new information in later stages of the flight. Therefore, we run an additional *batch* update with several iterations (until convergence) after the flight. Even a single additionally registered camera can provide the necessary descriptors for later iterations. During the batch update process we also update the inverted file tables which are used in vocabulary tree ranking.

**Validation using Additional Sensors.** A well-known problem in online learning and tracking is drift due to wrong feature updates, called the *template update problem* [138]. If wrong localizations are taken to be correct, the model is disturbed and the localization drifts towards the erroneous features. Incremental feature updates as presented in the previous section can also be affected by these issues, even though robust methods are applied already. To overcome this limitation, we propose to validate visual localization by using the additional global sensors available to an MAV.

We employ the magnetic field sensors and compass of the IMU to validate the estimated attitude, and the GPS readings and pressure sensor to evaluate the estimated position. Thanks to the alignment of our visual landmark introduced in Section 4.1, the comparison in a global coordinate system is straight–forward.

For representing the attitudinal error, we choose a measure based on a quaternion representation. A quaternion is defined as $Q = \begin{bmatrix} \boldsymbol{\rho}^T & q_w \end{bmatrix}^T$, with $\boldsymbol{\rho} = \boldsymbol{a} \sin\left(\frac{\theta}{2}\right)$ and $q_w = \cos\left(\frac{\theta}{2}\right)$, where $\boldsymbol{a}$ corresponds to the axis and $\theta$ to the angle of rotation. Thus, the attitude error between the camera attitude quaternion $Q_{cam}$ and the IMU attitude quaternion $Q_{magnetic}$ is calculated as

$$e_{attitude} = 2 \arccos\left( \, ||Q_{cam}^{-1} Q_{magnetic}||_w \, \right), \tag{4.6}$$

where $|| \cdot ||_w$ is an operation that first normalizes the quaternion resulting from the quaternion multiplication and then extracts $q_w$. The position error is given as

$$e_{position} = \| \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} - \begin{pmatrix} x_{latitude} \\ y_{longitude} \\ z_{pressure} \end{pmatrix} \|, \tag{4.7}$$

with $x_{latitude}$ and $y_{longitude}$ as GPS position converted to a local ECEF coordinate system, and $z_{pressure}$ as pressure height over ground. These error measures are required to be smaller than a threshold $th_{attitude}$ and $th_{position}$, respectively. Shadowing effects, weather changes, and magnetic fields may disturb these sensors as well, so the validation should serve as a sanity check rather than a narrow rejection criterion.

The combination of information about the rough orientation and position of the MAV in a world coordinate system also allows us to switch seamlessly between GPS and visual localization, for instance when navigating between several visual landmarks. Moreover, it enhances scalability to a world environment as the feasible set of virtual cameras discussed before is further restricted.

### 4.2.4 Summary

We have presented an algorithm for monocular visual localization of MAVs using prior knowledge about the scene. Our work is based on the concept of virtual views in 3D space, which allows to partition the search space for speeding up localization in large environments, but also supports the integration of in–flight information to improve the initial scene representation. Localization results generated by our algorithm are in a world coordinate system, which allows the MAV to switch seamlessly between GPS and visual localization.

## 4.3 Distributed Online Localization and Mapping

We have presented a novel approach for localization in known, point-based models, which is neither prone to drift nor bias as it is not based on feature tracking but matches directly against the 3D point cloud. While our scalable and flexible representation with virtual views allows a localization speed of up to 4 fps, this is not enough for visual navigation

of micro aerial vehicles. Moreover, the required hardware components to reach such a framerate, including a GPU, are typically not available on mobile devices.

In this section we show that we can apply visual localization and mapping even on devices with very low computational capabilities by distributing the computational load between the mobile device and a server. Image acquisition and tracking is performed directly on a mobile device with restricted computational abilities, and only selected keyframes are sent to a powerful server for sparse and dense mapping. Keyframes can be directly localized using prior reconstructions if available, thus geo-referencing them on-the-fly, or can be mapped as in standard visual SLAM approaches if no prior is available for a certain location. The resulting triangulated feature points are returned to the mobile device and can be directly used for tracking again.

In addition, we present the first system that is capable of generating live dense volumetric reconstructions based on monocular input provided by a remote platform. The user can additionally monitor the tracking quality using a live preview on a tablet computer and interact with the system on a touch–screen (see Figure 4.14). These dense maps could be used for various applications such as collision avoidance or reconstruction quality visualization.

(a) (b)

Figure 4.14: Our system is able to reconstruct a scene on-the-fly using a micro aerial vehicle. (a) Our distributed localization and mapping system allows to track the position of the MAV at full framerate while it is airborne, in completely unknown environments, or by exploiting geometric priors based on our virtual view localization framework. (b) A preview of the distributed SLAM tracking performance (left) and a dense reconstruction (right) are streamed to a tablet for visualization.

### 4.3.1 Related Work

A system capable of live, dense reconstruction of an environment, running on a remote mobile platform, requires three major components: Online localization and mapping for tracking the pose of the camera, a dense reconstruction approach to estimate the geometry of the scene, and a communication framework for distributing the workload to the most suitable entities in the system. The following paragraphs give an overview of selected work from these very active research areas.

Mobile visual localization and mapping is mainly used in augmented reality and robotics, where the computational resources are often much lower than in a desktop setting. We have already presented related work in augmented reality [115] and robotics [2, 21, 218], but all of these SLAM approaches are prone to drift and bias. Furthermore, these approaches do not allow to send high–quality imagery to the ground for real-time visualization or dense reconstruction because this would require more than the remaining processing power. In contrast, Reitinger et al. [167] presented an interactive 3D reconstruction system which uses a server–based reconstruction pipeline for urban modeling. A human scout selects suitable views and transmits the image and GPS positioning data to the server, which returns a dense point cloud after some seconds. Lee et al. [130] showed a similar concept on a mobile phone, featuring server–based pose estimation and Shape–from–Silhouette reconstruction. In comparison, our system performs pose estimation directly on the device but outsources sparse and dense reconstruction to the server. This allows on the one hand to process considerably more input images and to continuously obtain pose estimates, on the other hand the data transfer is restricted to high–quality keyframes.

Typical SLAM approaches are often not suitable for providing input to dense reconstruction, as they maintain a map of sparse feature points and do not store full frame information. Newcombe and Davison [151] presented a method which is able to generate a rough mesh based on sparse visual SLAM features. The mesh is then successively refined by depth information obtained using variational optical flow [228] between tracked frames, which leads to physically meaningful dense reconstructions. However, the topology of possible reconstructions is limited by the initial mesh and does not allow modeling of concave scenes containing holes or extrusions. Stühmer et al. [197] presented a similar real-time system which robustly generates accurate depthmaps based on variational optical flow, but their work lacks depthmap fusion and is thus restricted to 2.5D geometries. More recently, Graber et al. [70] presented a system based on variational depthmap fusion [234, 235] which is able to overcome the aforementioned limitations and works with a set of keyframes instead of all tracked frames. They propagate visibility information in a volumetric representation and extract the surface as zero level-set of a global convex energy, which leads to smooth reconstructions with arbitrary topology. All currently available live dense reconstruction systems based on passive visual sensors exploit general-purpose graphics processing units (GPGPUs) to achieve real-time performance and are

therefore not directly applicable in a mobile system.

A computationally viable alternative for real-time dense 3D reconstruction is the KinectFusion approach [152]. Unluckily, the structured light pattern projected by Kinect is only suitable for indoor usage. In general, all currently available RGB-D sensors suffer from very limited range in outdoor environments.

### 4.3.2 Overview of the Distributed SLAM System

We present a distributed system that is capable of real-time localization and mapping based on monocular input from a mobile platform. Matching our goal of visual navigation we choose a micro aerial vehicle with an on–board computer for image acquisition; however, our approach is as well perfectly suited for current tablet computers. The mobile device is used to capture live video and to track the pose of the camera in the environment. Having both capturing and tracking in one place allows to make use of the full frame rate delivered by the camera, and provides localization without delay and the need for persistent connectivity. This is for instance important when using a visual servoing approach in robotics [2, 224] (Section 5.2).

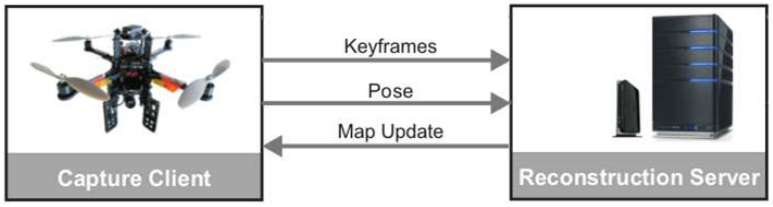



Figure 4.15: Overview of the distributed SLAM system. The tracker runs on the mobile device and computes a pose for every single frame. Keyframes are selected heuristically based on position and attitude changes, and only a sparse set of frames is thus transmitted to the server. After sparse mapping, the server only needs to send the 3D map information to the tracker; the extracted visual features are already there. This allows fast and efficient map updates despite the distributed operation.

We further employ a server equipped with high–performance hardware, including a state–of–the–art graphics processing unit (GPU), for sparse and dense mapping. The work of Klein and Murray [113] has shown that mapping can be done using keyframes only, which is the principle we rely on in this work. Keyframes are selected by the mobile device based on the coverage of the current map and transmitted in full quality to the server. Given the additional pose estimate by the tracker, they are directly integrated into the map and bundle adjustment is performed. Within seconds, the updated map is sent back to the mobile device and tracking continues on the updated map.

We want to stress that our system as described above and depicted in Figure 4.15 is very flexible in terms of hardware and algorithms. The coupling between individual

parts is very loose and based on the robotic operating system (ROS)[1] and a wireless link. Our work as described in the following paragraphs is based on the work of Klein and Murray [113, 114] with some major changes relating to the distributed operation. However, all parts can easily be adjusted to reflect the application at hand or the future developments in tracking and mapping.

### 4.3.3 Tracking on the Micro Aerial Vehicle

The tracking part of our system runs on the mobile device and has two important tasks: It has to deliver pose estimates for every input frame and it has to select keyframes based on the the scene coverage of the map. Processing the data directly avoids tracking on the ground, thus pose updates are not delayed and the low transmission bandwidth required enables operation over Wifi and 3G.

For every acquired frame, our tracking system extracts FAST-10 [173] features on a four-level image pyramid. We follow a two–stage tracking procedure which first searches for the 50 largest features on coarse levels of the image pyramid based on the prior pose estimate, updates the pose accordingly, and then performs accurate tracking by searching for 1000 features in finer levels. Due to computational constraints, we use very simple $8\times8$-pixel patch descriptors and match them to mapped descriptors at FAST corner location within a circular search region. This is especially useful when updating the map, because features do not need to be stored separately but can be indexed directly by the triplet $(id, s, \mathbf{x}_{2d})$, where $id$ denotes the unique keyframe index, $s$ denotes the pyramid scale, and $\mathbf{x}_{2d}$ the patch position in the image. Undistortion is only applied on a per–feature basis to reduce computational effort. A detailed description of the tracking and pose estimation process can be found in the original PTAM work [113].

Keyframe selection on the mobile device is important to avoid streaming too much data to the ground. To start the tracking process, an initial map consisting of two keyframes is necessary. PTAM requires the operator to select these two frames manually, which has several issues: It requires streaming the data to the ground, it requires some experience in moving the camera, and it results in maps with arbitrary scale and origin. In contrast, we employ a single artificial marker with known dimensions to initialize the map directly on the mobile device. We use ARToolkitPlus [215] to obtain an accurate, metrical localization relative to the marker. Once the marker is visible in the image, candidates for the first keyframe are stored and SURF features [12] are extracted. Candidates for the second keyframe are stored when a baseline of more than $b_{init}$ has been robustly measured (i.e. over 10 frames). Once more than $f_{init}$ SURF features can be successfully matched between any pair of candidates fulfilling the required baseline, the two keyframes are transmitted to the server-based mapper. Given the two poses obtained by ARToolkitPlus, the essential matrix is estimated. The initial map is triangulated and refined through bundle adjustment. The resulting map with metrical scaling and a defined origin in the center of the

[1] http://www.ros.org

marker is finally returned to the tracker. While marker-less initialization works as well, the proposed process is more comfortable and typically takes less than 5 seconds.

A proper initialization to metrical scale also aids further keyframe selection for expanding the map. We transmit a keyframe to the server if the distance $d$ to the nearest keyframe in the map exceeds a minimum baseline $b_{map}$, or if $d > 0.25b_{map}$ and the rotation angle $\alpha$ relative to the nearest keyframe exceeds a minimum angle $\alpha_{map}$. Additionally, keyframes are only sent if the tracking quality is good and if a certain time $t_{map}$ has passed since the last keyframe has been added. In comparison to PTAM, $\alpha_{map}$ ensures that keyframes are also added when the mobile device rotates, and $t_{map}$ compensates for the latency between requesting a map update and actually receiving it, while at the same time ensuring that the tracker does not freeze while waiting for a map update.

Every time the tracker is lost or the received map contains considerable changes, re-localization is necessary. We employ a fuzzy image re-localization approach [114] supported by GPS and compass data (if available) for outdoor usage. We store the initial GPS/compass pose of the mobile device and insert all further keyframes relative to that. The set of blurry, scale-reduced keyframes is ranked based on the distance to the current GPS/compass pose and then matched as in the original approach. As a result, our approach can re-localize in difficult outdoor scenes with repetitive features.

### 4.3.4 Sparse Mapping on the Server

The mapping part of our system runs on the server and is responsible for providing a sparse three-dimensional structure which can be used for pose estimation by the tracker. The mapper gets keyframes with an estimated pose and a unique identifier $id$ in irregular intervals from the remote tracker. These images are transmitted uncompressed and in full–resolution; an important detail that allows the mapper to detect the same features as the tracker. New map points are added by triangulation with neighboring views and refined by local bundle adjustment as described in [113]. Global bundle adjustment is applied whenever the mapper is idle, but interrupted if a new keyframe should be added.

Once local bundle adjustment has converged, the refined map is ready to be sent back to the tracker. All keyframes available to the mapper are already stored on the tracker, so it is sufficient to transmit the vector $(id, s, \mathbf{x}_{2d}, \mathbf{x}_{3d})$ for every new point, and for every point that was moved by bundle adjustment by

$$\|\mathbf{x}_{2d} - \mathbf{x}_{2d,old}\|_2 > \epsilon_{2d} \text{ or } \|\mathbf{x}_{3d} - \mathbf{x}_{3d,old}\|_2 > \epsilon_{3d}. \tag{4.8}$$

A typical map update can therefore be performed very quickly and does not require high bandwidth. If large maps need to be refined by bundle adjustment, we propose to iteratively update the map starting with points in the vicinity of the current pose and proceeding with points further away. When the map update is successful, the tracker automatically resets and continues to track using the new map.

### 4.3.5 Extension to Dense Mapping

Next to fast sparse mapping which is crucial for visual navigation, we run a second thread on the server for dense reconstruction. This extends our framework as depicted in Figure 4.16. Starting at the minimum of three keyframes, we estimate a depth map for every new keyframe and its set of neighboring views using multi–view plane sweep [28]. The depth maps are then refined with variational denoising and integrated into a volumetric representation using variational depth map fusion [234, 235]. This has already been described in Section 3.3. The resulting smooth three-dimensional surface is rendered on the GPU and transmitted to the user if new data is added to the volume or if the user triggers viewpoint changes.


Figure 4.16: Overview of the distributed live dense reconstruction system, which exploits the individual capabilities and requirements of the system entities. The tracker is again run on the quad–rotor, sparse and dense mapping on the server, and visualization on the tablet.

Finally, the live tracking preview and the dense reconstruction are visualized on the mobile device; for aerial acquisition, this is a tablet computer on the ground. The user can on the one hand control image acquisition and on the other hand rotate and zoom the live dense reconstruction.

Dense maps created in real-time could be used for various applications. Our localization and mapping framework based on virtual views as shown in previous sections is perfectly suitable for integrating the additional data as prior knowledge for future flights. Also, dense data is important for collision avoidance, which will be discussed in the next chapter. Finally, our system is perfectly suited for monitoring the reconstruction quality during acquisition; the lack of such a technique has so far prohibited a wide-spread use of 3D reconstruction on mobile devices.

### 4.3.6 Summary

We have shown how existing state-of-the-art approaches to online localization and mapping can be adapted to work in a distributed environment with modern technologies such as micro aerial vehicles for image acquisition and tablet computers for visualization. Our work focuses on exploiting the individual capabilities of the devices, which leads to an interactive system that is able to tolerate mistakes of the individual parts. Apart from

providing a low-level layer for our localization pipeline, we are also able to generate dense volumetric reconstructions in real-time and thus extend our maps of the environment [223].

## 4.4 Online Localization using Line-based Models

Localization in 3D point clouds is an important cue for visual navigation, but sometimes it is necessary to localize relative to a single, given object. While many objects could be simply represented as 3D point cloud as well, objects with little texture or even wiry geometry such as power pylons need special treatment.

As discussed in Section 2.2 three different groups of algorithms can be distinguished by means of their approach to determine the pose relative to a known object. Nevertheless, only correspondence-based approaches are suitable for objects without associated texture. Initial experiments have shown that it is difficult to estimate a pose relative to complex structures from single images. Given a sequence of images, the application of a visual SLAM method such as Parallel Tracking and Mapping (PTAM) [113] again seems to be most reasonable. To cope with the drawbacks of visual SLAM, we extend this keypoint-based approach to the use of model-based information for tracking, and additionally for refining the mapping process.

### 4.4.1 Related Work

Bleser et al. [20] proposed a robust real-time camera pose estimation approach for partially known scenes. The connection between the real and the virtual world is made by one known object, given as a CAD-model. However, the model is only used during initialization which is semi-automatic: The model is projected to the image using a predefined pose, which the user then has to register manually to the image by moving the camera. When the projected model is close enough to register the model lines to the image gradient, initialization completes automatically. After initialization, point-based tracking is applied to estimate the pose using warped patches around extracted feature locations.

Others use model-based pose estimation [112, 168] together with gyroscopic measurements to be able to cope with fast motions and the resulting motion blur. While [112] is used in augmented reality (AR) for head-mounted displays, [168] is used for outdoor AR and extends the former work of Klein by the usage of a texture-based model and online edge extraction. Both approaches use the edge-based tracking algorithm by Drummond and Cipolla [46] to improve over purely point-based tracking.

The work of Kyrki et al. [123] is most closely related to our work, as they investigated the tracking of rigid objects using the integration of model-based and model-free cues in an iterated Extended Kalman Filter (EKF). The model-based cue uses a wireframe-edge model of the object, whereas the model-free cue uses automatically generated surface texture features. Thus, the approach relies on both edge and texture information. The model-based tracking is similar to [46, 47], who estimate the normal flow for points along

edges. The normal flow is then projected in the direction of the contour normal so that the error can be minimized using robust *M-Estimators* [94]. For the model-free cue, Harris corners are extracted and tracked for visible parts of the object by minimizing the Sum of Squared Differences (SSD) in RGB values. The algorithm is able to deal not only with polyhedral objects, but also with spherical, cylindrical and conical objects. However, in contrast to our work this approach only makes use of texture information available on the object, and hence is not capable of dealing with non-solid, wiry objects.

We therefore focus on a tracking approach which exploits discriminative features in the background of the object. Thus we can track complex structures and determine a pose when parts or even the entire object is occluded. Additionally, a line-based model of the object is used for metrical initialization as well as in a refinement step, where the pose estimate delivered by keypoint-based tracking is enhanced. This helps to improve accuracy especially when being close to the object. Furthermore, when keypoint-based tracking fails due to a lack of distinctive features, purely model-based tracking takes over. This allows to avoid tracking failures until revisiting explored areas, where we seamlessly switch back to keypoint-based tracking. The previously presented virtual view localization can additionally be used to retrieve global keyframe poses in a geo-registered point cloud.

### 4.4.2 Line-based Tracking Algorithm

We extend the visual SLAM framework of Klein et al. [113] by several model-based components: First, we use a line-based model of the object to generate a metrically correct map during initialization. Hence, a metrically correct pose is determined in every frame and can be fused with other systems such as the GPS. During initialization some user interaction is required, however, we thus avoid object-dependent pose ambiguities, i.e. multiple poses for which the object appears the same, right from the beginning.

After having initialized the keypoint-based tracking, the object is in fact only tracked by its surroundings. This has on the one hand the advantage that the pose can be estimated over multiple scales, and that the pose estimation is robust to partial occlusions of the object. Additionally, pose estimation is even possible when the object completely vanishes, which is not possible using pose estimation approaches based solely on the object. On the other hand, given a good prior pose estimate from keypoint-based tracking, we can use the knowledge about the object's geometry to refine the pose, which is our second extension. Keypoint-based pose estimation works well when enough discriminative world information is detected in the current image.

Finally, when thinking for instance of inspection tasks, closeup views of the object are sometimes necessary. When being very close to an object, keypoint-based tracking often fails; however, the model is very prominent in such cases. As a third extension, we therefore impose a purely model-based tracking component. Model-based tracking tries to close the gap until revisiting already explored areas, where we can seamlessly switch back to keypoint-based tracking. An overview of the entire system is given in Figure 4.17.

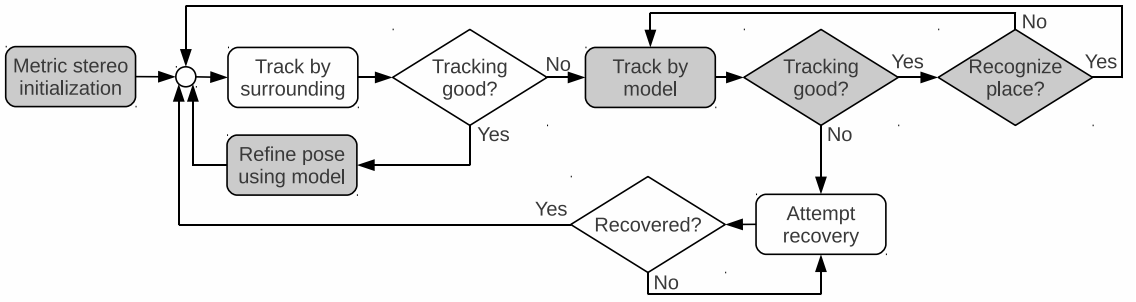



Figure 4.17: Model-based tracking system overview. After metric initialization using the model the object is tracked by its surroundings and refined using model edges. When this approach fails, a model-based tracking takes over. It seamlessly switches back to background-based tracking when visiting already explored areas. The gray blocks depict the extensions compared to *PTAM* [113].

**Metric Initialization.** Detailed information about a single object in the scene is given in form of a wireframe model or line-based reconstruction. This information is used to initialize the visual SLAM algorithm metrically. To determine the first camera pose $\mathbf{C}_1$, the user has to choose $n$ correspondences between 3D-model points and 2D-image points. Then, a Perspective $n$ Point problem (PnP) is solved using Zhang's method [239]. While the user translates the camera sideways, salient image points are tracked. From these correspondences we estimate a homography between $\mathbf{C}_1$ and $\mathbf{C}_2$ which is further decomposed into a rotation $\mathbf{R}$ and a relative translation vector $\hat{\mathbf{t}}$ using [56]. Finally, the user has to choose another previously selected model point. Now that two image locations of one and the same point, namely $\mathbf{x}_{\mathbf{C}_1}$ and $\mathbf{x}_{\mathbf{C}_2}$, and the relative camera translation $\hat{\mathbf{t}}$ and rotation $\mathbf{R}$ between the two views $\mathbf{C}_1$ and $\hat{\mathbf{C}}_2$ are known, the corresponding 3D-model point $\hat{\mathbf{X}}_{\mathbf{C}_1}$ can be triangulated in the camera coordinate frame of camera $\mathbf{C}_1$. After the 3D-model point $\mathbf{X}$ has been transformed to the same coordinate frame, the disparity in length between the true $\mathbf{X}_{\mathbf{C}_1}$ and the reconstructed 3D-model point $\hat{\mathbf{X}}_{\mathbf{C}_1}$ can be calculated. The scale factor

$$s = \frac{||\mathbf{X}_{\mathbf{C}_1}||}{||\hat{\mathbf{X}}_{\mathbf{C}_1}||} \tag{4.9}$$

can be derived using similar triangles, where $||\ \cdot\ ||$ depicts the length of the vector. The factor is then used to scale the baseline to its correct metric length, yielding the true metric translation vector $\mathbf{t} = s\hat{\mathbf{t}}$. This procedure is depicted in Figure 4.18.

As the true baseline is finally available, a metrically correct map is generated using triangulation of pairwise corresponding image features.

**Model-Based Refinement and Tracking.** We distinguish between model-based refinement which is applied when keypoint-based tracking is good, and purely model-based

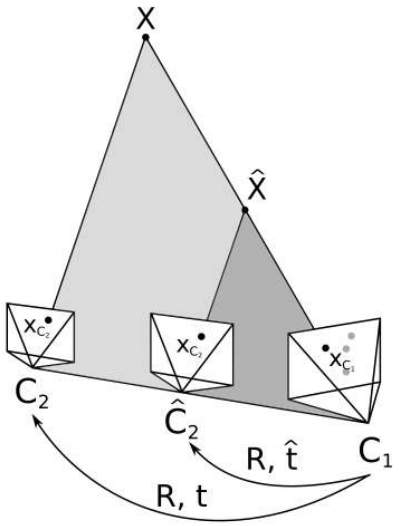



Figure 4.18: Initialization of model-based tracking. $\mathbf{C}_1$ is determined by solving a *PnP*-problem. $\mathbf{R}$ and the relative translation $\hat{\mathbf{t}}$ are extracted from an estimated homography, while the correct location of $\mathbf{C}_2$ is determined by rescaling the baseline. The scaling factor is determined from the disparity in length between a reconstructed $\hat{\mathbf{X}}$ and the true model-point $\mathbf{X}$.

tracking in all other cases. Both methods work the same, but during model-based tracking the prior pose estimate is not delivered by keypoint-based tracking but results from the refined pose of the last frame. However, in both cases the same steps are necessary.

First, all model points are projected to the image and a bounding box is calculated. A refinement is only triggered if the ratio of the areas

$$\frac{A_{\text{ bounding box}}}{A_{\text{ image}}} > t \tag{4.10}$$

exceeds a certain threshold $t$. Otherwise, the object is either not present in the image anyways, or so small that calculating a refinement based on the object's model would not improve the camera's pose notably. As the refinement is based on gradient information, the current image is convolved with the derivative of a Gaussian kernel $G_x(x, y, \sigma_c)$ in x-direction and $G_y(x, y, \sigma_c)$ in y-direction, giving the image derivatives $I_x$ and $I_y$, respectively. The previously calculated bounding box is increased by a certain percentage, and the gradient magnitude $|\nabla I(x, y)|$ as well as the gradient direction $\psi(x, y)$ are extracted for this region of interest only.

Then, additional model points need to be interpolated on the edges connecting a pair of model points and their visibility is determined by the use of a *rendering component*. Therefore, the CAD-model is rendered under its currently estimated pose $\mathbf{E}^*_{\mathcal{CW}}$ and the z-buffer is retrieved. Then the 3D-world points $\mathbf{X}_{\mathcal{W}}$ are transformed to the camera coor-

dinate frame via

$$\mathbf{X}_{\mathcal{C}} = \mathbf{E}^{*}_{\mathcal{C}\mathcal{W}}\mathbf{X}_{\mathcal{W}} \ . \tag{4.11}$$

Then, the 3D-point $\mathbf{X}_{\mathcal{C}}$ is projected from the camera coordinate frame to normalized clipping coordinates $(x_{ncc}, y_{ncc}, z_{ncc})^T$ using the *OpenGL* camera projection matrix which has been applied during rendering, and further linearly transformed to image coordinates $(u_{ncc}, v_{ncc})^T$. The z-buffer is finally evaluated at these coordinates to retrieve the correct depth value for the 3D-point. If the difference

$$|z_{ncc} - z_{buffer}\,(u_{ncc}, v_{ncc})\,| < \tau, \tag{4.12}$$

where $z_{buffer}\,(u_{ncc}, v_{ncc})$ is the corresponding z-buffer value and $\tau$ a scalar threshold, the 3D-point $\mathbf{X}_{\mathcal{W}}$ is considered visible, otherwise invisible.

The technique we use to refine our pose is based on an edge-based tracking algorithm proposed by Drummond and Cipolla [46], which has already been successfully applied to several augmented reality applications [112, 168]. For all model points including the interpolated ones, a linear search for the strongest gradient magnitude is performed as depicted in Figure 4.19. The search line $\eta$ is arranged perpendicular to the model's edge. The projected model point is now shifted along the perpendicular search line to the image point $(x, y)$ which maximizes

$$\max_{(x,y)} \left( |\nabla I(x,y)| e^{-\frac{v^2-\mu^2}{2\sigma^2}} - \lambda \sphericalangle (\boldsymbol{\psi}(x,y), \boldsymbol{\eta}) \right), \tag{4.13}$$

where the image gradient magnitude $|\nabla I(x, y)|$ is weighted with a Gaussian windowing function centered at the projected model point's position $\mu$, the scalar $v$ denotes the distance along the search line, and $\lambda \sphericalangle(\mathbf{n}_1, \mathbf{n_2})$ is a term that penalizes orientation deviations between the perpendicular search direction $\boldsymbol{\eta}$ and the image gradient direction $\boldsymbol{\psi}(x, y)$. In short, what we are looking for is the nearest image location of the projected model point with the strongest gradient response and approximately the same direction.

The image distances between old and new positions are assigned to an error vector $\mathbf{e}$. We then calculate a motion vector $\boldsymbol{\mu}$ which minimizes these image errors by solving the equation system

$$\mathbf{J}\boldsymbol{\mu} = \mathbf{e} \ , \tag{4.14}$$

where $\mathbf{J}$ is a Jacobian matrix describing the effect of each element of $\boldsymbol{\mu}$ onto each element of $\mathbf{e}$ in direction perpendicular to the edge. The elements of $\mathbf{J}$ are therefore calculated by taking the partial derivatives at the current pose along the edge normal as

$$J_{ij} = \frac{\partial e_i}{\partial \mu_j} = \langle \hat{\mathbf{n}}_i \ , \begin{pmatrix} \frac{\partial u}{\partial \mu_j} \\ \frac{\partial v}{\partial \mu_j} \end{pmatrix} \rangle \ , \tag{4.15}$$

where $(u, v)^T$ are the image coordinates of the $i^{th}$ projected model point, $\hat{\mathbf{n}}_i$ is the cor-

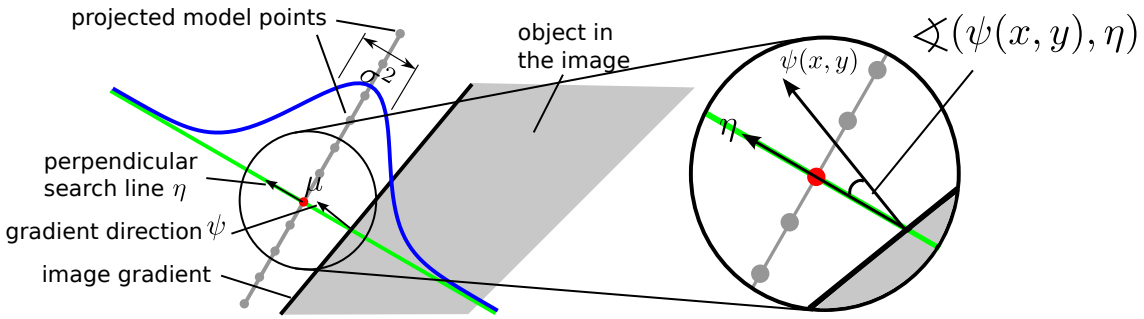



Figure 4.19: Perpendicular search strategy. For each projected model point, a linear search perpendicular to the model's edge is performed. The image point is thereby shifted along the search line $\boldsymbol{\eta}$, while looking for a strong image gradient with approximately the same direction. The gradient magnitude is weighted with a Gaussian windowing function centered at $\mu$, where $\sigma^2$ controls the expansion of the search region.

responding unit length edge normal, and $\langle x, y \rangle$ denotes the tensor dot product. A world point $\mathbf{p}_{i\mathcal{W}}$ is projected to the image coordinates $(u, v)^T$ by first transforming it to the camera coordinate frame using the actual pose estimate $\mathbf{E}^*_{\mathcal{CW}}$ and then performing the camera projection as

$$\begin{aligned} \begin{pmatrix} u_i \\ v_i \end{pmatrix} &= \text{CamProj}\left(\mathbf{E}^*_{\mathcal{CW}} \mathbf{p}_{i\mathcal{W}}\right) \\ &= \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \end{bmatrix} \begin{pmatrix} \frac{X_{i\mathcal{C}}}{Z_{i\mathcal{C}}} \\ \frac{Y_{i\mathcal{C}}}{Z_{i\mathcal{C}}} \\ 1 \end{pmatrix} . \end{aligned}$$

The radial lens distortion can be ignored here, as the images are undistorted prior to applying our algorithm. Thus, Equation 4.16 has to be differentiated with respect to the motion parameter vector $\boldsymbol{\mu}$ by applying the chain rule. The camera's pose $\mathbf{E}^*_{\mathcal{CW}}$ is represented by a $4 \times 4$ matrix of the Special Euclidean Lie Group *SE(3)* [211] which may be parameterized by a six-dimensional motion vector $\boldsymbol{\mu}$ via the exponential map, where $\mu_1$, $\mu_2$ and $\mu_3$ represent translation along $x$, $y$ and $z$ axes, and $\mu_4$, $\mu_5$ and $\mu_6$ describe the rotation around these axes, respectively. An *SE(3)* matrix $\mathbf{E}$ is calculated from a six-dimensional motion vector $\boldsymbol{\mu}$ as

$$\mathbf{E} = \exp\left(\sum_{i=1}^{6} \mu_i \mathbf{G}_i\right), \tag{4.16}$$

where $\mathbf{G}_i$ are called generator matrices. All operations on the *SE(3)* group are continuous and smooth, and hence differentiable in closed form. Differentiating a motion matrix $\mathbf{E}$

at the origin $\boldsymbol{\mu} = \mathbf{0}$, the partial derivatives

$$\frac{\partial \mathbf{E}}{\partial \boldsymbol{\mu}_i} = \mathbf{G}_i \tag{4.17}$$

are simply the generator matrices. Given all necessary mathematical expressions, Equation 4.14 may be solved for the motion-update vector $\boldsymbol{\mu}$ using a simple least-squares solution

$$\boldsymbol{\mu} = \mathbf{J}^{\dagger}\mathbf{e} \ , \tag{4.18}$$

with $\mathbf{J}^{\dagger} = \left[\mathbf{J}^T\mathbf{J}\right]^{-1}\mathbf{J}^T$ being the pseudo-inverse. However, as commonly known, least-squares solutions are not robust to outliers. As in this case gross outliers are very likely to occur due to noise, occlusions or falsely matched points during the perpendicular search, solving the above equation system in a least-squares manner is not appropriate. Therefore, a robust estimation for $\boldsymbol{\mu}$ is performed using Iteratively Reweighted Least-Squares (IRLS) by estimating

$$\boldsymbol{\mu}' = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_i w_{\mathrm{TUK}}\left(||\mathbf{e}_i||\right) ||\mathbf{e}_i||^2 \tag{4.19}$$

in each iteration. The weights $w$ are obtained using a Tukey *M-Estimator* [95]. Thus, the prior pose $\mathbf{E}^*_{\mathcal{CW}}$ is updated as

$$\mathbf{E}_{\mathcal{CW}} = \exp\left(\boldsymbol{\mu}'\right)\mathbf{E}^*_{\mathcal{CW}} \ . \tag{4.20}$$

An example for a refinement using ten iterations is given in Figure 4.20.

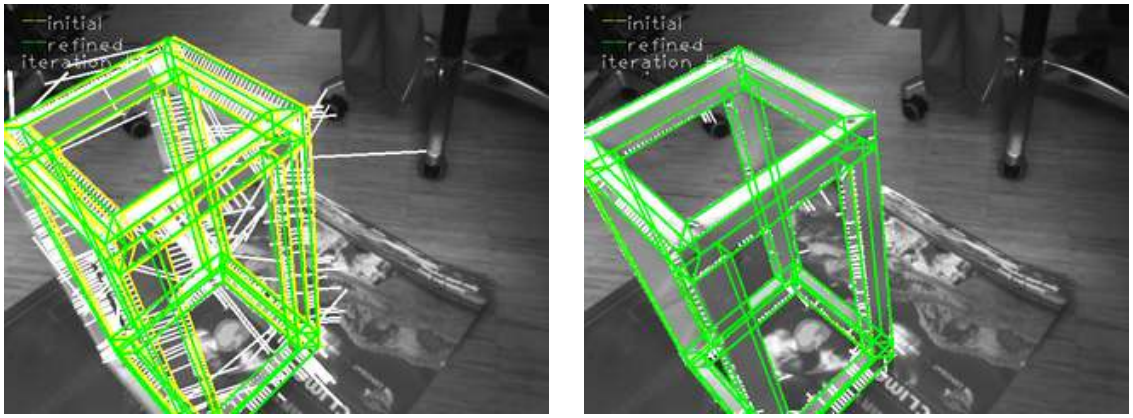



Figure 4.20: Iterative pose refinement result. The left image shows the result of the first iteration, while the right one depicts the last iteration and thus the final result. The pose of the prior iteration is shown in yellow, the refined pose in green, and the error vectors are depicted by white lines.

### 4.4.3 Summary

We have presented an online line-based pose estimation approach which builds upon a state-of-the-art visual SLAM implementation and integrates prior knowledge of the object of interest during the entire tracking procedure [109]. By exploiting discriminative features in the background we are able to estimate the pose of the camera relative to complex structures such as power pylons, and we are even able to determine a pose when the entire object is occluded. Beyond using the model for metric initialization we employ a refinement step which improves the accuracy of the pose estimate delivered by the SLAM system. Furthermore, when keypoint-based tracking fails as for closeup inspection, purely model-based tracking takes over and successfully reduces tracking failures.

# Chapter 5

# Multi-Scale Path Planning and Control

**Contents**



Path planning and control are important components of robot navigation. We have already presented common path planning methods and control strategies in Chapter 2, and it is evident that a variety of sensing and localization techniques could be employed as input providers. Path planning and control approaches typically require a dense map representing occupied and free space, high-framerate position or pose measurements, and a target position defined in a common coordinate system. For visual input, we have shown how to fulfill these requirements in the previous chapter. In this chapter, we focus on the incorporation of geometric priors and properties of visual input for path planning and control in three different layers.

**Global Layer.** When planning paths for aerial vehicles on a global scope, most approaches employ the global positioning system (GPS) for localization and assume an obstacle-free airspace, use radar technology [16], or rely on sense-and-avoid strategies [41]. However, when flying at lower altitudes in urban environments, many obstacles occur. Luckily, most can already be roughly inferred from geometric prior knowledge in form of digital surface models, as we will show in the next section.

**Local Trajectory Layer.** More locally, it is necessary to follow trajectories which are defined relative to a geo-referenced point cloud. Visual localization delivers very accurate pose estimates within a range of $\pm 10\,\mathrm{cm}$. However, these estimates might be affected

by non-Gaussian noise which poses a problem to many controllers. We present a visual servoing system for MAVs based on a fuzzy logic controller to overcome this issue.

**Reactive Layer.** Finally, not all environments are suitable for accurate 3D modeling. A reactive controller can overcome situations where an accurate pose estimate is not available, and it can handle visually difficult scenes such as forest environments. We demonstrate how a reactive control strategy can be learned purely based on visual input by imitating human experts.

## 5.1 Vision for High-Level Path Planning and Control

Large-scale, open-airspace path planning and control is traditionally based on the global positioning system (GPS) as primary navigation sensor. We investigate how prior knowledge in form of digital surface models can be used to for high-level path planning, as well as view planning for optimized image acquisition. Such an approach can for instance be applied to construction site monitoring, where temporally repeated flights in urban environments need to be performed. Large safety margins need to be incorporated to cope with the uncertainty of GPS-based control.

### 5.1.1 Related Work

Conway [30] presented one of the first examples for GPS-based autonomous flight. The system was able to deliver pose updates at 10 Hz, resulting in a position accuracy of a few meters. However, GPS-based systems tend to drift over time and the accuracy depends strongly on the number of available satellites. Current techniques fuse data from an IMU and GPS, called inertial/GPS approach, to fully stabilize and control MAVs [1, 233]. GPS can be replaced by much more accurate differential GPS (DGPS) receivers if high costs are acceptable, but this still does not circumvent shadowing effects caused by neighboring buildings in urban environments, and the general blindness of GPS regarding obstacles.

While GPS is still the most widely used sensor for outdoor localization, the trend goes towards visual collision avoidance. Zufferey et al. [243] combined GPS–based path following with vision–based collision avoidance. Their bio–inspired approach allowed them to fly close to the ground, approx. 9 m above the terrain, while effectively detecting and avoiding single trees during flight. Hrabar et al. [91, 92] used a similar approach based on optical flow and differential GPS to navigate an autonomous helicopter safely through a simulated urban canyon. We employ similar techniques but adapt the algorithms to plan on geometric prior knowledge.

### 5.1.2 Path Planning based on Digital Surface Models

Our approach to path planning in digital surface models consist of a combination of probabilistic and heuristic path planning methods. We first create a Probabilistic Roadmap

(PRM) [106] by sampling and connecting suitable states, then plan a path with D* Lite [119]. As the configuration space is sparsely sampled, the resulting path needs to be smoothed in the end. These steps are described in the following paragraphs.

**Constructing a Probabilistic Roadmap.** A probabilistic roadmap is a graph with randomly sampled but collision-free nodes and collision-free edges. Before sampling we add a safety margin of $d_{margin}$ meters to the height values of the digital surface model, and we apply a maximum filter with a spatial extent of

$$\sigma_{margin} = \frac{d_{margin}}{GSD_{DSM}} \text{ pixels} \tag{5.1}$$

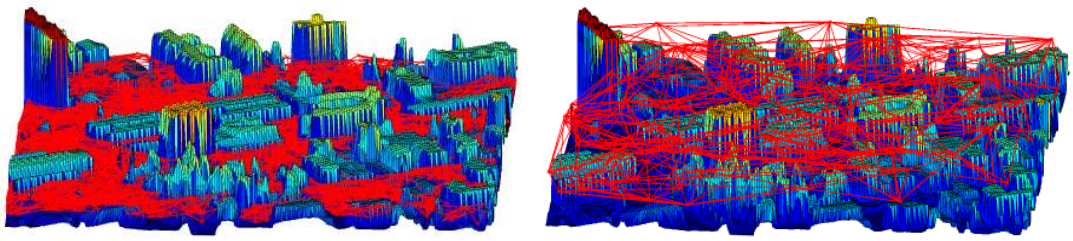
to the digital surface model, where $GSD_{DSM}$ is the ground sampling distance of the DSM. We do not model the yaw angle as part of the configuration space and thus sample uniformly from a three-dimensional $xyz$-waypoint space. While the limits in $x$ and $y$ are given by the DSM, valid altitudes $z$ are limited between the DSM height value $z_{DSM}(x, y)$ and a maximum flight altitude $z_{max}$. New nodes are sampled until a maximum of $N_{PRM}$ is reached.

Next, the nodes of the graph need to be connected to form the roadmap. While typically a $k$-nearest neighbors graph is used, we found that this connection strategy often leads to tightly connected clusters of points without a connection to neighboring clusters. We thus employ a 3D Delaunay triangulation [22] for connecting the nodes, which is not only computationally efficient but has proven to work well for PRMs [93]. Finally, all edges are assigned Euclidean distance costs. Edges causing collisions receive infinite costs and are thus virtually removed. This allows to update possible paths later when in-flight data is available. Note that there is no need to check for collisions surrounding the direct path as the DSM geometry has been dilated before.

A visualization of two probabilistic roadmaps based on DSM geometry is given in Figure 5.1. The graphs differ considerably when changing the maximum flight altitude from $z_{max} = 10\,\text{m}$ to $30\,\text{m}$; instead of planning only in between buildings, there are suddenly paths to fly over buildings available as well.

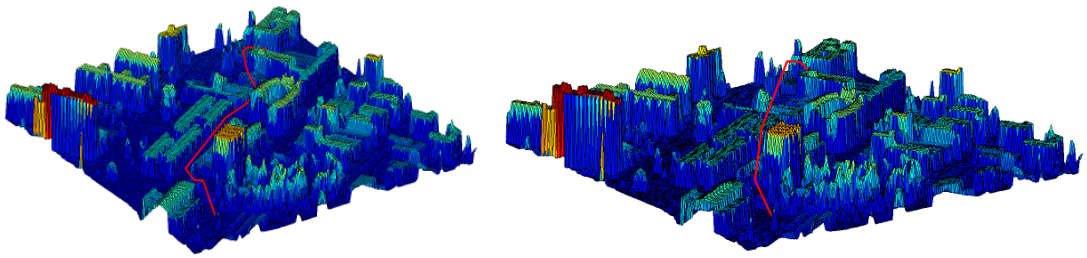
**Planning with D* Lite.** At query time, individual start and target nodes are added to the graph. We greedily check the nearest neighbor nodes using a kd-tree until a collision-free connection for both start and target node is found. This completes the roadmap map, and we employ the D* Lite algorithm to find the best path through the graph. Note that in the query phase no distinction between a 2D, 3D, or any other kind of world configuration is made; only the costs in the graph matter.

The resulting path consists of waypoints in a geo-referenced DSM, which can be easily converted back to GPS coordinates. A PID controller can then be used to follow these waypoints; most commercially available MAVs already include this feature. To prevent situations where the MAV is flying in a different direction than the one it is heading for,



(a) (b)

Figure 5.1: Probabilistic Roadmap based on DSM geometry. In (a), the maximum flight altitude is limited to $z_{max} = 10\,\text{m}$ above ground, whereas in (b) $z_{max} = 30\,\text{m}$ above ground. The difference in terms of available direct connections is clearly visible. For visualization of (b), only 10% of all nodes are depicted. $N_{PRM} = 2000$ in both cases.



(a) (b)

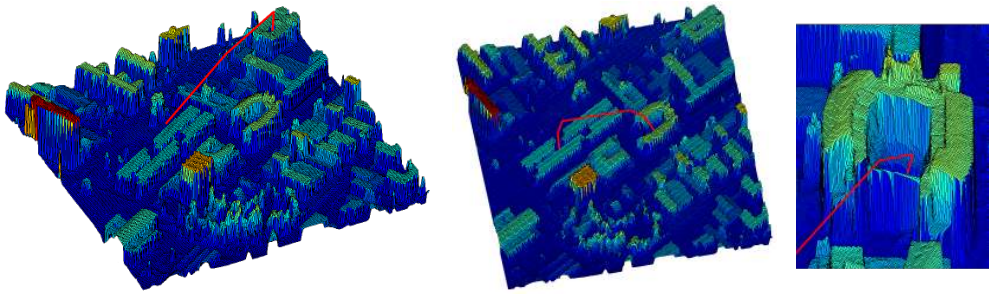
Figure 5.2: Planning results using the D* Lite algorithm. (a) Given a maximum flight altitude of $z_{max} = 10\,\text{m}$ above ground, the path has to be planned around and between buildings. (b) Switching to $z_{max} = 30\,\text{m}$ results in a flight plan on top of the buildings.

we adjust the yaw at every waypoint to look towards the next waypoint. As a result, we would like to minimize the number of waypoints in the path.

**Path Smoothing.** As the configuration space is sparsely sampled and thus does not necessarily prefer direct connections, the resulting path can be smoothed in most cases. We use the Douglas-Peucker algorithm [165] which iteratively reduces the number of nodes within a path using a split-and-merge strategy. Starting with a straight connection of start and target node, the path is iteratively split at the node with largest normal distance to the line in case of a collision, or kept if no collision occurs.

Figure 5.2 confirms what the construction of the PRM indicated; for a maximum flight altitude of $z_{max} = 10\,\text{m}$ the final path has to be planned around and in between buildings. In turn, this means that some points in the DSM cannot be reached because they are isolated from the graph. Figure 5.3 shows two examples where this is the case. Note that

planning on 2.5D DSM representations also has limits; the atrium in Figure 5.3(c) cannot be accessed although in reality there is just a bridge on top of the building separating it from the graph. Such scenarios motivate the need more detailed models with close-up views, which can be achieved by integrating terrestrial or oblique aerial views in the prior geometry.



(a) (b) (c)

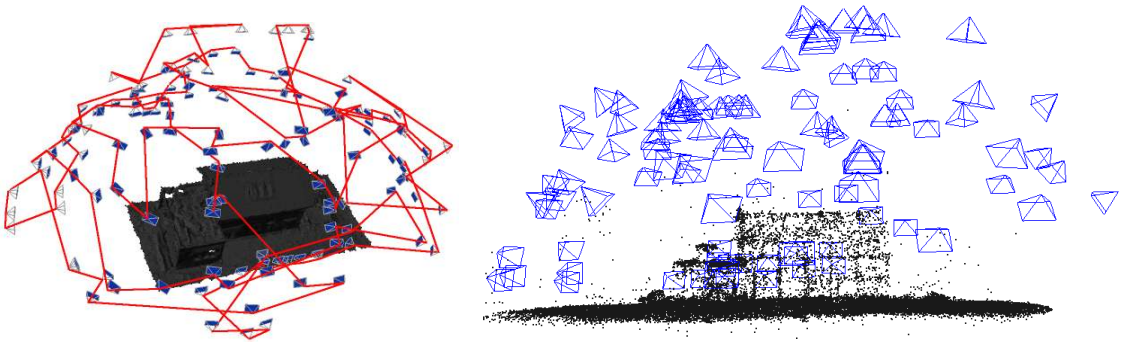
Figure 5.3: Planning results for narrow spaces. Our approach also works for accessing very narrow spaces such as the courtyard in (a) or the atrium in (b). In (c), a close-up view shows that the atrium is separated by a thin "wall" from the rest of the scene, which is actually a bridge on top of the building. This setting shows an inherent limitation of using 2.5D DSM representations for planning, as the atrium cannot be accessed when setting the flight altitude limit too low.

### 5.1.3 View Planning based on Geometric Priors

We have seen that geometric priors can be used to plan rough, suitable paths through urban environments. For enhancing large-scale geometric priors with more detailed close-up imagery, or when capturing scenarios where the environment changes considerably over time such as at construction sites, not only path planning but view planning is desired.

The goal of path planning as defined before is to get from a start to a target position with as little costs as possible. In comparison, during view planning we do not sample random nodes in configuration space but maximize the coverage and the reconstruction quality measures. When selecting views according to our algorithm as defined in Section 3.5.2, we first initialize our search space with a geo-referenced target surface given as a triangular mesh and create cameras that observe the mesh. All views that are not feasible to approach with the MAV are removed. Then, we cluster the created cameras to reduce the search space and incrementally select camera positions according to our quality measures until the desired reconstruction accuracy is reached. As a result we obtain a set of camera poses which forms the nodes of our view planning graph.

We use a greedy path optimization approach to obtain a view sequence with short paths in between. We start with the fully connected graph and remove all edges that



(a) (b)

Figure 5.4: View planning results. (a) Base mesh and computed camera network (blue) with optimal flight path for an MAV (red). (b) Sparse reconstruction result of a single family house after acquiring images using an autonomously flying MAV.

intersect with the prior geometry. Every remaining edge is given a weight corresponding to the Euclidean distance of the nodes. Changes in altitude are penalized by a factor of ten for two reasons: On the one hand changes in altitude consume more power, on the other hand these maneuvers are harder to monitor by the safety pilot. For the same reasons we add a penalty cost to direction changes of the MAV, which is based on the yaw angle.

Finally, given the weighted graph we start at the node with the lowest height over ground and greedily plan paths to all unvisited nodes using D* and choose the path with least costs. Figure 5.4 shows the resulting flight path for 133 views, and a reconstruction after autonomous flight according to our flight plan. This real-world outdoor example illustrates that the computed camera positions can be approached safely in autonomous flight mode by the MAV and all visible parts of the scene can be reconstructed.

### 5.1.4 Summary

We have shown that digital surface model can be used as prior knowledge for high-level path planning. Furthermore, we discussed the applicability of our view planning algorithm and showed that autonomous image acquisition with an MAV is indeed feasible. An evaluation of the reconstruction quality and densified results can be found in Section 6.3.1.

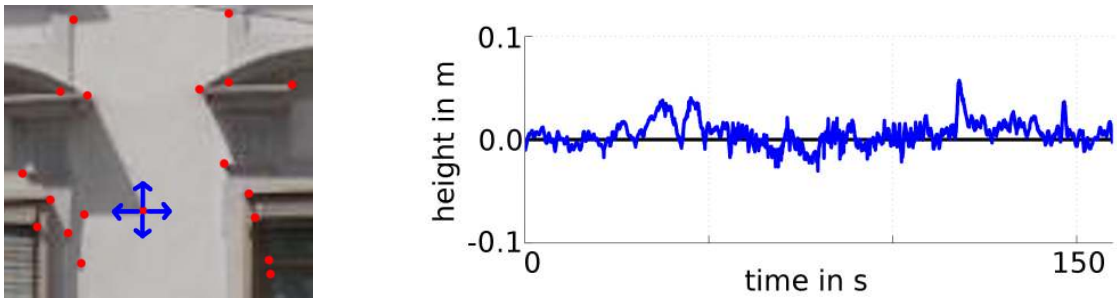
## 5.2 Trajectory Control based on Visual Localization

Off-the-shelf MAVs such as the Ascending Technologies Pelican, Falcon, and Parrot AR.Drone quad-rotors used in our work already feature an attitude controller based on IMU and accelerometer readings. Although these controllers make it possible to keep the

MAVs in a hovering state, any drift caused by accumulated error can only be detected based on exteroceptive sensors.

Visual pose estimation overcomes this issue. We use a single monocular camera and our set of localization approaches presented in Chapter 4 to estimate the pose of the MAV. Visual navigation of an MAV requires the correct scale of the prior models to be known, which is typically the case when we employ metric, aligned visual landmarks. When starting in an unknown environment or during our indoor experiments, we have solved this issue by placing a known ARToolKitPlus marker [215] into the scene during initialization. We thus recover a metrical baseline, which leads to correct metrical scaling of the entire map.

Visual localization typically delivers good pose estimates within a range of a few centimeters, depending on the distance to the scene. However, the position estimate can considerably jump within that range because of the error propagation of quantized feature measurements. This has a major effect on control: The derivative of the position, i.e. the estimated velocity, suffers from heavy salt-and-paper noise as depicted in Figure 5.5.




(a) (b)

Figure 5.5: Characteristics of visual pose noise. (a) The position of interest points is subject to noise which nonlinearly affects the pose estimate. (b) Visual poses are thus subject to noise that resembles salt-and-pepper noise more than Gaussian noise.

The following sections give an overview of our fuzzy control approach to this problem and discuss the advantages with respect to related work. Our system design incorporates several properties of the flight hardware, the visual perception methodology, and the application of airborne sensing.

### 5.2.1 Related Work

Different approaches have been presented to resolve this issue with visual pose input. Kemp [108] maintains a set of multiple smooth position and velocity estimates, where the best is selected based on likelihood and processed using simple PID control. This is similar to particle filtering and thus requires a lot of additional processing power. Blösch et al. [21]

and Achtelik et al. [2] try to accurately model the system in a Linear Quadratic Gaussian control design with Loop Transfer Recovery (LQG/LTR). They incorporate additional sensor information from IMU, accelerometers, and an air pressure sensor to cope with pose estimation uncertainties. While this fusion is beneficial, it requires considerable system knowledge about the aerial vehicle and restricts the flexibility. In contrast, our fuzzy control approach allows to interactively change payloads based on the application, but also to change the set of rules based on the current scene and the corresponding visual pose estimation quality.

Many airborne sensing and inspection applications require a camera which looks at the scene rather than on the ground. We have recently shown that this can be achieved with a single camera by interleaving detailed and overview images in a stream [140]. In contrast to most existing approaches [2, 21], we do not require a nadir camera view; we do not even require a rigid transformation between the camera and the MAV due to the cascaded controller. Our system only employs the 3D position and the yaw with respect to the scene for control, so mechanical stabilization of the camera is possible and the camera's pitch angle can be arbitrarily defined.

### 5.2.2 Position-based Visual Servoing with a Fuzzy Controller

Our position controller is built on top of a high-speed attitude controller [73] which is inherent to the MAV. Our controller consists of three identical PID fuzzy logic controllers for the individual axes and a proportional controller to maintain the orientation of the MAV. An overview can be found in Figure 5.6. In the following, the controller for the $x$ axis is described.

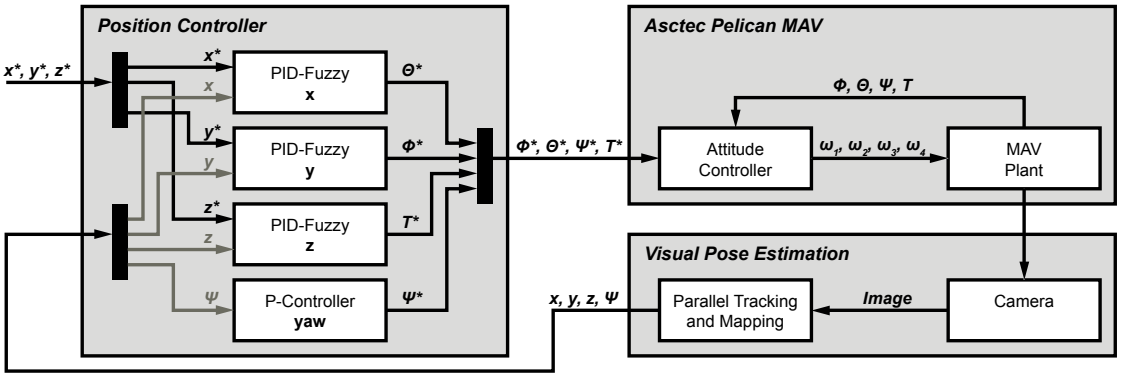
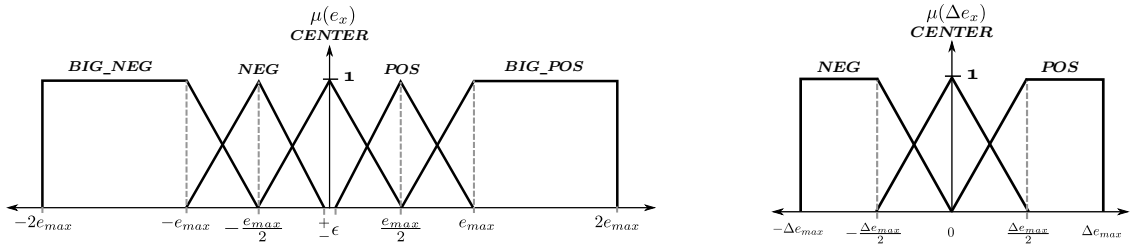



Figure 5.6: Fuzzy visual servoing. Our system consists of a cascaded control loop with an attitude controller and a position controller. Feedback is received using a camera mounted on the MAV and subsequent processing by our variant of PTAM [113, 223]. The position controller consists of three independent PID fuzzy logic controllers and a simple proportional controller to maintain the orientation of the MAV.




(a) (b)

Figure 5.7: Fuzzification of (a) the position error $e_x$ and (b) the error change $\Delta e_x$. The linguistic variables BIG_NEG, NEG, CENTER, POS, and BIG_POS are the membership functions to estimate the truth value of the fuzzy set. Note the dead zone $\pm\epsilon$ which is specifically added to avoid problems with visual pose estimation.

Based on the desired position $x^*$ and the current position $x$, the error $e_x$ and the error change $\Delta e_x$ can be calculated. Thus, two variables serve as controller input, whereas the output is the desired pitch angle $\Theta^*$. The error change

$$\Delta e_x[n] = e_x[n] - e_x[n-1], \tag{5.2}$$

where $n$ is the current time, gives a rough estimate for the velocity error. The desired pitch angle $\Theta^*$ is obtained by integrating the relative change of the pitch angle over time, i.e.

$$\Theta[n]^* = \Theta[n-1]^* + \Delta\Theta[n]. \tag{5.3}$$

The integration part is necessary to compensate MAV drifts along the roll angle. Moreover, the temporal integration of the fuzzy logic output enables to cope with external influences such as wind, battery drain or turbulences.

A typical fuzzy logic controller consists of three blocks, namely fuzzification, inference and defuzzification. First, the two inputs and the output have to be fuzzificated, meaning that the measured values are mapped to the truth values $\mu(e_x)$ and $\mu(\Delta e_x)$ of the fuzzy set (Figure 5.7). The width of the linguistic variables depends on the parameters $e_{max}$ and $\Delta e_{max}$ which are tuned during test flights. The major advantage of fuzzy logic is that the design of the membership functions can be chosen intuitively. We therefore add a dead zone $\epsilon$ around the hovering position (Figure 5.7(a)) which reflects the estimated noise level of visual pose estimation based on the distance to the scene.

Next, the rule-set of the inference block as shown in Table 5.1 is applied; it is defined in a way a human expert would control the pitch $\Theta$ when manually flying the MAV. It can be read in a linguistic way as

**IF** $e_x$ is *BIG_NEG* **AND** $\Delta e_x$ is *NEG* **THEN** $\Delta\Theta$ is *BIG_POS*,

Table 5.1: Fuzzy logic rules for determining the output based on the current position error $e_x$ and velocity error $\Delta e_x$.

| $\Delta e_x \backslash e_x$ | *BIG_NEG* | *NEG* | *CENTER* | *POS* | *BIG_POS* |
|---|---|---|---|---|---|
| *NEG* | *BIG_POS* | *POS* | *POS* | *CENTER* | *CENTER* |
| *CENTER* | *POS* | *POS* | *CENTER* | *NEG* | *NEG* |
| *POS* | *CENTER* | *CENTER* | *NEG* | *NEG* | *BIG_NEG* |

in other words if the position error is large and negative, and the velocity error is also negative then the desired output is large and positive. Based on the five membership functions of the first input $e_x$ and three of the second input $\Delta e_x$, fifteen rules exist to control the $x$-position of the MAV.

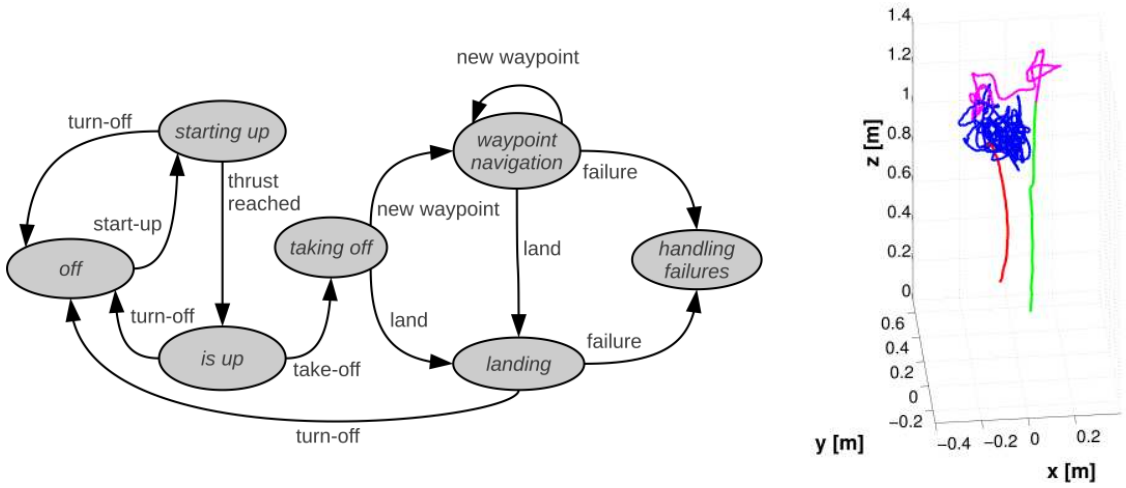
Finally, the last block of the fuzzy logic control is the defuzzification. The truth values of the output membership functions are mapped to a real value for the system input. In case of the $x$-position controller the change of the pitch angle is computed. We use the Center-of-Area method for defuzzification, where the center-of-gravity of the area of the composited output function is mapped to the $x$-axis.

### 5.2.3 Autonomous Take-off, Hovering, and Landing

Visual servoing is especially beneficial when high positioning accuracy is necessary, namely during take-off, landing, and when hovering close to objects. A state machine demonstrating the individual steps and a resulting trajectory is shown in Figure 5.8.

For autonomous take-off, we first acquire a sparse map of the area around the starting point by manually translating the vehicle while looking towards the marker. As the marker can be tilted, the reference coordinate system for navigation is set with an $xy$-plane parallel to the ground, based on the IMU readings of the MAV. Finally, the first waypoint is set just above the starting point, and the MAV increases the thrust until it starts climbing and receives the first valid pose estimate. Typically, the missing accurate system model leads to a drift compensation phase where the correct parameters are automatically estimated. Afterwards, the MAV automatically hovers at the commanded waypoint until it receives a new desired position. For landing, we decrease the thrust of the MAV so that a visually supervised decline is initiated. Once the map is lost close to the ground, the MAV continues to decline with the same thrust until it touches the ground.

**Failure Handling.** While our visual pose estimation algorithms are implemented in a robust way, sudden illumination changes as well as too homogeneous regions resulting in too few features cause problems. Our localization framework automatically detects those cases based on the effective inlier measure (Section 4.2.2). Invalid pose estimates due to repetitive structures are harder to detect with vision alone. We compute a speed




(a) (b)

Figure 5.8: Autonomous take-off, hovering, and landing. (a) State machine for autonomous flight. (b) MAV trajectory during take-off (green), drift compensation (magenta), hovering (blue) and landing (red).

estimate for the change between two poses and reject all measurements which exceed the maximal known speed of the vehicle. Further, we employ the verification step discussed in Section 4.2.3 which is based on a comparison to IMU and GPS measurements.

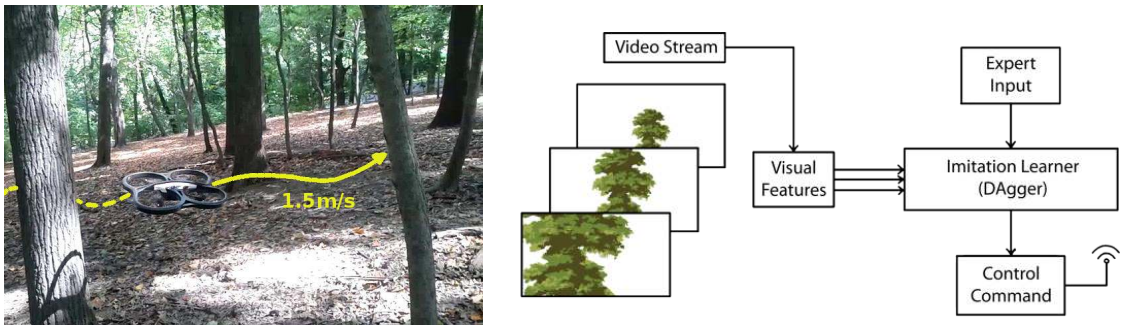
System failures are reported to the operator using audio-visual signals. Depending on the estimated GPS quality and altitude the vehicle either switches to the GPS position hold mode, or it lands autonomously using the technique described above. At any time, the operator can take over using the remote control.

### 5.2.4 Summary

We have presented a robust visual servoing system for MAVs based on monocular camera input and a fuzzy logic controller [224]. The system is well suited for aerial sensing tasks because the camera's pitch angle can be arbitrarily defined and is not restricted to nadir views. The incorporation of a fuzzy controller allows to quickly change payloads as no model knowledge about the MAV is required, and it tolerates quickly changing velocity measurements which are commonly observed in visual pose estimation. Thanks to our distributed localization and mapping architecture presented in Section 4.3, the controller can incorporate full-framerate pose updates and can run on–board.

## 5.3 Imitation Learning for Reactive Visual Control

Visual input is primarily beneficial when navigating MAVs that have very low payload capabilities, and operate close to the ground where they cannot avoid dense obstacle fields. In such a setting, situations or environments might be encountered where 3D mapping combined with high-level trajectory planning [14] fails. Reactive control tackles this issue by working on minimally preprocessed visual input (i.e. no keypoints, matching, or triangulation) and thus provides the lowest-level layer for autonomous control.




(a) (b)

Figure 5.9: Reactive control overview. We present a novel method for high-speed, autonomous MAV flight through dense forest areas. The system is based on purely visual input and imitates human reactive control.

Reactive behavior needs to be customized to fit a certain task or environment. In this section we present a generic approach to learn a reactive control policy through imitation of a human pilot, but apply our technique to high-speed flight through a cluttered forest environment (Figure 5.9). As in the rest of this thesis we rely on passive monocular vision as the only exteroceptive sensor, and we exploit prior knowledge in form of control commands given by a human pilot as a reaction to certain visual stimulus.

### 5.3.1 Related Work

Relatively simple yet efficient algorithms for reactive navigation can be derived when imitating animals and insects, who use optical flow for navigation [190]. Beyeler et al.[17] as well as Conroy et al. [29] implemented systems which exploit this fact and lead to good obstacle avoidance results. Later, Lee et al. [129] proposed to use a probabilistic method of computing optical flow for more robust distance calculation to obstacles for MAV navigation. Optical flow based controllers navigate by balancing flow on either side. However flow captures richer scene information than these controllers are able to use. We embed flow in a data-driven framework to automatically derive a controller which exploits this information.

Most closely related to our method are approaches which learn motion policies and depth from input data. Michels et al. [144] demonstrated driving a remote-controlled toy car outdoors using monocular vision and reinforcement learning. Hadsell et al. [75] showed in the LAGR project how to recognize and avoid obstacles within complex outdoor environments using vision and deep hierarchical networks. Bill et al. [18] used the often orthogonal structure of indoor scenes to estimate vanishing points and navigate an MAV in corridors by going towards the dominant vanishing point. We extend those approaches and employ a novel imitation learning technique that allows us to find a collision-free path through a forest despite the diverse appearance of visual input.

### 5.3.2 Learning to Imitate Reactive Human Control

Visual features extracted from camera input provide a rich set of information that we can use to control the MAV and avoid obstacles. However, programming a controller by hand using all this information would be a very time consuming and daunting task. Therefore, we leverage state-of-the-art imitation learning techniques [6, 162, 166, 171, 178] to learn such a controller. These data-driven approaches allow us to directly learn a control strategy that mimics an expert pilot's choice of actions based on demonstrations of the desired behavior, i.e., sample flight trajectories through cluttered environments.

The traditional imitation learning approach is formulated as a standard supervised learning problem similar to for instance spam filtering, in which a corpus of training examples is provided. Each example consists of an environment (an image acquired by the MAV) and the action taken by an expert in that same environment. The learning algorithm returns the policy that best mimics the expert's actions on these examples. The classic successful demonstration of this approach in robotics is that of *ALVINN* (Autonomous Land Vehicle in a Neural Network) [162] which demonstrated the ability to learn highway driving strategies by mapping camera images to steering angles.

While various learning techniques have been applied to imitation learning [6, 166, 178], these applications all violate the main assumption made by statistical learning approaches that the learner's predictions (actions) do not affect the distribution of inputs. As shown in previous work [170] and confirmed in the MAV setting here, ignoring the effects of the learner on the underlying state distribution leads to serious practical difficulties and poor performance. For example, during typical pilot demonstrations of the task, trees are avoided fairly early and most training examples consist of straight trajectories with trees on the side. However, since the learned controller does not behave perfectly, the MAV encounters situations where it is directly heading for a tree and is closer than the human ever was. As the hard turns it needs to perform in these cases are nonexistent in the training data, it simply cannot learn the proper recovery behavior.

Theoretically, [170] showed that even if a good policy that mimics the expert's actions well on the training examples is learned, when controlling the MAV, its divergence from the correct controls could be much larger (by as much as a factor $T$, when executing for

$T$ timesteps) due to the change in environments encountered under its own controls.

Fortunately, Ross et al. [171] proposed a simple iterative training procedure called DAgger, for Dataset Aggregation, that addresses this issue and provides improved performance guarantees. Due to its simplicity and practicality, we use this approach to learn the controller for our aerial vehicle. While [171] demonstrated successful application of this technique in simulated environments in video games, our experiments show that this technique can also be successfully applied on real robotic platforms. We briefly review the DAgger algorithm below.

**The DAgger Algorithm.** DAgger trains a policy that mimics the expert's behavior through multiple iterations of training. Initially, the expert demonstrates the task and a first policy $\pi_1$ is learned on this data by solving a classification or regression problem. Then, at iteration $n$, the learner's current policy $\pi_{n-1}$ is executed to collect more data about the expert's behavior. That is, in our particular scenario, the MAV executes its own controls based on $\pi_{n-1}$, and as it is flying, the pilot provides the correct actions to perform in the environments the MAV visits, via a joystick. This allows the learner to collect data in new situations which the current policy might visit, but which were not previously observed under the expert demonstrations, and learn the proper recovery behavior when these are encountered. The next policy $\pi_n$ is obtained by training a policy on all the training data collected over all iterations (from iteration 1 to $n$). This is iterated for a small number of iterations $N$ and the best policy found at mimicking the expert under its induced distribution of environments is returned.

The intuition is that, over the iterations, we collect a set of inputs the learner is likely to observe during its execution based on previous experience (training iterations), and obtain the proper behavior from the pilot in these situations. [171] showed theoretically that after a sufficient number of iterations, DAgger is guaranteed to find a policy that when executed at test time, mimics the expert at least as well as how it could do on the aggregate dataset of all training examples. Hence the divergence in controls is not increased by a factor $T$ as in the traditional supervised learning approach when the learned policy controls the MAV.

For our application, we aim to learn a linear controller of the vehicle's left-right velocity that mimics the pilot's behavior to avoid trees as the MAV moves forward at fixed velocity and altitude. That is, given a vector of visual features $x$ from the current image, we compute a left-right velocity $\hat{y} = w^\top x$ that is sent to the MAV, where $w$ are the parameters of the linear controller that we learn from the training examples. To optimize $w$, we solve a ridge regression problem at each iteration of DAgger. Given the matrix of observed visual features $X$ (each row is an observed feature vector), and the vector $y$ of associated left-right velocity commands by the pilot, over all iterations of training, we solve $w = (X^\top X + R)^{-1} X^\top y$, where $R$ is a diagonal matrix of per-feature regularization terms. We choose to have individual regularization for different types of features, which might represent different fractions of the feature vector $X$, so that every type contributes equally

to the controls. In other words, we regularize each feature of a certain type proportionally to the number of features of that type. Features are also normalized to have zero mean and unit variance, based on all the observed data, before computing $w$. When controlling the vehicle, $w$ is applied to normalized features again.

### 5.3.3 Reactive Control based on Monocular Visual Input

Our approach learns a controller that maps RGB images from the on-board camera to control commands. This requires mapping camera images to a set of features which can be used by DAgger. These visual features need to provide indirect information about the three-dimensional structure of the environment. Accordingly, we focused on features which have been shown to correlate well with depth cues such as those in [144], specifically Radon transform statistics, structure tensor statistics, Laws' masks and optical flow.

We compute features over square windows in the image, with a 50% overlap between neighboring windows. The feature vectors of all windows are then concatenated into a single feature vector. The choice of the number of windows is driven primarily by computational constraints. Using a $15 \times 7$ discretization (in $x$ and $y$ respectively) performs well and can be computed in real-time.

**Radon Features (30 dim.)** The Radon transform [82] of an image is computed by summing up the pixel values along a discretized set of lines in the image, resulting in a 2D matrix where the axes are the two parameters of a line in 2D, $\theta$ and $s$. We discretize this matrix in $15 \times 15$ bins, and for each angle $\theta$ the two highest values are recorded. This encodes the orientations of strong edges in the image.

**Structure Tensor Statistics (15 dim.)** At every point in a window the structure tensor [77] is computed and the angle between the two eigenvectors is used to index a 15-bin histogram for the entire window. The corresponding eigenvalues are accumulated in the bins. In contrast to the Radon transform, the structure tensor is a more local descriptor of texture. Together with Radon features the texture gradients are captured, which are strong monocular depth cues [231].

**Laws' Masks (8 dim.)** Laws' masks [36] encode texture intensities. We use six masks obtained by pairwise combinations of one-dimensional masks: (L)evel, (E)dge and (S)pot. The image is converted to the YCrCb colorspace and the LL mask is applied to all three channels. The remaining five masks are applied to the Y channel only. The results are computed for each window and the mean absolute value of each mask response is recorded.

**Optical Flow (5 dim.)** Finally, we compute dense optical flow [227] and extract the minimum and maximum of the flow magnitude, mean flow and standard deviation in $x$ and $y$. Since optical flow computations can be erroneous, we record the entropy of the

flow as a quality measure. Optical flow is also an important cue for depth estimation as closer objects result in higher flow magnitude.

Useful features must have two key properties. First, they need to be computed fast enough. Our set of features can be computed at 15 Hz using the graphics processing unit (GPU) for dense optical flow computation. Although optical flow is helpful, we show in our experiments that removing this feature on platforms without a GPU does not harm the approach significantly. Secondly, the features need to be sufficiently invariant to changes between training and testing conditions so that the system does not overfit to training conditions. We therefore refrained from adding color features, as considerable variations under different illumination conditions and confusions between trees and ground, as well as between leaves and grass, might occur. An experimental evaluation of the importance of every feature is given in Section 6.3.3, along with a detailed evaluation.

In addition to visual features, we append 9 additional features: The low-pass filtered history of previous commands at 7 different exponentially decaying time steps, the sideways drift measured by the on-board IMU, and the deviation in yaw from the initial direction. Previous commands encode past motion which helps to smooth the control output. The drift feature provides context to the pilot's commands and accounts for motion caused by inertia. The difference in yaw is meant to reduce drift from the initial orientation.

### 5.3.4 Usability Challenges when Learning Control Commands

When iteratively learning control commands, one faces several human-computer-interaction challenges in practice. Figure 5.10 shows the DAgger control interface used to provide correct actions to the MAV. Note that, at iteration $n$, the learner's current policy $\pi_{n-1}$ is in control of the MAV and the expert just provides the correct controls for the scenes that the MAV visits. The expert controls are recorded but not executed on the MAV. The three major usability challenges and our solutions are discussed in the following.

After the first iteration, the pilot must be able to provide the correct actions without feedback of how the MAV would react to the current command. While deciding whether the vehicle should go left or right is easy, it can be hard to input the correct magnitude of the turn the MAV should perform without feedback. In particular, we observed that this often makes the pilot turn excessively when providing the training examples after the first iteration. Performance can degrade quickly if the learner starts to mimic these imperfect actions. To address this issue, we provided partial feedback to the pilot by showing a vertical line in the camera image seen by the pilot that would slide left or right based on the current joystick command performed (Figure 5.10). As this line indicated roughly where the MAV would move under the current command, this led to improved actions provided by the pilot.

In addition to the lack of feedback, providing the correct actions in real-time after

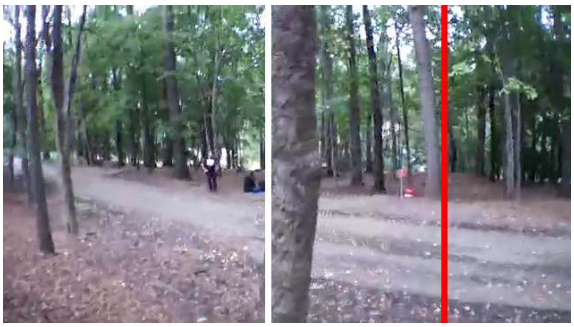


Figure 5.10: Usability challenges. In this exemplary frame from the MAV camera stream, the white line indicates the yaw commanded by the current DAgger policy $\pi_{n-1}$ while the red line indicates the yaw commanded by the expert. In this frame DAgger is wrongly heading for the tree in the middle while the expert is providing the correct yaw command to go to the right instead. These expert controls are recorded for training later iterations but not executed in the current run.

the first iteration when the MAV is in control can be hard for the pilot as he must react to what the vehicle is doing and not what he expects to happen. For instance, if the MAV suddenly starts turning towards a tree nearby, the pilot must quickly start turning the other way to indicate the proper behavior. The pilot's reaction time to the vehicle's behavior can lead to extra delay in the correct actions specified by the pilot. By trying to react quickly, he may provide imperfect actions as well. This becomes more and more of an issue the faster the MAV is flying. To address this issue, we allowed the pilot to indicate the correct actions offline while the camera stream from the MAV is replayed at a speed 3 times slower than real-time, using the interface seen in Figure 5.10. By replaying the stream slower, the pilot can provide more accurate commands and react more quickly to the robot's behavior. The faster the MAV is flying, the slower the trajectory should be replayed to provide good commands in time.

The third challenge is that DAgger requires to collect data for all situations encountered by the current policy in later iterations. This would include situations where the MAV crashes into obstacles if the current policy is not good enough. For safety reasons, we allow the pilot to take over or force an emergency landing to avoid crashes as much as possible. This implies that the training data used is not exactly what DAgger would require, but instead a subset of training examples encountered by the current policy when it is within a "safe" region. Despite this modification, the guarantees of DAgger still hold as long as a policy that can stay within this "safe" region can be learned.

### 5.3.5 Summary

We have presented a novel approach for high-speed, autonomous MAV flight through cluttered environments [172]. Our system learns to predict how a human expert would control the aircraft in a similar situation, and thus successfully avoids collisions using

passive, low-cost and low-weight visual sensors only. We have applied a novel imitation learning strategy which takes into account that the MAV is likely to end up in situations where the human pilot does not, and thus needs to learn how to react in such cases. To evaluate our approach, we have performed extensive outdoor experiments which are presented in the next chapter.

# Chapter 6

# Experiments and Results

**Contents**



We have presented a set of techniques for reconstructing geometric prior knowledge, for scalable visual localization within such geometric priors, and for computer vision techniques related to path planning and control of micro aerial vehicles. In this chapter, we provide quantitative and qualitative experiments and results for all parts of our work.

**Experimental Platforms.** In the course of this thesis, three types of micro aerial vehicles with individual properties and payloads have been used. An overview is given in Figure 6.1.

First, the Ascending Technologies *Falcon* octo-rotor MAV is mainly used for obtaining high-resolution, low-framerate aerial imagery. Equipped with a mechanically stabilized 10M pixels Panasonic Lumix DMC-LX3 consumer camera and a 24 mm lens, this MAV is designed for single image acquisition and offline processing. However, we have added an SD storage card featuring ad-hoc WiFi connections and a WiFi repeater to enhance the connection range. This setup enables us to send the obtained imagery to a notebook on the ground, which allows us to process the imagery in an online fashion at about 0.2 fps.

Second, the Ascending Technologies *Pelican* quad–rotor MAV is equipped with a single IDS UI-1240SE industrial camera with a resolution of $1280 \times 1024$ pixels, global shutter, and a 12 mm lens. Thanks to the 1.6 GHz Intel Atom on–board computer running the Robot Operating System (ROS) and a wireless 802.11n link, we are able to process imagery on-board and just transmit the results (e.g. the visual pose information and selected keyframes), or to stream full-resolution images to the ground at 4 fps.

(a) (b) (c)

Figure 6.1: Aerial platforms for experiments. We use three different types of micro aerial vehicles during our experiments, namely (a) an Ascending Technologies *Falcon* octo-rotor with a consumer camera, (b) an Ascending Technologies *Pelican* quad-rotor with on-board processing capabilities, and (c) a low-cost Parrot *ARDrone* quad-rotor with video streaming capabilities.

Finally, the Parrot *ARDrone* MAV is distributed as a toy and costs 20 times less than the other two MAVs. It weights only 420g and has a size of $0.3 \times 0.3$m. The system features a front-facing camera with $320 \times 240$ pixels and a 93 deg field of view (FOV), an ultrasound altimeter, a low resolution down-facing camera and an IMU. The MAV's built-in on-board position controller allows control through high-level desired velocity commands (forward-backward, left-right and up-down velocities, as well as yaw rotation). The *ARDrone* can reach a maximum velocity of about 5 m/s. Communication is based on WiFi, with camera images streamed at about 10-15 Hz. This allows us to control the MAV on a separate computer that receives and processes the images and then sends corresponding commands at around 10 Hz.

## 6.1 Modeling Geometric Prior Knowledge

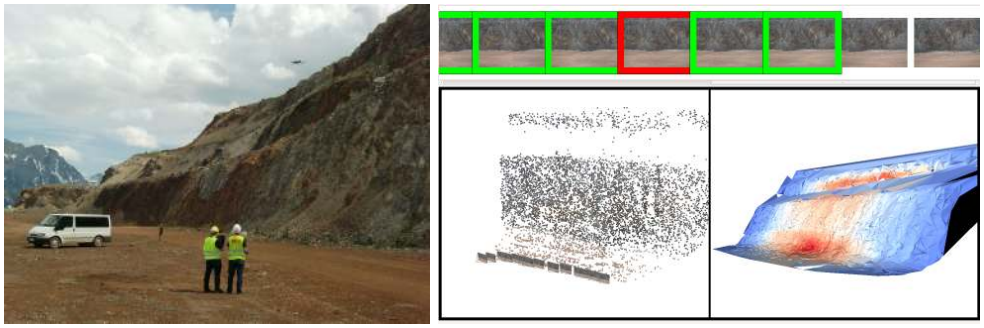
In the following, we evaluate the accuracy of our 3D reconstruction pipeline and demonstrate its applicability to real-world problems. The methodology presented in Chapter 3 is shown to deliver state-of-the-art results.

### 6.1.1 Point-based Scene Reconstruction

We evaluate our 3D multi-view reconstruction methods based on a dataset recorded in a stone quarry. The recorded quarry wall with a height of about 24 m and a length of about 100 m contains a set of markers which have been accurately localized using a total station, and thus provides a reliable ground truth for evaluation.

**Surveying Flight.** The data used for this experiment has been recorded while flying the *Falcon* MAV equipped with a consumer camera (Figure 6.2(a)). Within two flights of

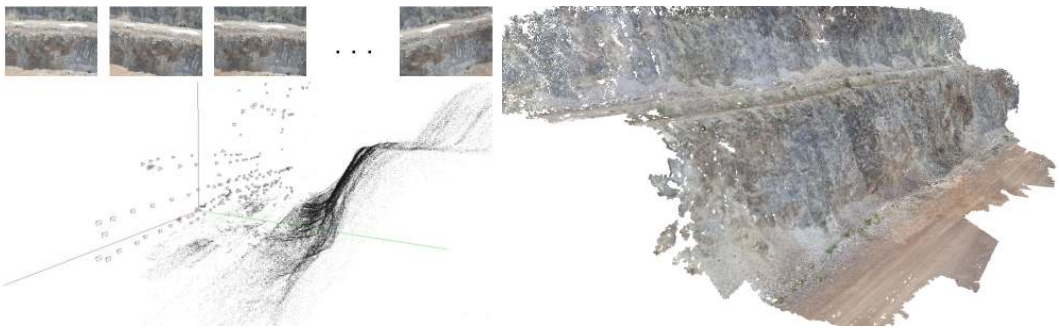
about 30 min total duration, we recorded 294 images with a resolution of 10 Mpx each. We aimed for a ground sampling distance of about 1 cm per pixel and chose our acquisition strategy accordingly. Given a focal length of 24 mm we took pictures from a distance of about 25 m between camera and surface, maintaining a vertical overlap of 80% and a horizontal overlap of 30%. To reduce the error propagation between neighboring views, we additionally acquired several overview images from a larger distance to the quarry wall.



(a) (b)

Figure 6.2: Surveying flight at Erzberg, Austria. (a) Acquisition of 294 images of a quarry wall. (b) Online Structure-from-Motion for visualizing the reconstruction result and some quality measures in real-time. The pilot receives information whether the images were integrated in the reconstruction (green border) or not (red border). Apart from the point cloud, a rough surface model is created and textured with the redundancy measure. Red color indicates that the area has been seen several times, while areas depicted in blue still lack some measurements.

**Reconstruction Result.** Given the stream of images taken approximately every 5 seconds and a known camera calibration, we applied our online Structure-from-Motion approach as presented in Section 3.1.5 to lower resolution images. The results are on the one hand the extrinsic parameters of the cameras, on the other hand a sparse point cloud showing triangulated interest points. The user interface as shown in Figure 6.2(b) allows to get an impression of the completeness of the reconstruction while the MAV is airborne, which is a huge benefit compared to offline methods. For increased quality, the imagery is again processed in full resolution later on. As can be seen in Figure 6.3, the resulting sparse model already contains a large number of points due to the well-textured scene. After densification using PMVS [65] the model contains even more geometric details and more than 10 million 3D points. In contrast to terrestrial image acquisition the MAV can take pictures of the upper regions of the quarry wall as well, so the entire profile can be reconstructed. Being able to extract such additional information is especially important for mining applications, for instance the planning of rock blasting operations.



(a) (b)

Figure 6.3: Reconstruction of a quarry wall (a) Extrinsic camera orientations and triangulated 3D points. The sparse point cloud consists of about 130.000 points. (b) After densification using PMVS [65], the point cloud consists of about 10 million points.

**Accuracy Evaluation.** In computer vision, the reprojection error of triangulated interest points is commonly used as a measure for the accuracy of a reconstruction. Our reconstruction pipeline optimizes for the reprojection error during bundle adjustment, thus we obtain at least a local optimum. However, in a setting where the reconstruction quality influences the localization quality later on, the photogrammetric approach of computing the geometric error to ground control points (GCPs) is more suitable. A set of 77 prism targets has been attached to the rock, and the position of every target has been measured with a total station. Figure 6.4 depicts the distribution of the GCPs. The targets are numbered and can be clearly distinguished from the rock by color. We were thus able to recognize and manually label 74 out of 77 markers in the densified point cloud. As a result, we retrieve the respective 3D coordinates for the markers in a local coordinate system.

Structure-from-Motion reconstructions are always up to scale, meaning that an unknown scale factor has to be estimated for comparing the reconstruction with ground truth data. The scale factor can be computed given the metric distance between two cameras, or as in our case the distance between two points in the scene. For our evaluation, we are interested in the relative and absolute accuracies of the reconstructed ground control points given two different lengths for scaling, as depicted in Figure 6.4. First, we select two points which are rather close to each other ($d_1 = 6.92m$), then we select two GCPs with a large distance ($d_2 = 52.38m$). While the second case seems more reasonable for accurate scaling, the first case is more user-friendly.

For both cases we evaluate the relative distances between all pairs of points, and depict the error between ground truth and reconstruction over the relative distance in ground truth. Given an inconsistent reconstruction, the errors at a large distance would increase compared to small distances. Further, we evaluate the influence of camera calibration on

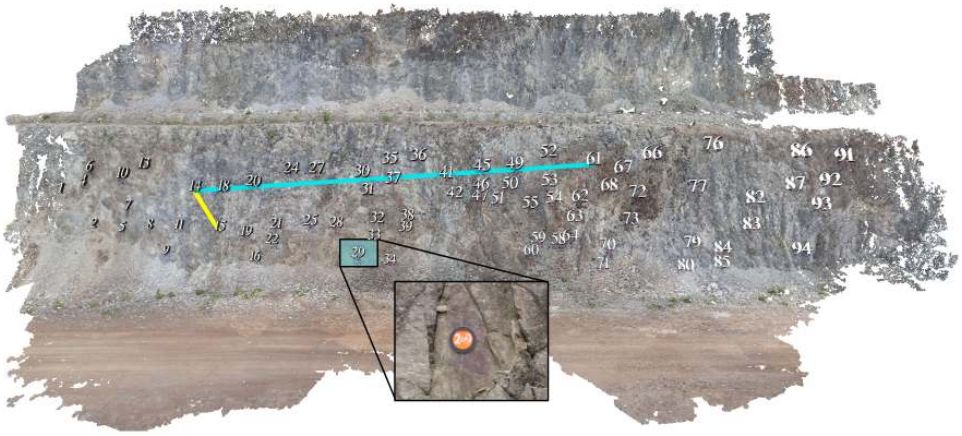
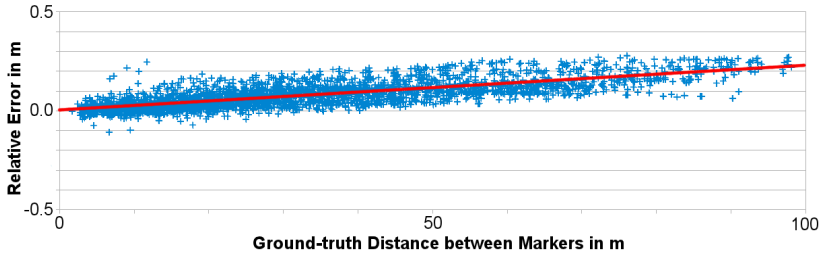


Figure 6.4: Layout of the ground control points on the quarry wall. The position of every prism target has been measured using a total station, and the targets have been numbered. The known metric distances of GCP 14-15 (small distance, yellow) and of GCP 14-61 (long distance, blue) are used for scaling the resulting reconstruction.

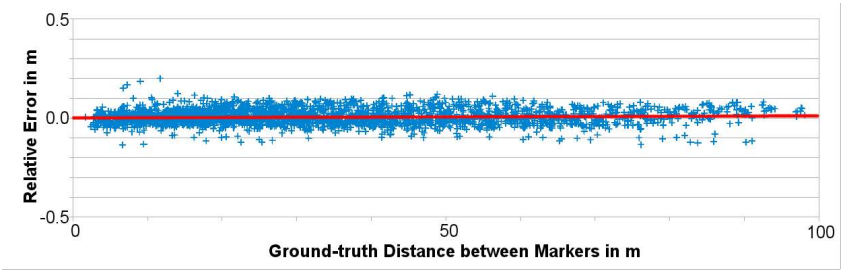
the reconstruction accuracy by comparing the reconstruction obtained with pre-calibrated parameters with a reconstruction where the intrinsic parameters were optimized during bundle adjustment. The results can be seen in Figure 6.5.

When scaling with a short known distance $d_1$ and precomputed intrinsics, the relative error increases considerably with the distance between the two points (Figure 6.5(a)). If the intrinsic parameters are optimized during reconstruction, the error remains approximately constant over the distance (Figure 6.5(b)). We derive that a high-quality camera calibration is important to avoid error propagation. When scaling with a longer known distance $d_2$ the effect of camera calibration is less significant (Figure 6.5(c) and 6.5(d)). The standard deviation of the relative errors is 3.8 cm, which means that 99.7% of all measurements are within an error margin of $\pm 11.4$ cm.

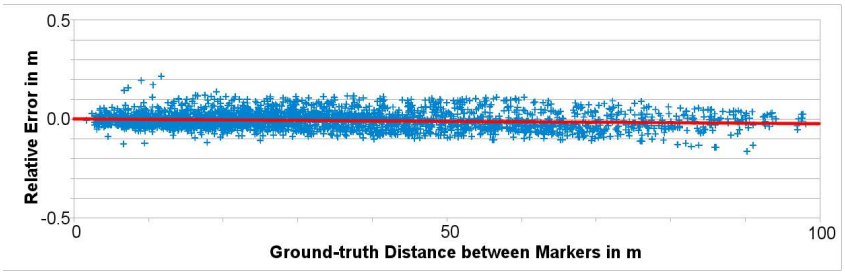
To recover the absolute reconstruction accuracy for the GCPs in scene coordinates, the 3D point cloud needs to be transformed in the reference coordinate system using a least-squares similarity transform. Figure 6.6 shows the resulting absolute errors between GCPs and reconstructed positions in the model. The visualization shows that the error in the central part of the quarry wall is in the order of 5 cm on average, while the error increases towards the borders of the model. The reason for this behavior is the lack of imagery along the border, which causes less overlap and thus less measurements and pairwise constraints per point. As a result, the optimization routine in bundle adjustment is able to distort these points when optimizing the reprojection error. The issue can be circumvented by selecting a region for image acquisition which is larger than required, and cropping the borders of the model in the end. Even without this procedure the mean absolute error is 6.87 cm, which is comparable to the prism target positioning accuracy.



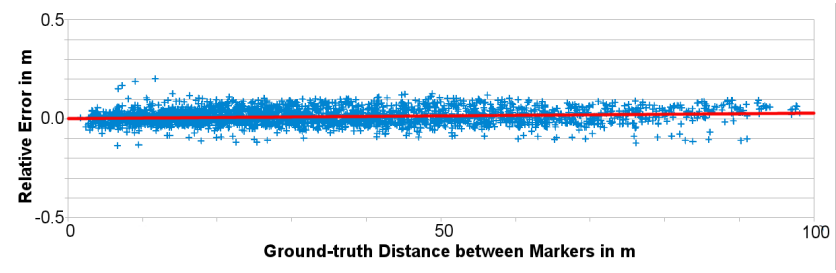

(a) Short known distance without intrinsic optimization




(b) Short known distance with intrinsic optimization




(c) Long known distance without intrinsic optimization




(d) Long known distance with intrinsic optimization

Figure 6.5: Evaluation of the relative reconstruction errors. Accurate intrinsics are especially beneficial when scaling with a short known distance, as the relative error over the entire model is significantly reduced. When optimizing the intrinsic camera parameters during bundle adjustment, the standard deviation of the relative errors is 3.8 cm.

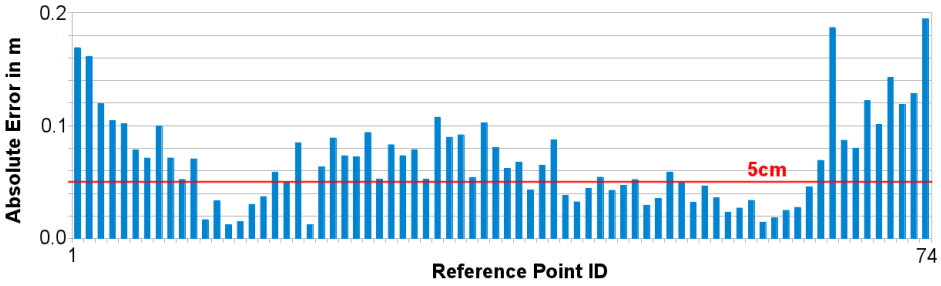



Figure 6.6: Evaluation of the absolute reconstruction errors. The error is measured between corresponding points after transforming the point cloud with a least-squares similarity transform. The intrinsic camera parameters are optimized for all images.

Given a proper acquisition strategy, our point-based reconstruction approach delivers not only visually pleasing but also very accurate results. The point clouds can be used for further densification steps or for line-model extraction, but also directly for localization within point-based geometric prior knowledge.

### 6.1.2 Line-based Scene Reconstruction

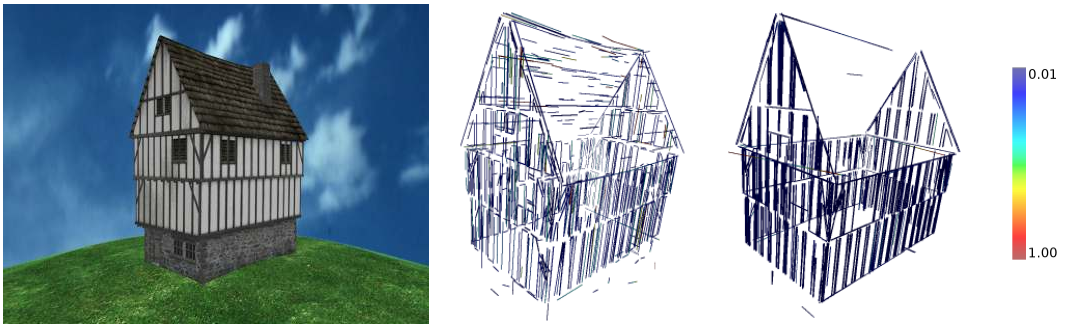
Line-based scene reconstruction has been shown to help in man-made environments with a low amount of distinctive interest points or when dealing with wiry structures. While qualitative examples for modeling a power pylon and solid objects have been shown in Section 3.2 already, we compare the algorithm to a benchmark dataset here.

**Ground-truth Evaluation.** In order to compare our approach to Jain et al. [102] we reconstruct their *Timberframe house* sequence[1] consisting of 240 synthetic images using our algorithm. Figure 6.7 shows exemplar views from the sequence along with our 3D reconstruction and the result of Jain et al., colored using the Hausdorff distance as similarity measure (for densely sampled points along the lines) to the ground truth model. We achieve a root mean square (RMS) error of 0.0013 units while their approach has an error of 0.0036 units compared to the CAD model.

It can be seen that both algorithms manage to reconstruct the building in a qualitatively accurate way. Our approach performs better in terms of the RMS error, while Jain et al. are able to reconstruct a few more lines, especially on the roof. Even though the resulting models are similar, the computation time differs considerably. The authors report that their algorithm needs several hours to deliver the result for a subset of 72 images, while our method reconstructs this sequence in 7.5 min using all 240 images.

**Performance Evaluation.** Since we have to evaluate many possible matches for each 2D line segment to avoid appearance-based matching our algorithm is more time con-

[1] http://www.mpi-inf.mpg.de/resources/LineReconstruction/




(a) (b) (c)

Figure 6.7: Qualitative evaluation of line-based reconstruction using the synthetic Timberframe house sequence (240 images). (a) Example view of the ground truth model. (b) Reconstruction by [102], and (c) our reconstruction. The color reveals the errors compared to ground truth (from 0.01 to 1.00).

| Sequence | #img | resolution | #raw | #final | $t_{LSD}$ | $t_{match}$ | $t_{group}$ | $t_{total}$ |
|---|---|---|---|---|---|---|---|---|
| Pylon | 106 | $5616 \times 3744$ | 71538 | 1381 | 17.2 | 15.5 | 34.1 | 66.8 |
| House | 93 | $3048 \times 2736$ | 30967 | 1262 | 13.8 | 3.2 | 6.7 | 23.7 |
| Stairs | 14 | $3072 \times 2304$ | 16517 | 265 | 2.7 | 0.8 | 1.1 | 4.6 |
| Timberf. House | 240 | $1280 \times 960$ | 29927 | 2113 | 4.2 | 2.5 | 0.8 | 7.5 |

Table 6.1: Performance evaluation of line-based scene reconstruction. All times are given in minutes.

suming than traditional appearance-based line-matching approaches. Nevertheless, we manage to generate accurate results for the general case in reasonable time. Table 6.1 shows a performance evaluation for the four test sequences presented in this thesis. All experiments were performed on a desktop PC equipped with an Intel Core2 $4 \times 2.66$ GHz processor. Note that for our own sequences (Pylon, House, and Stairs) the image resolution was significantly larger than for the Timber-frame house, which explains the difference in speed.

### 6.1.3 Dense Multi-View Scene Reconstruction

In the following, we evaluate our combination of PMVS [65] point cloud densification and meshing based on the method of Labatut et al. [124]. Densification using planesweep stereo is evaluated in the next section for digital surface models, and surface extraction based on volumetric variational methods is shown for dense reconstruction on-the-fly in Section 6.2.3.

**Evaluation Strategy.** We evaluate the resulting dense geometry using the benchmark dataset *fountain-P11* of Strecha et al. [196]. Following the standard evaluation strategy, we first create depth maps from our result at the groundtruth camera positions, and then compare them to the depth maps that were generated from the ground truth mesh. We further analyze the spectrum of the error in 10 logarithmic bins, starting at deviations smaller than 5 mm up to 2.56 m. Additionally, a bin for *infinite* errors (INF) resulting from missing depth values is added.

**Quantitative Results.** Figure 6.8 shows the quantitative comparison of different meshing approaches. It can be seen that densification is clearly beneficial, as meshing on the sparse point cloud requires considerable interpolation to take place. Additionally, it can be seen that our meshing approach performs much better than Poisson reconstruction [107]. This is due to the fact that visibility information is incorporated. Our results are very accurate, with more than 50% of the mesh within 1 cm of the groundtruth. Note that due to the sparsity of the raw SfM result, our space carving-based approach results in more undefined values (INF) than Poisson meshing which might create triangles without any evidence. However, the higher accuracy comes at a price: On a standard desktop PC, space carving-based meshing takes about 10 minutes, which is about 30 times more than Poisson surface extraction.

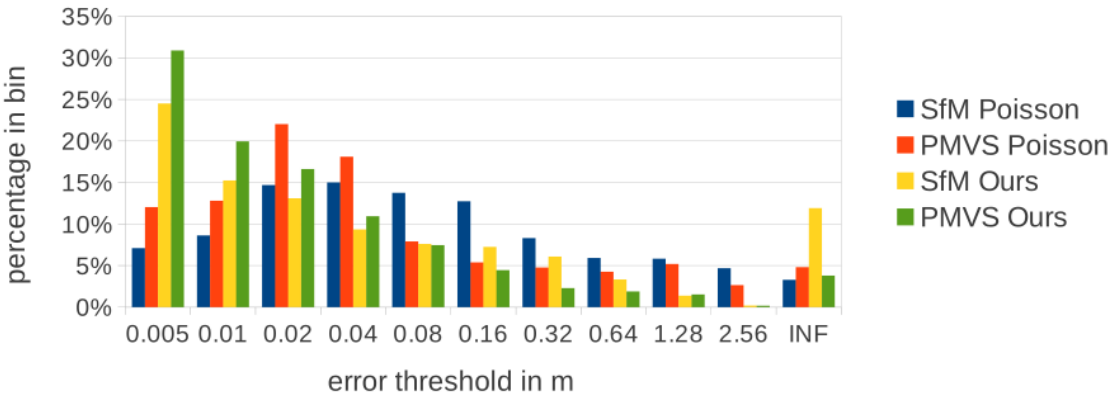
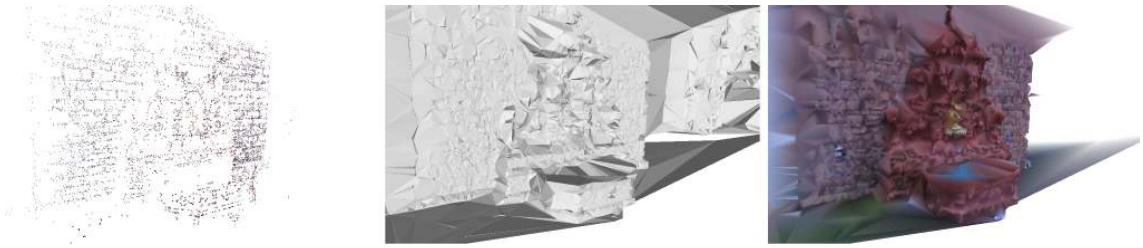
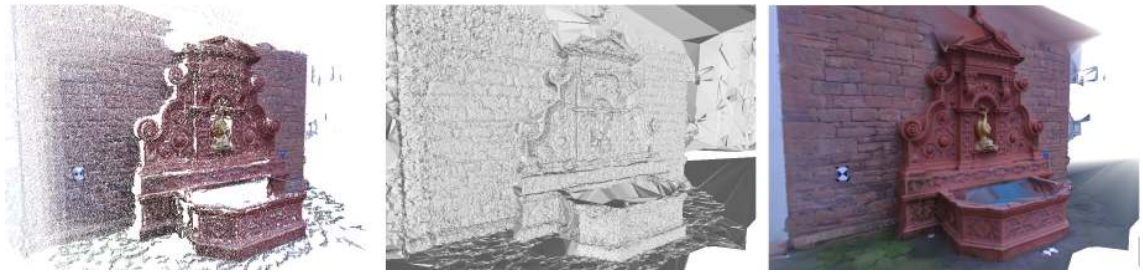



Figure 6.8: Quantitative results of the meshing approach on the *fountain-P11* scene (11 images) of Strecha et al. [196]. We compare the raw and densified pointclouds with Poisson meshing [107], as well as the same pointclouds with our meshing based on Labatut et al. [124] to groundtruth. In general, our results are much closer to groundtruth, with more than 50% of the mesh within 1 cm for the densified and meshed 3D reconstruction. Note that due to the sparsity of the raw SfM result, our space carving-based approach results in more undefined values (INF) than Poisson meshing.

**Qualitative Results.** While the Delaunay triangulation can even extract a surface from such a sparse point cloud, a densified set of points is clearly beneficial. Figure 6.9 compares the meshed and colored *fountain-P11* scene for sparse SfM and densified PMVS input. Our dense multi-view scene reconstruction pipeline can be applied to various indoor and outdoor scenes, and it delivers state-of-the-art results for many applications.



(a)



(b)

Figure 6.9: Qualitative results of the meshing approach on the *fountain-P11* scene (11 images) [196]. Meshing based on (a) the SfM output (7123 points) and (b) the PMVS [65] output (375 409 points). From left to right: Point cloud, uncolored mesh, colored mesh. Only vertex colors are applied, no texturing is performed.

### 6.1.4 Digital Surface Model Reconstruction

We compare our 2.5D range image fusion approach qualitatively on a large-scale aerial dataset captured with a Microsoft UltraCamX with a resolution of $14.430 \times 9.420$ pixels (GSD 8 cm per pixel). We also perform a quantitative evaluation on the ISPRS Vaihingen test dataset [31] with 20 infrared-red-green (IR-R-G) images comprising a resolution of $7.680 \times 13.824$ pixels (GSD 8 cm per pixel). For this dataset, a ground-truth DSM acquired by airborne laser scanning with 25 cm resolution has been provided.

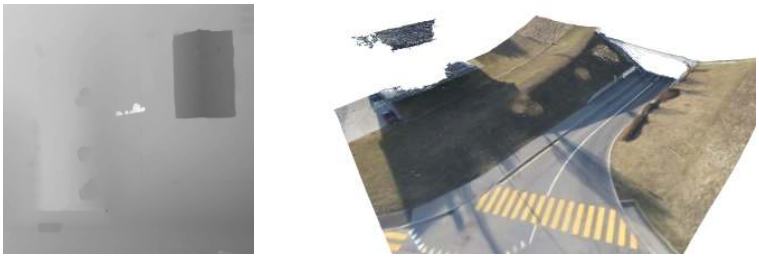
**Comparison to Global Fusion Approaches.** The results for the aerial UltraCamX dataset in form of a textured DSM and a typical height map representation are shown in

Figure 6.10. Note that the DSM shows sharp edges and no visible outliers. In Figure 6.11 we visualize the trade-off between putting large effort in generating clean, outlier-free range maps using global optimization methods [99], and fast depth estimation using local methods. Once the input data is smoothed it may be fused using a simple method like median fusion, but with the drawback of not being able to recover fine detail. In contrast, our method can deal with a substantial amount of outliers in the range data as long as one distinct dense point cluster can be estimated, and it preserves details. Moreover, it is scalable and much faster: The final DSM generated from the UltraCamX dataset shows an area of approximately $3.46\,km^2$ and has a resolution of $20.699 \times 26.096$ pixels with 8 cm GSD. The GPU-based calculation of all 21 depth maps with our local method required 9 hours, the fusion took 15 hours using four CPU cores. In comparison, the global fusion method requires more than 12 days.

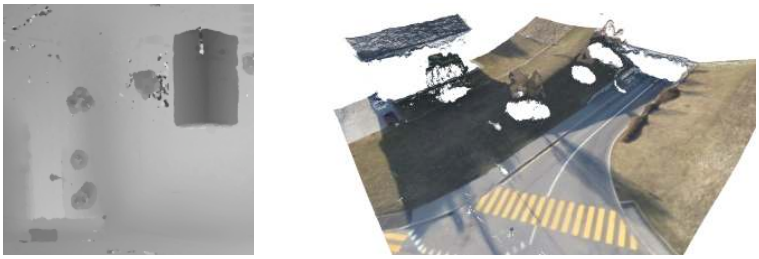
(a) (b)

Figure 6.10: DSM fusion result for a large-format aerial imagery dataset. (a) 3D visualization of the resulting DSM as a textured triangular mesh. (b) Sample part of the DSM represented as a height map.

**Comparison to Groundtruth.** Finally, for a quantitative evaluation we compared the results of our probability-based clustering fusion approach to a median fusion of the range image depth hypotheses. The range images were computed using cost volume filtering [169] before performing winner-takes-all. The chosen evaluation strategy is similar to [196]. Although, we achieve only 5% better results than median fusion, a direct visual comparison between median fusion and our proposed method clearly shows a significant qualitative difference. A small fraction of outliers which do not have a big impact in the quantitative analysis can have a drastic impact on the qualitative appearance, as is shown in Figure 6.12. For automatic alignment of 3D reconstructions in a global coordinate system, as well as high-level path planning and view planning, a scalable method which creates outlier-free DSMs is most important, thus our approach is perfectly suitable.

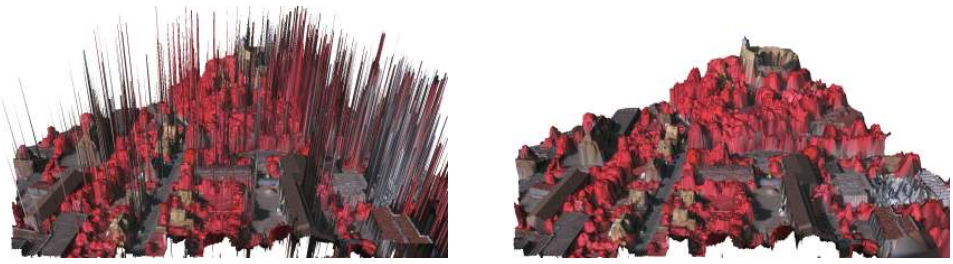


(a) (b)



(c) (d)

Figure 6.11: Comparison between different methods for depth estimation and fusion on the MAV datasets of [159]. (a) Depth map calculated using the method of [99] with global optimization, which produces nearly outlier free but over-smoothed depth maps. (b) Details like the trees and street lights are lost in the fused DSM using simple and fast median fusion. (c) Depth map estimated using our approach with fast cost volume filtering [169], which contains a few outliers but preserves details. (d) Our probabilistic range image fusion method [175] is able to handle large amounts of outliers and still produces visually competitive DSM quality.



(a) (b)

Figure 6.12: Range image integration results on the ISPRS Vaihingen test dataset [31]. (a) Fused DSM computed by median fusion, which shows considerably non-robust results. Single isolated outliers are still present in the results. (b) Our probabilistic fusion approach [175] effectively discards outliers and produces visually clean results.

## 6.2 Visual Localization and Mapping

Building upon the different forms of geometric prior knowledge, we evaluate our multi-scale visual localization and mapping framework as presented in Chapter 4. We demonstrate the effectiveness of alignment in a global coordinate system and the suitability of our approach for long-term localization. Furthermore, we evaluate the accuracy of visual localization compared to differential GPS, and we present results for dense reconstruction in real-time on an MAV.

### 6.2.1 Global Alignment of Geometric Priors

In order to demonstrate the accuracy of our alignment and fusion approach, we perform experiments on two different scenarios, namely the alignment to an overhead DSM and an iterative alignment to a single model.

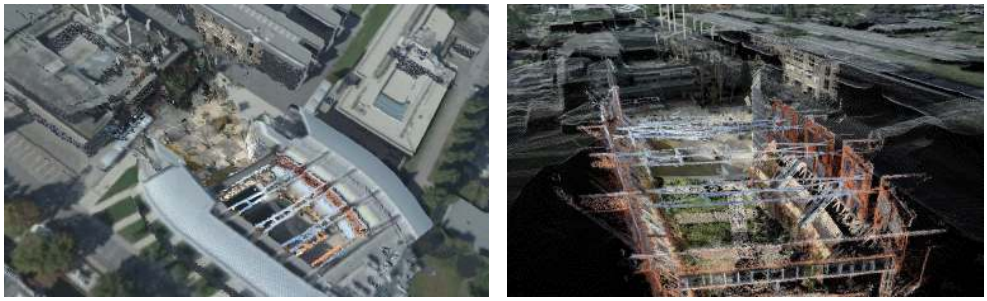
We reduce the search space for precise alignment according to the expected uncertainty of the rough GPS alignment, with a translational uncertainty $\Delta T = \pm 5m$, roll $\Delta\phi = \pm 10°$, pitch $\Delta\theta = \pm 10°$, yaw $\Delta\psi = \pm 20°$, and scale $s = 1.0 \pm 0.2$. A brute force approach for checking all combinations is not feasible due to the huge amount of required templates. Therefore, we exploit a coarse–to–fine scheme that iteratively increases the GSD of the DSM for every pyramid level $G_{gsd}$. We weight the penalizing term with $\lambda = 0.1$.

**DSM Alignment Results.** First, we apply our alignment pipeline to a geo-referenced digital surface model with a resolution of $4096 \times 4096$ pixels. We demonstrate the alignment of two point clouds acquired in winter using our *Falcon* MAV and in summer from the ground, respectively. We additionally employed a consumer–grade GPS which allows rough geo–referencing of the reconstructions. The distance between the object and the camera position varies between 20 m and 60 m which results in a ground sampling distance (GSD) of 8 mm to 24 mm per pixel; however, as our reconstructions are only semi-dense the resulting point clouds are often sparser if texture is missing.

An evaluation by visual inspection shows very good results: For instance, the steel wires in Figure 6.13(a) are well aligned to the steel wires visible in the orthophoto. All vertical walls fit exactly to the gradients defined in the DSM. Figure 6.13(b) shows an oblique view from the top of the first model to the second model.

A simple, small SfM model can be seen in Figure 6.14(a). Figure 6.14(b) shows the rough GPS alignment, whereas Figure 6.14(c) depicts the refined alignment. The model is well aligned even though just one corner of a building has been reconstructed.

**Iterative Alignment Results.** The second dataset shows a large office building which is reconstructed from 400 images taken by the consumer camera on our *Falcon* MAV. Due to limited power supply, images were acquired in two different flights resulting in two partial reconstructions with 2 million and 1.3 million 3D points, respectively.



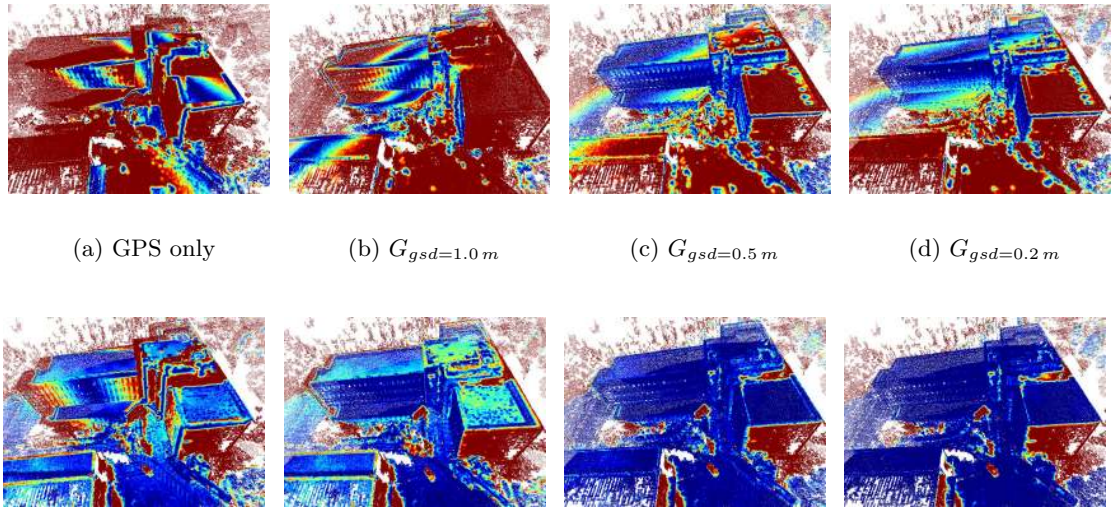
(a) (b)

Figure 6.13: Qualitative evaluation of the DSM alignment. (a) Visual inspection shows that the steel wires of the SfM model are perfectly aligned to those visible in the texture. (b) An oblique view from the top of the first model (atrium) to the second model (opposite buildings). The resolution of the aligned models is considerably better than the resolution of the orthophoto.

(a) 3D Model (b) GPS Alignment (c) Correlation Alignment

Figure 6.14: Evaluation of the DSM alignment with correlation. (a) A partial, densified SfM model (just one corner). (b) Rough alignment using GPS; note the shift and rotation. The model has been manually shifted on the vertical axis for visualization. (c) Alignment with correlation is able to get rid of the remaining error. The shift on the vertical axis has been automatically corrected here.

In this experiment we demonstrate that our approach is also suitable if one of the two models is considered as a small-scale digital surface model by projecting it to the ground. We compare the accuracy of our alignment algorithm to a standard 3D ICP [238] registration method and demonstrate that our approach is able to successfully fuse such reconstructions while ICP is not. The discrete correlation approach successfully avoids getting stuck in local minima, but continuous optimization can nearly always improve the final result. We thus further improve the accuracy by applying 3D ICP on top of



(a) GPS only (b) $G_{gsd=1.0\,m}$ (c) $G_{gsd=0.5\,m}$ (d) $G_{gsd=0.2\,m}$

(e) GPS & ICP (f) $G_{gsd=1.0m}$ & ICP (g) $G_{gsd=0.5m}$ & ICP (h) $G_{gsd=0.2m}$ & ICP

Figure 6.15: Evaluation of the iterative alignment accuracy. Red points have a Hausdorff distance larger than 10 cm. Blue points are closer than 1 cm to the second model. (a)-(d) Accuracy after alignment using different ground sampling distances for $G_{gsd}$ without ICP. (e)-(h) Remaining error after solving Equation 4.3 with subsequent ICP. Note that points present in only one part of the reconstruction by definition have a Hausdorff distance of more than 10 cm and are thus colored in red.

our result. Given the already very good alignment, a sparse set of $N_{icp} = 1000$ points is selected and ICP typically converges within seconds.

First, we evaluate pure GPS-based transformation in a common coordinate system. The two parts of the model form some kind of groundtruth, so we can measure the alignment error by evaluating the Hausdorff distance from the smaller to the larger part [96]. Figure 6.15(a) shows the resulting pseudo color visualization, where red points have an error larger than 10 cm and blue ones have an error smaller than 1 cm. After the previous rough alignment, we perform 3D ICP to reduce the error. However, ICP gets stuck in a local minimum and therefore the alignment is far away from the desired registration (Figure 6.15(e)). This experiment demonstrates that, due to the small convergence radius of ICP, a consumer–grade GPS alignment is not sufficient to obtain an accurate fusion using a standard 3D ICP algorithm. On a closer look, we also observe that the scale estimated by the rough GPS alignment differs considerably between the two parts. Therefore, a proper fusion method has to estimate the scaling between the parts.

In the second experiment, we apply our proposed algorithm to the transformed second part. As shown in Figure 6.15(b)-6.15(d), the alignment error reduces with increasing

resolution of the ground template $G_{gsd}$. On a scale level of $G_{gsd=0.5\,m}$, the fusion method already allows the subsequent ICP to converge to the global optimum. For most points, the remaining alignment error is smaller than 1 cm. Points that are present in only one part of the reconstruction have a Hausdorff distance of more than 10 cm which is obvious since they do not have a corresponding counterpart. Since we correctly estimate the scale difference of $\Delta s = 0.04$, the result does not show errors in scaling. This observation is confirmed by Figure 6.15(h) which shows a constant error over the entire surface. Subsequently applying ICP has shown to converge within seconds and mainly corrects for rotational errors introduced by the discrete search space of our approach. The entire fusion process requires less than 5 minutes on any of our evaluation datasets for finding the correct transformation with an accuracy in the range of $\pm 1\,cm$. This corresponds to the point density of the model.

In summary, our approach allows to align several Structure-from-Motion point clouds in a common, global coordinate system despite considerably different viewing angles and ground sampling distances. Accurate alignment is an important part of our localization framework.

### 6.2.2 Point-based Localization using Geometric Priors

In the following paragraphs we evaluate our localization approach, and accordingly the visual landmark creation and alignment compared to visual and GPS-based groundtruth. The evaluation site, the atrium of a modern building, has already been used to visualize the individual steps of our approach. This environment is very challenging, as there is little texture on the facades and repetitive structures such as windows and vegetation prevail.

**Evaluation Dataset and Hardware.** For the creation of the initial visual landmark we captured 157 images from eye-level above ground using a Canon EOS 5D. This SLR camera has a resolution of $5616 \times 3744$ pixels with a fixed 20 mm lens.

Aerial imagery is obtained using the *Pelican* MAV, and the images are streamed to the ground at 4 fps. Our ground station is equipped with a powerful graphics card running NVIDIA CUDA, which allows highly parallel data processing for feature extraction and matching. Additional sensor data from the inertial measurement unit (IMU), compass, and global positioning system (GPS) are automatically preprocessed and fused by the MAV. While hardware synchronization with the camera trigger would be desirable, we currently exploit the high update rates of the sensors and collect IMU and GPS messages between triggering and receiving an image. Attitude and position are then estimated using robust statistics.

For the iterative update of the model we acquired two image sequences ($1280 \times 1024$ pixels at 4 fps) of 260 images (*flight1*) and 230 images (*flight2*) by exploring the atrium in manual flight mode. Each flight started at the ground and covered a volume of about

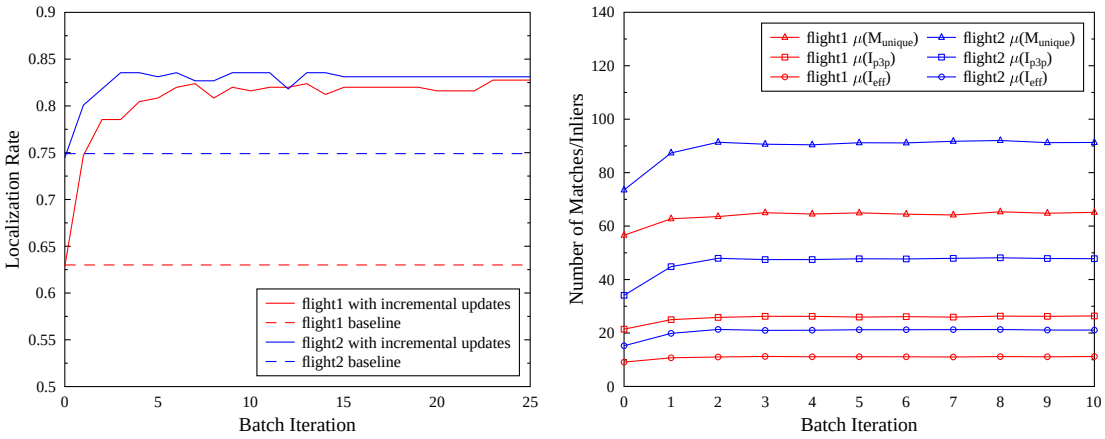
$10 \times 10 \times 15\,m^3$. Additionally, we have acquired a video stream showing the flight from an observer's point of view. This stream has been recorded with a resolution of $1920 \times 1028$ pixels at 25 fps, and is used as groundtruth for evaluating our localization performance qualitatively. Finally, for a quantitative validation of the position accuracy we captured images at fixed points which were globally localized using differential GPS (D-GPS).

The localization process including the online update runs at 4 fps. The runtime of a batch update depends on the length of the flight and the number of modified virtual cameras, and results to approx. 5 min per iteration on our datasets. For the following experiments, a minimal virtual camera coverage $N_{coverage} = 3$ is achieved by setting the grid to $G = \{7, 4, 4\}$ and the angular step to $\gamma = 30°$. This results in $112 \times 62 = 6944$ virtual cameras, which are reduced to a set of of 1903 virtual views by thresholding the minimal number of features to $|F|_{min} = 200$. For localization, we estimate the pose of at most $k = 3$ top–scoring views in a neighborhood of $t_{max} = 1.0\,\text{m}$ and $R_{max} = 45°$ with respect to the previously established pose, and accept the pose if it contains at least $th_{eff} = 3$ effective inliers. Incremental updates are evaluated with $th_{attitude} = 10°$, $th_{position} = 20\,m$, and $th_{update} = 1°$.

**Comparison of Adaptive and Non-Adaptive Localization.** Our first experiment evaluates the localization performance as the ratio of correctly localized frames based on additional sensor data, as previously described in Section 4.2.3. The ratio of frames which are localized correctly is named $R_{valid}$. $R_{invalid}$ describes the ratio of frames which exceed $th_{attitude}$ or $th_{position}$ and therefore fail validation.

We achieve a baseline of 63% for *flight1* and 75% for *flight2* using the non-adaptive approach [221]. Our evaluation shows that issues with high–altitude views occur starting at a height of about $6\,m$, which corresponds to the theoretical result. In contrast, Figure 6.16(a) shows that adaptive localization can considerably improve these results. After applying 25 incremental updates, meaning that the inliers of all frames were taken into account at every iteration except for the first, we achieve a localization rate of 83% for both *flight1* and *flight2*. This corresponds to an improvement of up to 30% in comparison to non-adaptive localization. Even using just the online update during the flight, and neglecting the batch updates afterwards, localization rates of 71% and 80% are reached. The online update heavily depends on the flight trajectory, as early frames do not benefit from later frames and thus many camera poses might be initially unknown.

Not only the localization rate but also the localization quality in terms of matches and pose estimation inliers is improved by using the adaptive localization. Figure 6.16(b) thus compares the mean values of unique matches ($M_{unique}$), inliers ($I_{p3p}$) and effective inliers ($I_{eff}$). These values considerably increase during the first update and then converge to a maximum, for both test datasets. Especially $I_{eff}$ is important to reliably estimate the MAVs pose, as it additionally incorporates the distribution of RANSAC inliers in an image. $\mu(I_{eff})$ increases from 9 to 12 points for *flight1*, and from 15 to 21 points for *flight2*.




(a) (b)

Figure 6.16: Localization performance for non-adaptive and adaptive methods. (a) Dashed lines show the baseline using the non-adaptive algorithm [221] whereas solid lines depict correctly localized frames ($R_{valid}$) based on IMU validation. Our adaptive method improves the localization performance $R_{valid}$ from 62.8% to 83.1% for *flight1*, and 74.5% to 83.5% for *flight2*. (b) For both testsets, all quality measures improve as well when applying incremental feature updates.

The update of the model is a critical point because miss-aligned frames degrade the model quality. We expect that all localizations passing the additional validation step are correct. However, this heavily depends on the parameters $th_{attitude}$ and $th_{position}$, so the residual errors have to be checked. The selection of the attitude threshold $th_{attitude}$ was done by evaluating the valid localizations, the according drop rates by validation, and the resulting mean attitude errors $\mu(e_{attitude})$. Tab. 6.2 shows that for increasing $th_{attitude}$ the ratio $R_{valid}$ and $\mu(e_{attitude})$ increase, whereas $R_{invalid}$ decreases. The selection of the threshold, in our case $th_{attitude} = 10°$, is a trade-off between detection rates, outlier removal, and expected IMU accuracy, and it corresponds to earlier work on comparing visual and inertial data [125]. In contrast, GPS is in our experience very noisy. We use the real-time accuracy delivered by the GPS receiver, which results in an average position threshold of $\mu(th_{position}) = 16.2m$. It thus only acts as a sanity check for localization in models with repetitive structures.

Figure 6.17 shows the resulting camera positions after applying our approach. Initially localized cameras are depicted in gray, positions found after several batch updates are marked in red. A common source of error are frames distorted by motion blur, because only little features can be detected. Nevertheless, our proposed approach is able to handle such local failures since it automatically recovers with the next successful pose estimate.

Table 6.2: Localization Results for Different Attitude Errors after 25 Incremental Update Iterations.

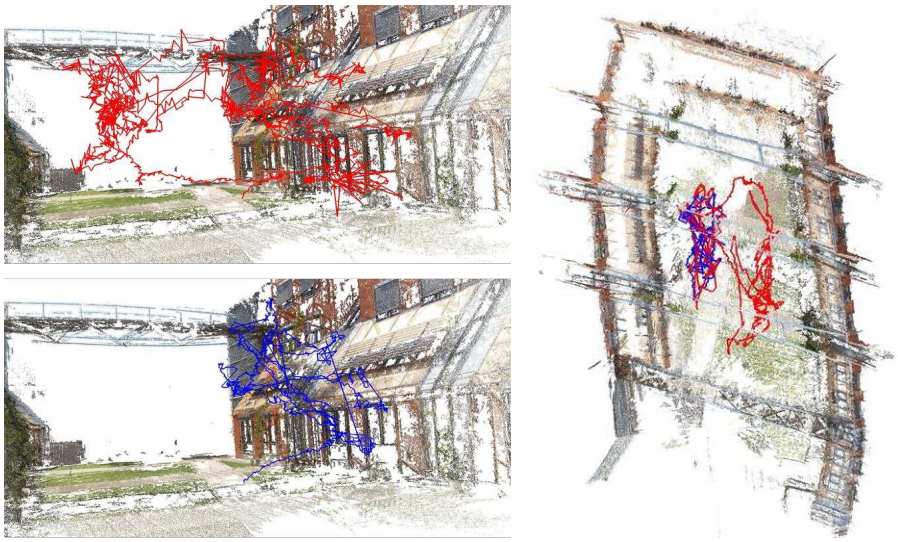
| Flight | $th_{attitude}$ | $R_{valid}$ | $R_{invalid}$ | $\mu(e_{attitude})$ |
|---|---|---|---|---|
| 1 | 5° | 43.7% | 41.4% | $3.61° \pm 1.03°$ |
| 1 | 10° | 82.8% | 2.3% | $5.04° \pm 1.85°$ |
| 1 | 15° | 84.3% | 0.8% | $5.05° \pm 1.90°$ |
| 2 | 5° | 53.2% | 42.0% | $3.52° \pm 0.96°$ |
| 2 | 10° | 83.1% | 12.1% | $4.77° \pm 2.00°$ |
| 2 | 15° | 86.5% | 8.7% | $5.12° \pm 2.55°$ |

Figure 6.17: Visual localization result for an entire MAV flight. In a non-adaptive setting, we can localize 62.8% of all frames of this dataset (gray camera positions). By using our technique of incremental feature updates, we are able to boost the localization performance to 83.1%, especially for high–altitude views (red camera positions).

**Comparison to Visual SLAM.** We compare our approach to localization using PTAM [113], which has recently been proposed for MAV navigation [2, 21, 218, 224]. While this is an unfair comparison as PTAM has to create the map on–the–fly while we use prior knowledge, our intention is to demonstrate the significance of using prior information. Our experiments showed that the camera positions obtained by running the original PTAM code vary considerably according to the stereo initialization. Therefore, we used the recorded stream of images (with radial distortion corrected) and selected the initialization which showed best results for comparison. As the global coordinate system and the scale are not defined and differ in every execution, we aligned the camera centers of PTAM to those retrieved by our proposed algorithm in a least–squares sense.

Figure 6.18 depicts the reconstructed flight path from take–off to landing. While PTAM can compete with our approach locally, i.e. in the environment it was initialized,

it only returns a valid pose for 36% of all frames. In a direct comparison to our approach it is clearly visible that PTAM misses large parts of the flight. As soon as the track is lost, the relative pose cannot be established anymore and the localization fails. Another problem is the repetitiveness of the scene, which causes misleading model updates and inhibits successful re-localization within a larger space.



(a) (b)

Figure 6.18: Comparison of our approach to visual localization using SLAM. (a) Top: Flight path of our visual landmark–based approach with 83% of all frames correctly localized. Bottom: In comparison, a state–of–the–art visual SLAM approach (PTAM [113]) is not suitable for such a large–scale outdoor environment, and achieves a localization rate of only 36%. (b) The overlay of both paths shows that PTAM has lost track of its initial map at some point during the flight and is not able to recover.

**Comparison to Visual Groundtruth.** We use the video stream showing the quad–rotor's flight from a ground observer's viewpoint as a visual groundtruth. It shares the set of virtual cameras, so we can localize the video within our scene using the same localization method as above. Under the premise of synchronized video streams and a correct current pose estimate from the quad–rotor's flight and ground observing video, the backprojection of the MAV camera center must coincide with the quad–rotor position in the observer's view.

Figure 6.19(a) shows the 3D box of the tracked quad–rotor and the respective ground video frame. We use the widespread PASCAL criterion [53] and consider the backprojected

quad–rotor bounding box $\mathcal{B}_Q$ as successfully registered if the criterion,

$$\frac{\mathcal{B}_Q \bigcap \mathcal{B}_G}{\mathcal{B}_Q \bigcup \mathcal{B}_G} > 0.5 \tag{6.1}$$

is satisfied. $\mathcal{B}_G$ denotes the bounding box of the quad–rotor in the groundtruth view. By visual inspection, we conclude that the localization approach as presented in this paper satisfies the overlap criterion in all frames. This is not very surprising, as the automatic validation process using the IMU removes erroneous localizations. In contrast, visual SLAM without validation (PTAM) is only correct in 22% of all frames. Figure 6.19(b) gives an example of the typical offset.

**Comparison to Differential GPS.** For the evaluation in a global coordinate system we acquired images as well as GPS and IMU data at a fixed reference point. Further, at the same point we recorded ground–truth position data using a Novatel OEMV-2 L1/L2 Real-Time Kinematic (RTK) receiver with an estimated precision of 0.02 m at the time of image acquisition. Figure 6.20 depicts the results in a nadir and oblique view. The reference point has been chosen on the bridge of the atrium scene because even D–GPS was not able to measure accurate positions on the ground. The mean offset in the $xy$–plane for 10 measured GPS points is $9.76 \pm 1.31$ m, whereas the mean offset of the 10 localized cameras is $0.31 \pm 0.04$ m. In comparison to our previous work [222] we were able to further reduce the constant offset from 0.52 m to 0.31 m by refining the calibration of the camera. The residual error depends on accurate alignment in a global coordinate system, which in turn depends on the ground sampling distance of the given digital surface model. However, we want to highlight the very good standard deviation of 4 cm in a global coordinate system given the distance to the scene of approx. 20 m; this clearly shows that visual localization is capable of outperforming GPS in urban environments.

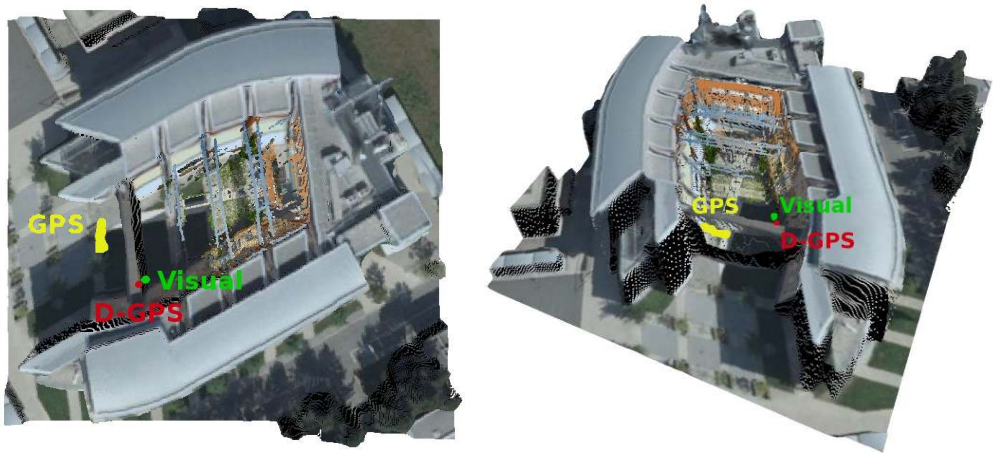
We made another interesting observation in the course of producing these ground–truth results: Visual localization did not work for any acquired image on the atrium's bridge in a height of approx. 13 m over ground when using the initial visual landmark. Only after updating the landmark using our adaptive localization algorithm were we able to localize the MAV, which again shows that our proposed incremental update approach is beneficial.

**Long-Term Localization.** Long-term localization represents the localization of a flight using a model that has been updated by previous flights. In this experiment we evaluate the improvements in localization rate and quality from the initial model to an automatically adapted model. Tab. 6.3 shows that using a model updated by previous flights improves the detection rate by up to 17%, and the measurements for the localization quality $\mu(M_{unique})$, $\mu(I_{p3p})$, and $\mu(I_{eff})$ are also improved.

In other words, flying several times in the same area and using the updated model

(a) (b)

Figure 6.19: Visual comparison of the localization results based on a registered groundtruth video. (a) Oblique view of registered groundtruth showing the current camera frame and MAV positions (path and bounding box) estimated by our localization approach (red) and the aligned PTAM result (blue). (b) Back–projected bounding box from the observer's point of view.




(a) (b)

Figure 6.20: Comparison of visual and GPS localization accuracies. Camera poses based on GPS, vision, and differential GPS as reference point are visualized in (a) a nadir view and (b) an oblique view. Visual localization has been performed on a visual landmark trained by *flight2* to overcome the height limitation, and performs considerably better than GPS.

Table 6.3: Long-Term localization results. Localization based on the initial visual landmark (flight 1@0, flight 2@0) is compared to localization based on adapted visual landmarks. No further incremental updates were performed on the result.

| Flight | $R_{valid}$ | $R_{invalid}$ | $\mu(M_u)$ | $\mu(I_{p3p})$ | $\mu(I_{eff})$ |
|---|---|---|---|---|---|
| 1@0 | 62.8% | 3.1% | 56.6 | 21.4 | 9.1 |
| 1@2 | 73.6% | 2.7% | 60.4 | 23.7 | 10.2 |
| 2@0 | 74.5% | 11.3% | 73.5 | 34.1 | 15.2 |
| 2@1 | 79.7% | 11.3% | 80.4 | 38.9 | 17.4 |

of previous flights improves the localizations of the current flight and allows incremental improvement of the visual landmark. In contrast to visual SLAM, where the environment is expected to be unknown and the map is re-created for every flight, our approach is able to incorporate and propagate prior knowledge in a certain area to improve localization results. Moreover, we can even integrate the maps created by keyframe-based visual SLAM as discussed in the following.

### 6.2.3 Distributed Online Localization and Mapping

The lowest level of our localization and mapping framework is again a SLAM component. However, a deliberate implementation as a distributed system allows to run both localization and mapping without algorithmic simplifications. On the contrary, our system exploits several state-of-the-art algorithms for tracking, mapping, and dense reconstruction [113, 114, 234, 235] despite the low computational resources on the MAV.

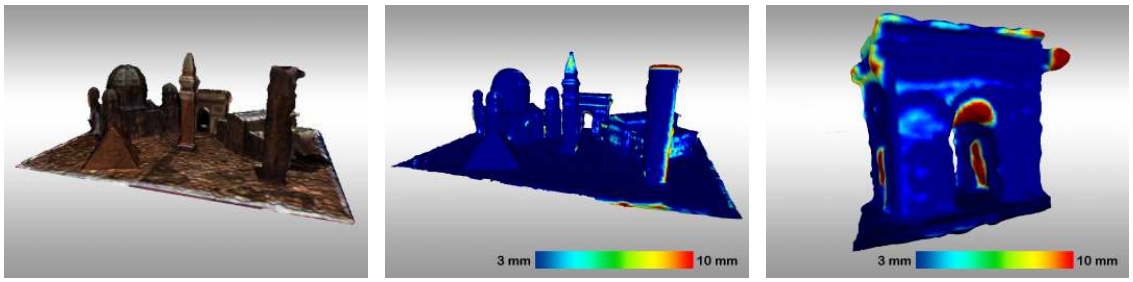
**Experimental Setup.** In all of our experiments, we run the tracking thread onboard our *Pelican* MAV. Our mapping component runs on a powerful server with a 2.54 GHz Quad-Core CPU and an NVIDIA GTX480 GPU which is used for highly parallelized dense reconstruction. The user interface is implemented on an Android-based NVIDIA Tegra3 tablet, and all three parts are connected by ROS via standard WiFi.

A typical live reconstruction process starts with the creation of an initial map, which happens automatically if the user has placed an artificial ARToolkitPlus marker [215] into the scene, or if the system is connected to our virtual view-based global localization. Then, the desired volume of interest for dense reconstruction, relative to the center and orientation of the marker, has to be defined by the user. For outdoor experiments, this process can be done on the ground or while the aerial vehicle is airborne. Once the system is initialized, reconstruction is fully automated and starts as soon as a minimum of three keyframes is available.

In the following paragraphs we experimentally evaluate the effect of different keyframe resolutions on the reconstruction. All other parameters can be interactively tuned to

obtain good reconstructions in different environments. However, for the evaluation we used the following fixed set of parameters: For initialization $b_{init} = 0.3m$ and $f_{init} = 20$, for keyframe selection $b_{map} = 0.1m$ (indoors) and $b_{map} = 0.4m$ (outdoors), $\alpha_{map} = 20°$, and $t_{map} = 3$s, for map updates $\epsilon_{2d} = \sqrt{2}$px, $\epsilon_{3d} = 0.001m$ (indoors), and $\epsilon_{3d} = 0.01m$ (outdoors), for dense mapping $N = 4$, $\lambda = 0.4$, $\lambda_d = 0.5$, and $c_{min} = 0.65$.

**Ground-truth Evaluation.** Typical evaluation datasets providing dense ground–truth are not designed for SLAM, and thus tracking typically fails. We recorded our own evaluation data based on standardized geometry and texture using the *City of Sights* paper model [71], which allows us to demonstrate the quality of our reconstructions by comparing to ground–truth. Digital blueprints and a virtual model of the scene are provided online[2]. As proposed in the original paper, we use the Iterative Closest Points method [238] to align the reconstructed volume to the ground–truth mesh. We finally measure the Hausdorff distance from reconstructed to ground–truth points and visualize the residual in pseudo colors. Gruber et al. [71] state that typical paper models have a mean deviation of 3 mm compared to ground–truth, thus we denote our reconstruction as correct within this limit. Errors of more than 10 mm occur only in regions which are either seen by a small number of views or which are hard to triangulate due to missing texture. In the *City of Sights* model, this applies to the top of the Irish Round Tower and the inner part of the Arc de Triomphe. A standard result for an input resolution of 640 × 512px, as well as a pseudo–colored distance image are shown in Figure 6.21.

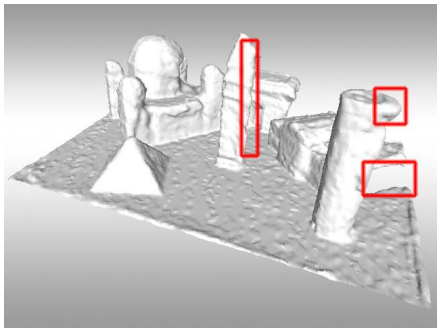



(a) Textured reconstruction (b) Difference to groundtruth (c) Maximum error

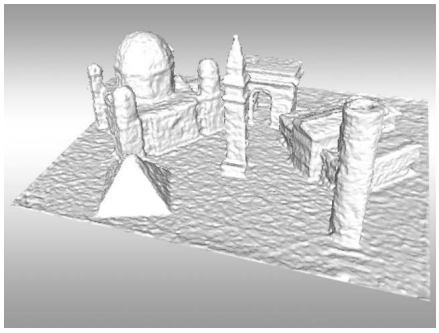
Figure 6.21: Dense reconstruction of the *City of Sights* model using an input resolution of 640 × 512px. The mean Hausdorff distance to ground–truth is 2.4mm, the RMS distance is 3.9mm, and the maximum distance underneath the side gates of the Arc de Triomphe is 42.8mm.

**Resolution and Bandwidth Considerations.** Our system can capture and process the full image resolution of 1280×1024px with a frame rate of $5fps$ on–board the quadrotor

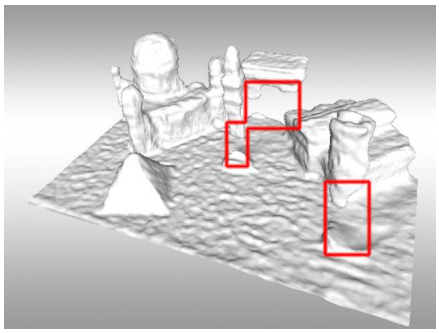
[2] http://cityofsights.icg.tugraz.at



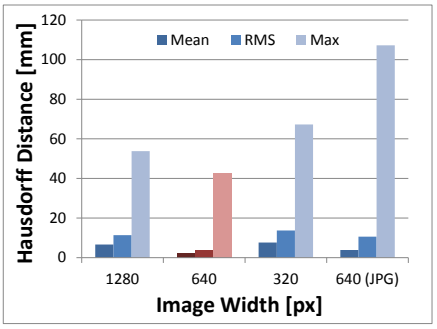
(a) 1280 × 1024px



(b) 640 × 512px



(c) 320 × 256px




(d) Overview

Figure 6.22: Qualitative and quantitative evaluation of the input resolution. (a) to (c) show the reconstructed geometry for different input resolutions with erroneous regions marked in red. The diagram (d) shows mean, RMS, and maximum Hausdorff distance compared to ground–truth. The best result is obtained using an uncompressed keyframe resolution of 640 × 512px.

helicopter. Frame rates increase to $10fps$ when lowering the image resolution to 640 × 512px, and $20fps$ for $320 \times 256$px. We compare the different resolutions and frame rates by recording uncompressed video with full resolution and $20fps$ on a desktop computer. The data is then ported to a laptop, resampled to the correct resolution and frame rate, and used as input to the tracker which sends keyframes over WiFi to the server. The rather complicated setup perfectly simulates our system; however, repeated reconstructions might still differ due to network and CPU workloads. Figure 6.22 shows a qualitative and quantitative comparison of reconstructions with different keyframe resolutions based on the Hausdorff distance. While the full resolution provides the smoothest result as expected, tracking often fails due to the low frame rate and some holes remain. Medium resolution performs best, whereas the low resolution of 320 × 256px cannot connect all extrusions in the model.

Directly streaming the imagery to the server has also been evaluated, and is possible with the same frame rate for compressed medium resolution (80% JPG) and uncompressed low resolution images. While low image resolution has already been shown to be undesirable, compression also affects the reconstruction quality (see Figure 6.22(d)). Additionally, the benefit of having latency-free pose updates on the MAV is omitted.

Finally, when comparing to the non–distributed system we did not see significant deviations in reconstruction quality. A difference is only apparent if the tracker has to quickly update the map several times in a row, which results in a short-term loss of the map because of the network latency. However, this is in general not an issue when operating outdoors because the distance to the scene is larger and less frequent map updates are required.

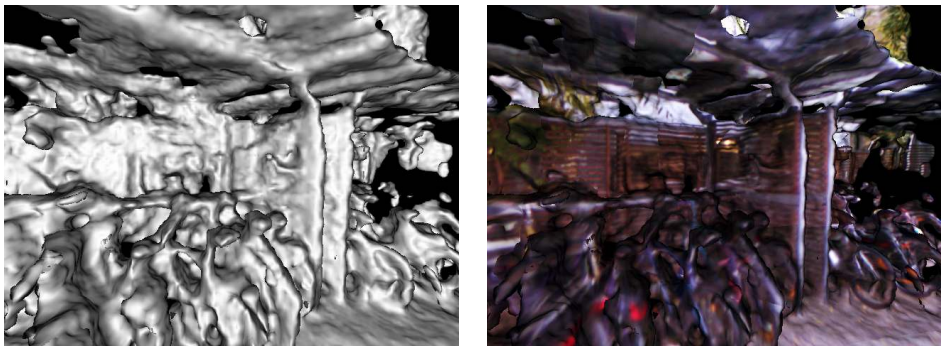
(a) Scene photo (b) Textured reconstruction

Figure 6.23: Office scene (indoor) with a reconstruction rotated to an overhead view. Note that even weakly textured regions are reconstructed. Volume size $320 \times 320 \times 320$ voxels.

**Real–World Evaluation.** Our system has also proven to deliver very good results in real–world scenes. We have successfully reconstructed indoor office scenes by moving the micro aerial vehicle (Figure 6.23) and outdoor scenes by flying it (Figure 6.25). The typical acquisition time is below 10 minutes for all the scenes shown, and dense reconstruction is computed in real-time, with visualization on the tablet. Our reconstruction approach is not tied to a specific topology but can visualize very complex scenes as shown in Figure 6.24. We can also handle different input such as thermal imaging data, where colored texture is ultimately required, and reconstruct the dense 3D scene on–the–fly.

### 6.2.4 Online Localization using Line-based Models

3D point clouds are an important cue for visual navigation, but sometimes it is necessary to localize relative to geometry given as wireframe model instead of exploiting the appearance. In this section we evaluate our model-based multi-scale pose estimation approach which



(a) Untextured reconstruction (b) Textured reconstruction

Figure 6.24: Dense reconstruction of complex geometry. Thanks to our volumetric reconstruction approach, we obtain smooth surfaces and do not depend on topological constraints.

(a) (b) (c) (d)

Figure 6.25: Airborne outdoor results. In (a) to (c), various outdoor scenes are depicted where the top row shows an image of the scene and the bottom row depicts the reconstruction using a volume of $320 \times 320 \times 320$ voxels. (d) shows the reconstruction of a car park using thermal images as input.

builds on PTAM and integrates prior knowledge of the object of interest during the entire tracking procedure. Several experiments with solid as well as wiry objects have been performed. Our implementation runs in real-time with 15-20 fps, while exploiting the computational resources of multicore processors.

**Evaluation using Outside-in Tracking.** We have evaluated our adaptations in terms of tracking accuracy by comparison to groundtruth, acquired by an outside-in tracking system. All evaluation tasks are performed using an Intel Core i5 2,66 GHz processor. While for the translational error the Euclidean distance

$$t_{err} = ||\mathbf{t} - \mathbf{t}_{true}|| \tag{6.2}$$

between the two camera centers is used, an attitudinal error measure based on a quaternion representation has been chosen. A quaternion is defined as

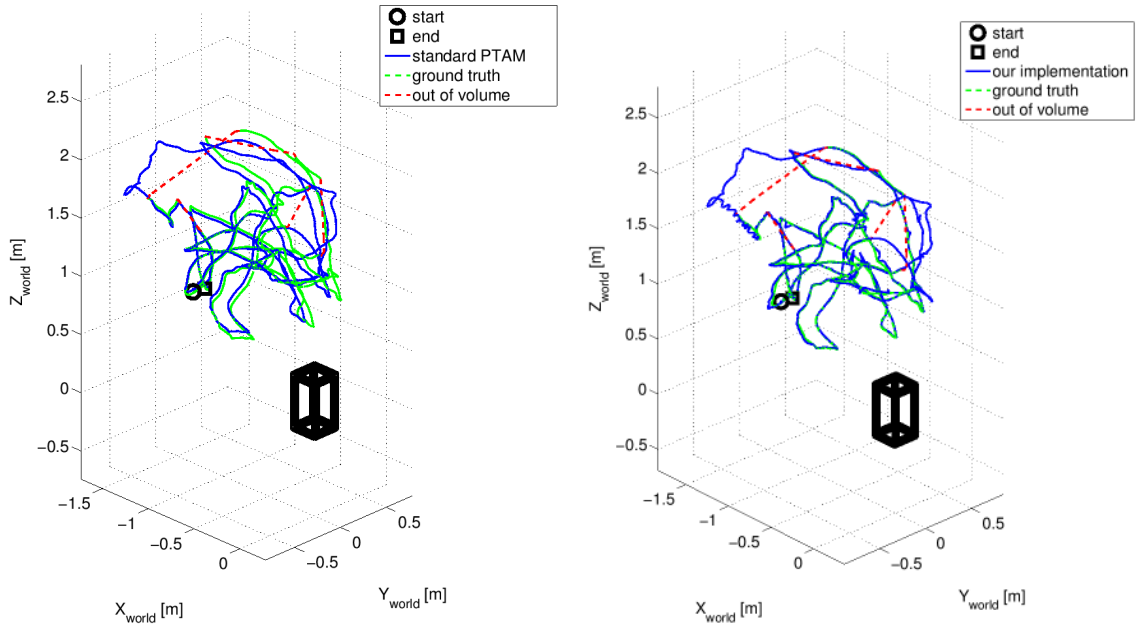
$$\mathbf{q} = \begin{bmatrix} \boldsymbol{\rho}^T & q_w \end{bmatrix}^T \text{ with } \boldsymbol{\rho} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^T = \hat{\mathbf{e}} \sin\left(\frac{\theta}{2}\right) \text{ and } q_w = \cos\left(\frac{\theta}{2}\right), \tag{6.3}$$

where $\hat{\mathbf{e}}$ is the axis of rotation and $\theta$ is the angle of rotation [184]. Thus, the angular difference between two quaternions $\mathbf{q}_{true}$ and $\mathbf{q}$ can be calculated from $q_w$ as

$$\theta_{err} = 2 \arccos\left( \, ||\mathbf{q}_{true}^{-1}\mathbf{q}||_w \, \right) \, , \tag{6.4}$$

where $||\ \cdot\ ||_w$ is an operation that first normalizes the quaternion resulting from the quaternion multiplication, and then extracts $q_w$. The parameters have been fixed to $t = 0.2$, $\sigma_c = 0.7$, $\lambda = 0.8$, $\tau = 0.025$, and an enlargement of the object bounding box by 30% for all experiments is chosen.

**Comparison to Standard PTAM.** Evaluation results for a solid object are presented in Table 6.4, those regarding a wiry object are listed in Table 6.5 and depicted in Figure 6.26. The results show that our algorithm clearly outperforms standard PTAM. While this is also due to model-based refinement during tracking, the major improvement is caused by refinement during initialization. In standard PTAM, relative motion estimation between the first and second keyframe estimated from the piecewise planar environment using [56] is not robust. However, when the model information is incorporated into the initialization process to refine the camera poses for the first and the second keyframe, respectively, then standard PTAM is almost as accurate as our implementation. The accuracy gained when constantly refining the pose during tracking using the model is quantitatively significant, but is only essential when being very close to the object in our indoor experiments. However, outdoors the pose estimate provided by PTAM is noisier and therefore model-based refinement is beneficial. Qualitative results of our implementation for different wiry objects are shown in Figure 6.27 and 6.28.




(a) (b)

Figure 6.26: Model-based tracking trajectories. Comparison of the tracked trajectories (blue) to ground-truth (green) for the wiry object. (a) PTAM has a considerable offset, whereas our implementation (b) aligns well.

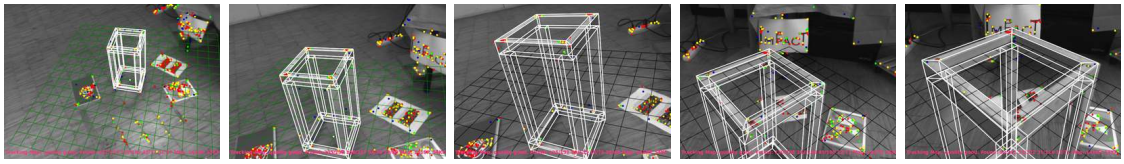


Figure 6.27: Qualitative results of our implementation for the wiry object used during evaluation. The pose is estimated from keypoint-based tracking and then refined using the available information about the object's geometry.

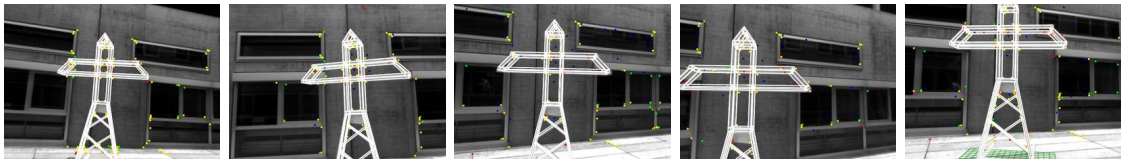


Figure 6.28: Qualitative results of our implementation for an outdoor scene with a wooden model of a power pylon.

Table 6.4: Model-based tracking accuracy for a solid object. The accuracy of the standard PTAM implementation is compared against our implementation with model-based refinement.

| | transl. error [cm] | | | rot. error [deg] | | |
|---|---|---|---|---|---|---|
| | $\mu$ | $\pm\sigma$ | max | $\mu$ | $\pm\sigma$ | max |
| PTAM | 6.1 | 3.4 | 17.7 | 1.2 | 0.4 | 3.7 |
| Ours | 1.7 | 0.8 | 6.4 | 1.0 | 0.4 | 3.2 |

Table 6.5: Model-based tracking accuracy for a wiry object. The accuracy of the standard *PTAM* implementation is compared against our implementation with model-based refinement.

| | transl. error [cm] | | | rot. error [deg] | | |
|---|---|---|---|---|---|---|
| | $\mu$ | $\pm\sigma$ | max | $\mu$ | $\pm\sigma$ | max |
| PTAM | 6.9 | 2.9 | 32.0 | 1.4 | 1.3 | 11.3 |
| Ours | 2.6 | 3.1 | 31.3 | 1.6 | 1.3 | 11.7 |

**Overcoming Keypoint-based Tracking Failures.** The main benefit of our implementation is that the combination of the keypoint-based tracking procedure with model-based tracking keeps working in cases where standard PTAM would lose track. This is especially important when capturing close-up views of parts of the object as it is often necessary during aerial inspection tasks. We have simulated such close-up views using the wiry wooden box, and compared the standard PTAM implementation against our implementation. Loss of tracking could always be prevented for the 2:20 min evaluation sequence (4110 frames) when using model-based tracking for failure recovery. This was not possible using the standard PTAM implementation where tracking is completely lost in 20.9%. Moreover, we could not only keep up tracking during longer periods of time, but even avoid further loss of frames by the use of model-based tracking. The frames where recovery was necessary reduced from 20.9% to 6.6%. Qualitative tracking results can be seen in Figure 6.29.

## 6.3 Path Planning and Control

Prior knowledge about geometrical structures also aids path planning and control, as we have shown in Chapter 5. The following sections show ground-truth evaluations and qualitative results for view planning, visual servoing, and reactive control strategies.

### 6.3.1 Vision for High-Level Path Planning and Control

For large-scale view planning, we show that we are able to determine a set of camera positions that guarantees that our SfM algorithm can compute an overall connected reconstruction with accurate scene estimation. Therefore, we test the performance of our

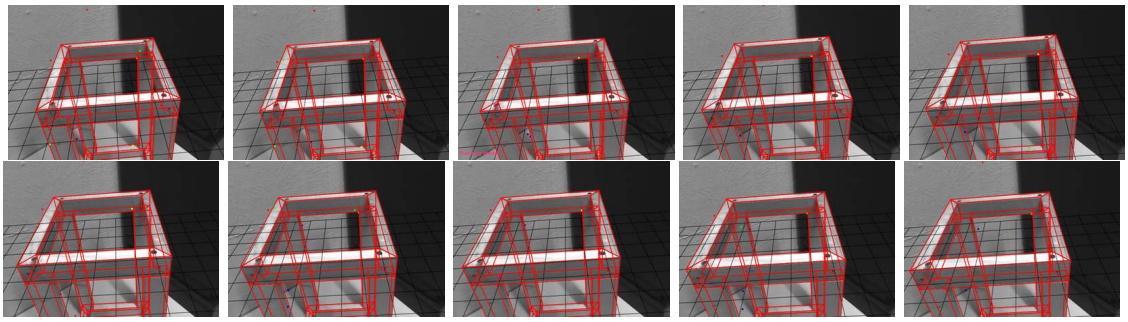


Figure 6.29: Purely model-based tracking for failure avoidance. When not enough features for tracking are available, we automatically switch to a model-based tracking stage in order to avoid tracking failure. Qualitative results are shown here for ten continuous frames.

view planning algorithm described in Section 3.5.2 in simulation and during a real-world outdoor experiment.

In both experiments, we ensure a minimum object–to–camera distance of 5 m. Since the position of the MAV is controlled by a GPS receiver, the horizontal accuracy ranges from 1 m to 5 m and decreases if the receiver is located close to walls. Furthermore, the height of the MAV is controlled by a barometric height sensor and therefore is prone to air pressure changes, which may lead to wrong height estimates.

**Simulation Results.** Our experiment is based on the House sequence that has been presented previously. The images have been acquired during manual flight with the *Falcon* MAV, meshed using our approach presented in Section 3.3.4, and geo-registered. Since the mesh contains more than 200.000 faces, we simplify the model to 7489 faces using a quadric edge collapsing algorithm [67]. This preserves the main structure of the building, but removes fine details.

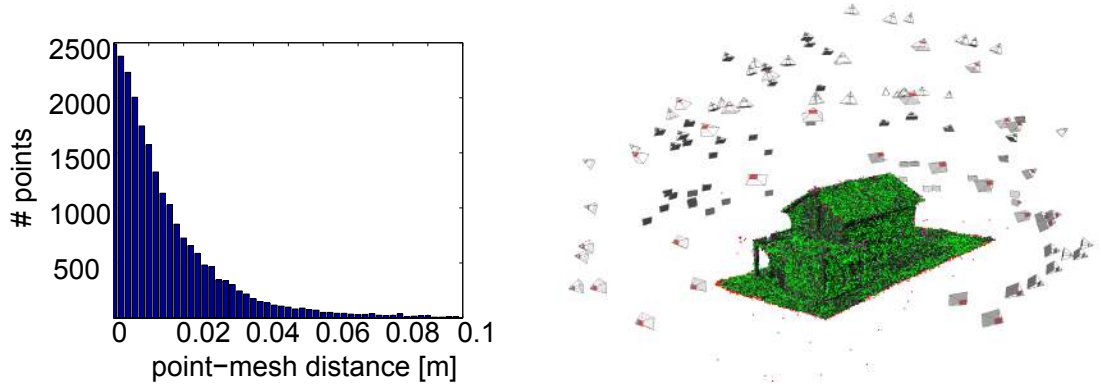
We claim a GSD of $s = 8\,mm$ and a focal length $f = 24\,mm$ resulting in a distance parameter $d = 20\,m$. Given the prior House geometry, our algorithm selects 4643 camera positions that are reachable by the MAV out of 7446 proposals. The remaining camera positions are removed because they are too close to the building or to the ground. From this set, the algorithm selects 106 poses which are necessary to achieve the desired GSD.

Since the mesh is untextured, we project a random texture generated from patches of written text onto each triangle. We then render the mesh for the determined camera position using OpenGL. The intrinsic parameters of the OpenGL camera are set to match those of the MAV camera. The rendered images serve as input for our SfM algorithm.

When running our 3D reconstruction algorithm, we are able to establish 106 (all) localized camera poses and 23.645 feature points. To measure the accuracy of the reconstructed sparse points we compute their normal distance to the closest triangle. Only 597 (2.5%) points have a distance larger than 0.5 m and can be classified as outliers. 1235 (5.2%) points range between 0.05 m and 0.5 m and the remaining 92.3% are closer than 0.05 m to

the mesh. Figure 6.30(a) shows the histogram of the point-to-mesh distances.

The camera pose error $cpr = \|C_{sfm} - C_{rendered}\|^2$ measures the difference between the pose estimated by the SfM approach $C_{sfm}$ and the rendering viewpoint $C_{rendered}$. The average camera pose error for all cameras on the single family house scene is $0.022\,m$ and the maximum error is $0.033\,m$. Figure 6.30(b) shows the estimated camera poses and the triangulated feature points.




(a) (b)

Figure 6.30: View planning results for the simulated House scene. (a) The histogram of distances between reconstructed SfM points and the groundtruth surface shows that 92% of the reconstructed points have a maximum distance of 0.05 m to the surface. (b) Selected views and SfM result of the House scene. Our approach selects 106 camera positions (gray) out of 4643 possible positions. Groundtruth camera positions are shown in red. The color coding of the reconstructed feature points represents the distance $t$ to the mesh (red: $t > 0.5m$, magenta: $0.5m > t > 0.05m$, green: $t < 0.05m$).

**Real-World Results.** In contrast to the simulated experiment where images are rendered from noise-free camera positions, we investigate the performance of our algorithm when taking images acquired by an autonomously flying *Falcon* MAV.

For the real-world experiment we use the same prior geometry model as before, but also take neighboring buildings and trees into account and therefore set $d = 15\,m$. After running our view–planning algorithm we obtain 133 desired camera poses and a respective flight plan according to our algorithm in Section 5.1.3.

Our SfM algorithm reconstructs 20.762 feature points based on 81 images. The discrepancy between the number of acquired images and the images which are successfully aligned by SfM can be explained by the lack of positioning accuracy using GPS. The MAV is equipped with a standard GPS receiver whose accuracy ranges between 1 m and about 5 m, and it can approach a GPS waypoint with a precision of about 2 m. In the worst case

both errors sum up and lead to a positioning error of 7 m. As a result, some views do not observe the expected parts of the object. Figure 6.31 shows the difference between the expected view and the image acquired by the MAV. Furthermore, we assume a discriminative texture, which cannot be guaranteed in the real-world example. Another challenge are too large geometric changes over time that are not reflected in the input mesh.

(a) (b)

Figure 6.31: Comparison of planned view and approached position. View (a) is rendered from the desired viewpoint, but the image taken by the MAV in (b) shows a slightly different part of the scene. The two poses mainly differ by a translation along the building caused by noisy GPS positions.

To analyze if all parts of the building are observed by the 81 aligned images, we run a semi-dense PMVS [65] reconstruction. The result shown in Figure 6.32 indicates that most parts of the surface have been captured by the MAV. The holes in the facade are due to missing texture that prevents accurate patch-matching. This result illustrates that even a subset of the selected cameras is sufficient to capture most parts of the object. However, for guaranteeing a certain accuracy of the reconstructed points, the entire planned set of cameras is required.

### 6.3.2 Trajectory Control based on Visual Localization

In a more local environment, our visual localization methods can be used for vision-based trajectory following. We performed several experiments to evaluate the accuracy of our fuzzy controller for position-based visual servoing. To obtain these results, we used the *Pelican* MAV and streamed images with a resolution of $320 \times 240$ px at 30 Hz via a wireless 802.11n link to the ground station. To ensure a reliable connection for control commands, we send control commands over a separate X-Bee wireless data link directly to the MAV's autopilot.

Note that a fair comparison of our visual servoing approach to other work would only be possible in a simulated environment. All results depend on the scene, the distance to the scene and for outdoor experiments on the wind conditions.

Figure 6.32: Densified reconstruction after executing an autonomous flight plan. This is result is achieved by using patch-based densification [65]. The missing parts of the facade are caused by weak texturing.

**Hovering Results.** For measuring the hovering accuracy in an outdoor setting we hovered directly above the take-off position at an altitude of 1.5 $m$ and a horizontal mean distance to the feature points in the map of about 10 $m$. As error measurement we use the Root Mean Square error (RMS) in individual directions, and we compare our results to the work of Achtelik et al. [2] in Table 6.6. It can be seen that we are outperforming their approach, although we do not yet incorporate IMU measurements and employ a non-nadir view with larger distance to the scene. An example for the hovering trajectory is visualized in Figure 5.8.

Table 6.6: Performance for outdoor hovering. The RMS and maximum error for the $z$ and $xy$-plane and in 3D space ($xyz$) are compared to [2].

| Approach | Distance [m] | RMS error [m] | | | Max error [m] | | |
|---|---|---|---|---|---|---|---|
| | | $z$ | $xy$ | $xyz$ | $z$ | $xy$ | $xyz$ |
| Ours, 320 × 256 px | 10.0 | 0.0643 | 0.1495 | 0.1627 | 0.2189 | 0.2716 | 0.3471 |
| Achtelik et al.[2] | 3.3 | 0.11 | 0.44 | – | – | – | – |

**Trajectory Flight Results.** Not only the accuracy at a fixed position is important, but also the trajectory between defined waypoints is of interest. Therefore, we evaluated the accuracy of flight trajectories during an indoor experiment. We defined a 2 × 2 m square trajectory sampled in path lengths of 1.0 m at an altitude of 1.5 m as depicted in Figure 6.33 and measured the RMS error in individual directions. The results are

compared to those of Blösch et al. [21] in Table 6.7. Our results can compete, and we are not dependent on a wired connection between MAV and ground station. Furthermore, we do not rely on a textured ground plane which is often not available when flying above a road or grass, and our approach does not require an accurate system model of the MAV.

Table 6.7: Performance for the trajectory flight. The RMS and maximum error for $xyz$ are evaluated by sampling along the trajectory in 0.5 mm steps.

| Approach | RMS error [m] | | | |
|---|---|---|---|---|
| | $x$ | $y$ | $z$ | $xyz$ |
| Ours, $320 \times 256$ px | 0.0704 | 0.1109 | 0.0663 | 0.1471 |
| Blösch et al. [21] | 0.0995 | 0.0748 | 0.0423 | – |

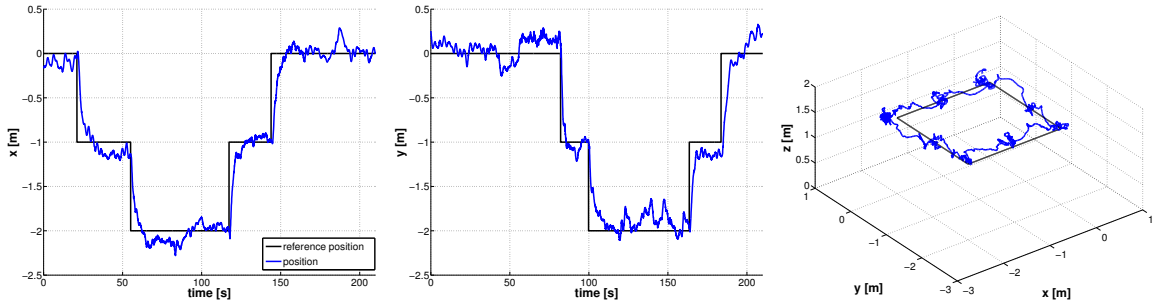


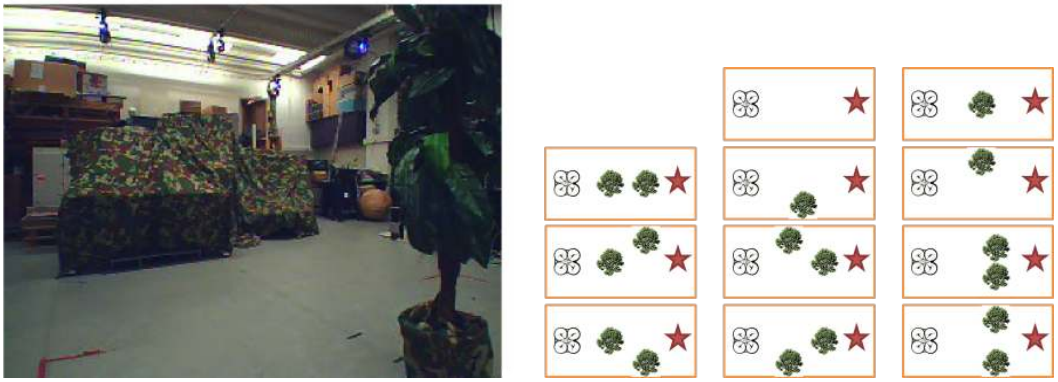

Figure 6.33: Visual servoing trajectory along a predefined $2.0 \times 2.0$ m square path. The first two figures show the temporal change of the $x$ and $y$ coordinates and the third the spatial representation of the trajectory in 3D space. Waypoints are defined in steps of 1.0 m, thus the vehicle briefly hovers at these spots.

### 6.3.3 Imitation Learning for Reactive Visual Control

For our experiments on reactive visual control, the *ARDrone* MAV is employed. When experimenting with learned control strategies, crashed are unavoidable. Moreover, high-speed flight in a forest is an extremely challenging tasks where a confidence of 100% during collision avoidance cannot be achieved yet. Due to its robust structure and cheap spare parts, the *ARDrone* is perfectly suited for this task.

**Indoor Experiments.** We first tested our approach indoors in a motion capture arena. We use fake indoor trees as obstacles and camouflage to hide background clutter (Figure 6.34(a)). While this is a very controlled environment that lacks many of the complexities of real outdoor scenes, it allows us to obtain better quantitative results to determine

the effectiveness of our approach. The motion capture system is only used to track the *ARDrone* and adjust its heading so that it is always heading straight towards a given goal location. The MAV moves at a fixed altitude and forward velocity of 0.35 m/s. We learned a controller that controls the left-right velocity using DAgger over 3 training iterations. At each iteration, we used 11 fixed scenarios to collect training data, including 1 scenario without obstacles, 3 with one obstacle, and 7 with two obstacles (Figure 6.34(b)).

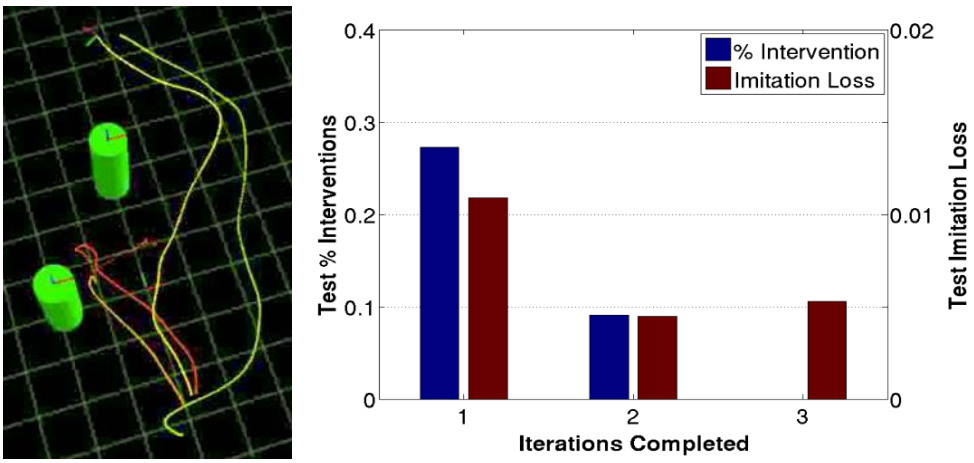


(a) (b)

Figure 6.34: Indoor setup for reactive control experiments. (a) Motion capture arena with fake plastic trees and camouflage in background. (b) 11 obstacle arrangements used to train DAgger for every iteration in the motion capture arena. The star indicates the goal location.

Figure 6.35(a) qualitatively compares the trajectories taken by the MAV in the motion capture arena after each iteration of training on one of the particular scenario. In the first iteration, the green trajectory to the farthest right is the demonstrated trajectory by the human expert pilot. The short red and orange trajectories are the trajectories taken by the MAV after the first and second iterations were completed. Note that both fail to avoid the obstacle. After the third iteration, however, the controller learned a trajectory which avoids both obstacles. The percentage of scenarios where the pilot had to intervene for the learned controller after each iteration can be found in Figure 6.35(b). The number of required interventions decreases between iterations, and after 3 iterations there was no need to intervene as the MAV successfully avoided all obstacles in all scenarios.

**Feature Evaluation.** After verifying the general functionality of our approach, we evaluate the benefit of all four feature types. An ablative analysis on the data shows that the structure tensor features are most important, followed by Laws features. Figure 6.36 shows how the contribution of different features varies for different control signal strengths. Optical flow, for example, carries little information in scenes where small commands are predicted. This is intuitive since in these cases there are typically no close obstacles and




(a) (b)

Figure 6.35: Indoor results for training DAgger. (a) Improvement of trajectory by DAgger over the iterations. The rightmost green trajectory is the pilot demonstration. The short trajectories in red & orange show the controller learned in the first and second iterations which fail. In the third iteration the controller successfully avoids both obstacles and is similar to the demonstrated trajectory. (b) Percentage of scenarios the pilot had to intervene and the imitation loss (average squared error in controls of controller to human expert on hold-out data) after each iteration of DAgger. After 3 iterations, there was no need for the pilot to intervene and the MAV could successfully avoid all obstacles.

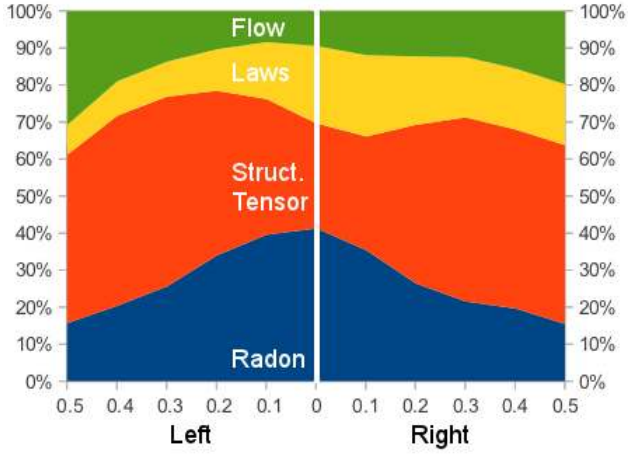
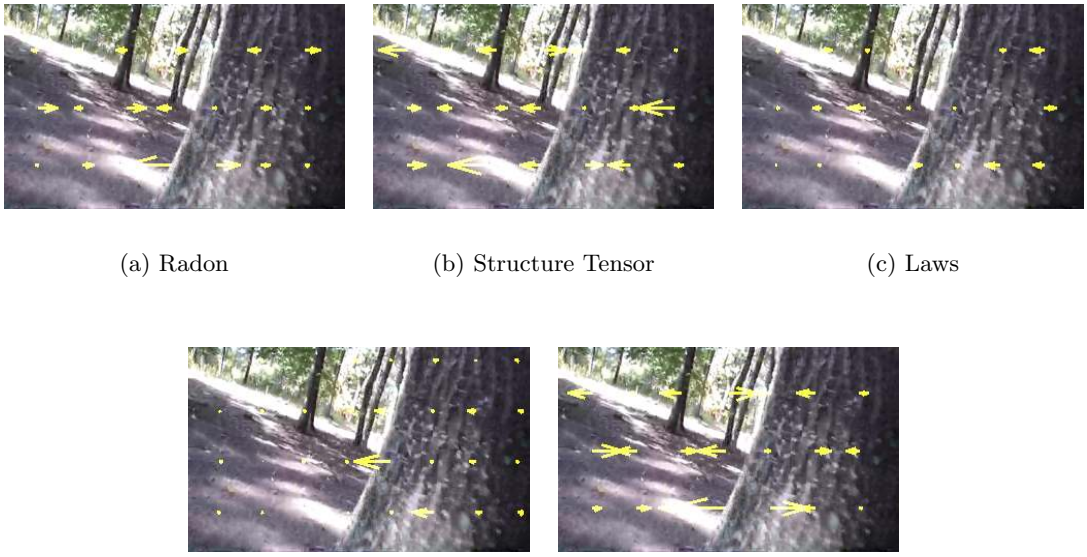



Figure 6.36: Breakdown of the contribution of the different features for different control prediction strengths, averaged over 9389 datapoints. Laws and Radon are more significant in cases when small controls are performed (e.g. empty scenes), whereas the structure tensor and optical flow are responsible for strong controls (i.e. in cases where the scene contains an imminent obstacle). A slight bias to the left can be seen, which is consistent to observations in the field.



(a) Radon (b) Structure Tensor (c) Laws

(d) Flow (e) Combined Features

Figure 6.37: Visualization of the contribution of the different features to the predicted control. The overall control was a hard left command. The arrows show the contribution of a given feature at every window. Structure tensor features have the largest contribution in this example, while Radon has the least.

subsequently no significant variation in optical flow. In fact, removing the optical flow feature on platforms without sufficient computational capabilities only results in a 6.5% increase in imitation loss.

Figure 6.37 shows the contribution of each of the features at different window centers in the image. While structure tensor features mainly fire due to texture in the background and thus indicate free space, strong optical flow vectors correspond to very close objects. In this example the predictor commands a hard left turn (numerical value: 0.47L on a scale of [0,1]), and all visual features contribute to this. Consistent with the above analysis, the contribution of the structure tensor was greatest (0.38L), Laws masks and optical flow contribute the same (0.05L) while Radon features provide the least contribution (0.01L). In this particular example, the non-visual features actually predict a small right command (0.02R).

**Outdoor Experiments.** After validating our approach indoors in the motion capture arena, we conducted experiments outdoors to test in real-world scenarios. As we could not use the motion capture system outdoors to make the MAV head towards a specific goal location, we made the vehicle move forward at a fixed speed and aimed for learning a controller that would swerve left or right to avoid any trees on the way, while maintaining

(a) (b) (c)

Figure 6.38: Typical failures during training iterations. While the controller has problems with tree trunks during the first iteration (a), this improves considerably towards the third iteration, where mainly foliage causes problems (b). Over all iterations, the most common failures are due to the narrow FOV of the camera where some trees barely appear to one side of the camera or are just hidden outside the view (c). When the UAV turns to avoid a visible tree a bit farther away it collides with the tree to the side.

the initial heading. Training and testing were conducted in forest areas while restraining the aircraft using a light-weight tether. We performed two experiments with DAgger to evaluate its performance in different regions, one in a park with relatively low tree density, and another in a dense forest.

The first area is a park area with a low tree density of approximately one tree per $12 \times 12\,\text{m}$, consisting mostly of large trees and a few thinner trees. In this area we flew at a fixed velocity of around $1\,\text{m/s}$, and learned a heading (left-right) controller for avoiding trees using DAgger over 3 training iterations. This represented a total of $1\,\text{km}$ of flight training data. Then, we exhaustively tested the final controller over an additional $800\,\text{m}$ of flight in the training area and a separate test area.

Qualitatively, we observed that the behavior of the reactive controller improved over iterations. After the first iteration of training, the MAV sometimes failed to avoid large trees even when they were in the middle of the image in plain view (Figure 6.38(a)). At later iterations however, this rarely occurred. On the other hand, we observed that the MAV had more difficulty detecting branches or bushes. The fact that fewer of such obstacles were seen in the training data, coupled with the inability of the human pilot to distinguish them from the background, contributed to the difficulty of dealing with these obstacles. We expect that better visual features or improved camera resolution might help, as small branches often cannot be seen in $320 \times 240$ pixel images.

As expected, we found that the narrow field-of-view was the largest contributor to failures of the reactive approach (Figure 6.38(c)). The typical issue occurs when the learned controller avoids a tree, and as it turns a new tree comes into view. This may cause the controller to turn in a way such that it collides sideways into the tree it just avoided. This problem inevitably afflicts purely reactive controllers and could be solved

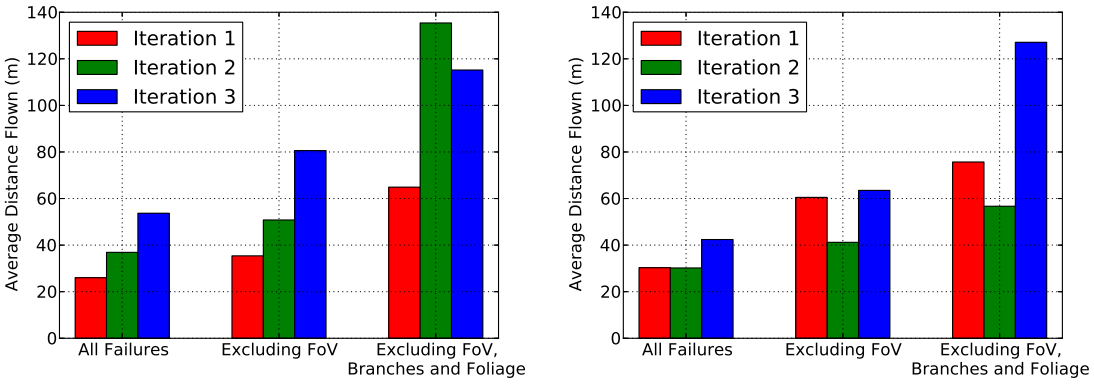



Figure 6.39: Average distance flown autonomously by the MAV before a failure. (a) Low-density region, (b) high-density region.

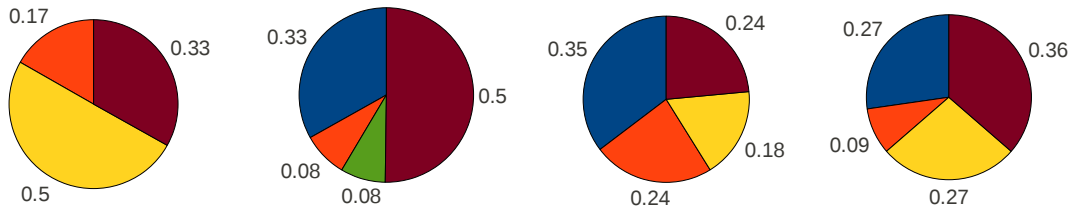
by adding a higher level of reasoning [14], or memory of recent visual features.

The type of failures are broken down by the type of obstacle the vehicle failed to avoid, or whether the obstacle was not in the FOV. Overall, 29.3% of the failures were due to a too narrow FOV and 31.7% on hard to perceive obstacles like branches and leaves.

Quantitatively, we compare the evolution of the average distance flown autonomously by the MAV before a failure occurred over the iterations of training. We compare these results when accounting for different types of failures in Figure 6.39(a). When accounting for all failure types, the average distance flown per failure after all iterations of training was around 50 m. On the other hand, when only accounting for failures that are not due to the narrow FOV, or branches/leaves, the average distance flown increases to around 120 m. For comparison, the pilot successfully flown over 220 m during the initial demonstrations, avoiding all trees in this sparse area.

To achieve these results the *ARDrone* has to avoid a significant number of trees. A tree is *avoided* when the MAV can see the tree pass from within its FOV to the edge of the image. Over all the data, we counted the number of times the MAV avoided a tree, and observed that it passed 1 tree every 7.5 m on average. We also checked whether the vehicle was actively avoiding trees by performing significant commands. A tree is *actively avoided* when the controller issues a command larger than 25% of the full range, passively in all other cases. 41% of the trees were passed actively by our MAV, compared to 54% for the human pilot.

We further tested whether the learned controller generalizes to new regions by testing it in a separate test area. The test area was slightly denser, around 1 tree per $10 \times 10$ m. The controller performed very well and was successfully able to avoid trees and perform at a similar level than in the training area. In particular, the MAV was able to fly




(a) Pilot (b) After iter. 1 (c) After iter. 2 (d) After iter. 3

Figure 6.40: Percentage of failures of each type for DAgger over the iterations of training in the high-density region. Blue: large trees, orange: thin trees, yellow: leaves and branches, green: other obstacles (poles, signs, etc.), brown: too narrow FOV. Clearly, a majority of the crashes happen due to a too narrow FOV and obstacles which are hard to perceive, such as branches and leaves.

autonomously without crashing in any trees over a 100 m distance, reaching the limit of our communication range for the tests.

The second set of experiments was conducted in a thickly wooded region. The tree density was significantly higher, around 1 tree per $3 \times 3$ m, and the area included a much more diverse range of trees, ranging from very small and thin to full-grown trees. In this area we flew at a faster fixed velocity of around 1.5 m/s, and again learned the heading (left-right) controller to avoid trees using DAgger over 3 iterations of training. This represented a total of 1.2 km of flight training data. The final controller was also tested over additional 400 m of flight in this area. For this experiment however, we used the new *ARDrone 2.0* quad-rotor helicopter, which has an improved camera that can stream $640 \times 360$ pixel images at 30 Hz. The increased resolution likely helped to detect the thinner trees.

Qualitatively, in this experiment we observed that the performance of the learned behavior slightly decreased in the second iteration, but then improved significantly after the third iteration. For example, we observed more failures to avoid both thin and large trees in the second iteration compared to the other iterations. This is shown in Figure 6.40, which compares the percentage of the different failures for the human pilot and after each iteration of DAgger in this area. We can also observe that the percentage of failures attributed to large or thin trees is smallest after the third iteration, and that again a large fraction of the failures occur when obstacles are not visible in the FOV of the MAV. Additionally, we can observe that the percentage of failures due to branches or leaves diminishes slightly over the iterations, which could be attributed to the better camera that can better perceive these obstacles. A visualization of a typical sequence is given in Figure 6.41 and 6.42.

Quantitatively, we compare the evolution of the average distance flown autonomously by the MAV before a failure occurred over the iterations of training. Again, we compare these results when accounting for different types of failures in Figure 6.39(b). When


MAV's on-board view Observer's view

Figure 6.41: Example flight in a dense forest area. The image sequence is chronologically ordered from top ($t = 0s$) to bottom ($t = 2.9s$) and split into the MAV's on-board view on the left and an observer's view to the right. Note the direction label of the MAV in the first frame, and the color-coded commands issued by DAgger. It can be observed that after avoiding tree $A$ in frame 3 the vehicle still rolls strongly to the left in frame 4. This is due to the small but ubiquitous latency and should be addressed in future work to fly the MAV in even denser areas.


MAV's on-board view Observer's view

Figure 6.42: Example flight in a dense forest area, continued. The image sequence is chronologically ordered from top ($t = 3.8s$) to bottom ($t = 6.6s$) and split into the MAV's on-board view on the left and an observer's view to the right. In frames 1-3, tree $B$ is avoided on the left, rather than on the more intuitive right. DAgger prefers this decision based on the drift feature, which indicates that the vehicle still moves left and thus a swerve to the right would be more difficult.

accounting for all failure types, the average distance flown per failure after all iterations of training was around 40 m. Surprisingly, despite the large increase in tree density and faster forward velocity, this is only slightly worse than our previous results in the sparse region. Furthermore, when only accounting for failures that are not due to the narrow FOV or branches and leaves, the average distance flown increases to 120 m per failure, which is on par with our results in the sparser area. For comparison, when only accounting for failures due to tree trunks, the pilot flew around 500 m during the initial demonstrations and only failed to avoid one thin tree. However, the pilot also failed to avoid thin branches and foliage more often (Figure 6.40). When accounting for all types of obstacles, the pilot's average distance until failure in this high-density forest was around 80 m.

The increase in tree density required our MAV to avoid a significantly larger number of trees to achieve these results. Over all the data, we observed that it was avoiding on average of 1 tree every 5 m. In this dense region, both the human pilot and the reactive controller had to use larger commands to avoid all the trees, leading to an increase in the proportions of trees that were passed actively. 62% of the trees we passed actively by the MAV, compared to a similar ratio of 66% for the pilot.

The major issue of the too narrow FOV presented in this section may be addressed by two approaches in the future. First, imitation learning methods that integrate a small amount of memory may allow to overcome the simplest failure cases without resorting to a complete and expensive mapping of the environment. Second, the biologically-inspired solution is to simply ensure a wider FOV for the camera system. For example, pigeons rely mostly on monocular vision and have a FOV more than 3 times larger, while owls have binocular vision with around 1.5 times the FOV of the *ARDrone*.

## 6.4 Summary

In this chapter, we have evaluated the core components of this thesis. We were able to show that we can accurately reconstruct geometric outdoor structures to represent prior knowledge and that we can quickly localize an MAV in a global coordinate system using monocular vision only. We have presented an experimental evaluation of our live dense reconstruction system with distributed SLAM, and we have shown our visual navigation capabilities by adding a path planning and control component. Further illustrative results in form of videos can be found online[3].

[3] `http://aerial.icg.tugraz.at`

# Chapter 7

# Conclusion

**Contents**



In this thesis, we have presented a scalable framework for visual navigation of micro aerial vehicles which incorporates the rich set of already available knowledge about geometric structures in the world to enhance localization quality and robustness. The presented algorithms facilitate autonomous flight in cluttered urban outdoor environments based on a monocular camera as the main exteroceptive sensor of the MAV. To conclude, we summarize our main contributions and give directions for future research.

## 7.1 Summary

Our first contribution is a toolbox of approaches for modeling geometric prior knowledge. We have presented methods for offline and online Structure-from-Motion reconstruction which result in sparse point clouds. Our generic reconstruction pipeline is widely applicable since no prior knowledge about the scene is necessary. For scenes with few keypoints, 3D line segments can be used to densify the resulting model. We have presented an approach for 3D line-based scene reconstruction without explicit appearance-based matching, which is suitable for solid but also for wiry objects. We have further discussed multi-view stereo techniques for the densification of point clouds and for dense surface extraction, including space carving-based meshing and a variational optimization framework for volumetric representations. While sparse, feature-based point clouds are used for localization, we rely on dense geometry for alignment, occlusion handling, collision avoidance for path planning, and visualization. Dense matching techniques are also required to build large-scale digital surface models. We have presented a probabilistic method for range image integration that considers depth hypotheses as samples from an underlying probability density function. Our approach scales very well because it avoids a volumetric voxel representation

for fusion. Finally, we have proposed quality measures which support users during the difficult task of acquiring all necessary images for Structure from Motion reconstruction of a scene. We have discussed the visualization of the expected redundancy and resolution of a reconstruction, and we have shown how to select suitable views automatically based on prior knowledge about the geometry.

Our second contribution concerns the perception part of the navigation problem. A large amount of related work relies on Simultaneous Localization and Mapping techniques, but especially monocular visual SLAM requires considerable amounts of computational power to operate in real-time. The few available methods suffer from error accumulation during tracking and from missing robustness to cope with large-scale outdoor environments. In contrast, we have presented a multi-scale framework for visual localization and mapping which exploits the previously modeled geometric priors and thus is not affected by these issues. On the top level we have introduced an algorithm for the automatic alignment of 3D reconstructions in a world coordinate system. Given a georeferenced digital surface model and approximate GPS tags for the individual camera positions, we refine the alignment based on the correlation of orthographic depth maps. The localization accuracy mainly depends on the ground sampling distance of the DSM, which is in the range of 5-30 cm per pixel for modern cameras. For global localization within these geometric priors, we have further proposed the concept of virtual views in 3D space which allows to partition the search space and thus is completely scalable. Our approach also supports the integration of in-flight information to improve the initial scene representation. Pose estimates can be delivered at 4 fps with an accuracy comparable to differential GPS, which allows to switch seamlessly between GPS and visual localization. However, global localization is still too computationally heavy to run onboard micro aerial vehicles, and changes in geometry as well as unknown areas have not yet been covered. Thus, on the lowest layer of our framework we have shown how state-of-the-art approaches to online localization and mapping can be adapted to a distributed environment. We run a full-framerate pose tracker on the mobile robot, but only transmit selected keyframes to the server for global localization and reconstruction. A deliberate implementation then allows to update the local map of the tracker, while at the same time providing dense volumetric reconstructions to a possible user or to a collision avoidance system. Finally, we have presented an online localization method which extends visual SLAM by incorporating line-based priors. This allows to estimate the pose of the camera relative to complex and possibly wiry structures. Such a tracking component is crucial when discriminative keypoints are missing, which is the case in many man-made environments, or for applications like power pylon inspection.

Our third contribution is the application of computer vision methods to path planning and control. We have shown that digital surface models can be used as prior knowledge for high-level path and view planning, and we have combined our distributed visual localization methods with fuzzy control techniques to build a visual navigation system. Finally, we have demonstrated a novel approach for high-speed, autonomous MAV flight through a dense forest environment. Our system learns to predict how a human expert would con-

trol the vehicle in a similar situation, and thus successfully avoids collisions using passive, low-cost and low-weight visual sensors only.

After having quantitatively and qualitatively evaluated all parts of this thesis, we conclude that highly accurate geometric prior knowledge indeed helps to build a scalable, accurate, and computationally feasible visual navigation system for micro aerial vehicles.

## 7.2 Directions for Future Research

In this thesis, we have presented a variety of approaches to overcome problems of the current state-of-the-art in visual navigation, localization, and mapping. However, the challenging task of visual navigation is by far not solved yet. In the following paragraphs, we thus conclude our work with suggestions for future research directions.

**Appearance Variations.** Outdoor environments are subject to severe changes in illumination and appearance, spanning from the regular variation in solar illumination during a day to weather-dependent changes, and further to seasonal variations. While navigation approaches should ideally be invariant to those changes in appearance, the underlying algorithms are only to a certain degree. The major problem is the correspondence computation between images, which is already a challenge when images of the exactly same scene are taken at a sunny day and at a cloudy day. Even visual SLAM approaches have problems in such situations, although the change is much more gradual due to the high framerate. In Section 4.1.3 we have proposed to geometrically align several models of the same location, acquired at different days and in different seasons. The geo-registered feature clouds can then be used to perform season-invariant matching. While this approach is of course feasible, a lot of imagery has to be acquired at the right time to successfully model the appearance changes. Thus, a trade-off between feature invariance, feature mismatches due to invariance, and modeling effort needs to be researched. Again, prior knowledge on expected weather changes could be used to predict the possible appearance of features in real-time.

**Weakly Textured Scenes.** Another challenge for keypoint-based reconstruction and localization are weakly textured scenes. We have tackled this issue by introducing line-based methods in Section 3.2 and 4.4, and we have emphasized the benefits of such approaches. Another concept is to exploit the full, dense image data rather than extracting a sparse representation. If the prior geometry is modeled accurately enough, dense image data can be compared and aligned to projected dense model data. This works not only for appearance information, but also for 2.5D depths [152, 153]. The only drawback of these methods is their considerable need for computational power. However, both line-based and dense geometry concepts should be further investigated.

**Explorative Navigation based on the Perception Quality.** For successful localization in dynamic, real-world outdoor environments, at least the two issues discussed before have to be overcome. However, navigation also involves a control component, and thus it is possible to define where a robot should ideally go to optimize its perception quality. We propose to research a multi-objective optimization problem which incorporates four components: The predicted localization quality in terms of pose uncertainty, the likelihood for successful map expansions, the safety of commanded MAV poses based on a probabilistic occupancy grid of the local neighborhood, and finally, the decrease in distance to the target location. Such an explorative approach is in our opinion the fastest way towards fully autonomous visual navigation.

**Large-Scale System Design.** Another important aspect of future research should be thorough and extensive experimental evaluation on a systems level. While we have presented a series of experiments which have been designed to test the individual components, effective visual navigation can only be tested on a much larger scale. Such experiments require considerable implementation efforts as distributed systems with server-based data management and intelligent, probabilistic integration of new scene knowledge have to be created. Luckily, initiatives such as RoboEarth [216] have recently presented first steps towards this more general goal of sharing knowledge between robots, which has been coined *Cloud Robotics* [72]. We expect that it will be possible to distribute localization and mapping services in a web for robots in the near future.

**Applications.** Finally, we propose to further investigate the applications for visual navigation that we have presented in this thesis. Our work establishes a basis for automated power pylon inspection, construction site monitoring with automatic view planning, and large-scale but cost efficient city modeling. In all of these tasks, imagery delivered and processed by a micro aerial vehicle is highly beneficial. Furthermore, most of our algorithms can also be applied to other emerging applications of visual localization, mapping, and navigation, such as augmented reality and self-driving cars.

# Appendix A

# Publications

The publications created during the course of this thesis are grouped by topic and roughly sorted by date.

## A.1 Sparse and Dense 3D Reconstruction

- *Photogrammetric Camera Network Design for Micro Aerial Vehicles.* Christof Hoppe, Andreas Wendel, Stefanie Zollmann, Katrin Pirker, Arnold Irschara, Horst Bischof, and Stefan Kluckner. In Proceedings of the Computer Vision Winter Workshop (CVWW), 2012.
- *Rapid 3D City Model Approximation from Publicly Available Geographic Data Sources and Georeferenced Aerial Images.* Markus Rumpler, Arnold Irschara, Andreas Wendel, and Horst Bischof. In Proceedings of the Computer Vision Winter Workshop (CVWW), 2012.
- *Dense Reconstruction On-the-Fly.* Andreas Wendel, Michael Maurer, Gottfried Graber, Thomas Pock, and Horst Bischof. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2012.
- *Online Feedback for Structure-from-Motion Image Acquisition.* Christof Hoppe, Manfred Klopschitz, Markus Rumpler, Andreas Wendel, Stefan Kluckner, Horst Bischof, and Gerhard Reitmayr. In Proceedings of the British Machine Vision Conference (BMVC), 2012.
- *Line-based 3D Reconstruction of Wiry Objects.* Manuel Hofer, Andreas Wendel, and Horst Bischof. In Proceedings of the Computer Vision Winter Workshop (CVWW), 2013.
- *Probabilistic Range Image Integration for DSM and True-Orthophoto Generation.* Markus Rumpler, Andreas Wendel, and Horst Bischof. In Proceedings of the Scandinavian Conference on Image Analysis (SCIA), 2013.

## A.2 Alignment and Fusion of Point Clouds

- *Automatic Alignment of 3D Reconstructions using a Digital Surface Model.* Andreas Wendel, Arnold Irschara, and Horst Bischof. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2011.

- *Geo-Referenced 3D Reconstruction: Fusing Public Geographic Data and Aerial Imagery.* Michael Maurer, Markus Rumpler, Andreas Wendel, Christof Hoppe, Arnold Irschara, Horst Bischof. In Proceedings of the International Conference on Robotics and Automation (ICRA), 2012.

- *Automatic Fusion of Partial Reconstructions.* Andreas Wendel, Christof Hoppe, Horst Bischof, and Franz Leberl. In Annals of the International Symposium on Photogrammetry and Remote Sensing (ISPRS), 2012.

## A.3 Image-based Localization

- *Natural Landmark-based Monocular Localization for MAVs.* Andreas Wendel, Arnold Irschara, and Horst Bischof. In Proceedings of the International Conference on Robotics and Automation (ICRA), 2011.

- *Online Model-Based Multi-Scale Pose Estimation.* Thomas Kempter, Andreas Wendel, Horst Bischof. In Proceedings of the Computer Vision Winter Workshop (CVWW), 2012.

- *Visual Landmark-based Localization for MAVs using Incremental Feature Updates.* Andreas Wendel, Michael Maurer, and Horst Bischof. In Proceedings of the Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT), 2012.

- *Visual Localization for Micro Aerial Vehicles in Urban Outdoor Environments.* Andreas Wendel and Horst Bischof. Advanced Topics in Computer Vision, G.M. Farinella, S. Battiato, R. Cipolla (eds.), Springer London, 2013.

## A.4 Control of Robotic Systems

- *Fuzzy Visual Servoing for Micro Aerial Vehicles.* Andreas Wendel, Michael Maurer, Mario Katusic, and Horst Bischof. In Proceedings of the Austrian Robotics Workshop (ARW), 2012.

- *Learning Monocular Reactive UAV Control in Cluttered Natural Environments.* Stephane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, James Andrew Bagnell, and Martial Hebert. In Proceedings of the International Conference on Robotics and Automation (ICRA), 2013.

## A.5 Applications

- *3D Vision Applications for MAVs: Localization and Reconstruction.* Andreas Wendel, Michael Maurer, Arnold Irschara, and Horst Bischof. In Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), 2011.

- *Construction Site Monitoring from Highly-Overlapping MAV Images.* Stefan Kluckner, Josef A. Birchbauer, Claudia Windisch, Christof Hoppe, Arnold Irschara, Andreas Wendel, Stefanie Zollmann, Gerhard Reitmayr, and Horst Bischof. In Proceedings of the IEEE International Conference on Advanced Video- and Signal-based Surveillance (AVSS), Industrial Session, 2011.

- *Airborne Inspection using Single-Camera Interleaved Imagery.* Michael Maurer, Andreas Wendel, and Horst Bischof. In Proceedings of the Computer Vision Winter Workshop (CVWW), 2012.

- *Automated Photogrammetry for Three-Dimensional Models of Urban Spaces.* Franz Leberl, Philipp Meixner, Andreas Wendel, and Arnold Irschara. Optical Engineering, 51(2). SPIE, 2012.

## A.6 Other Publications

- *Scene Categorization from Tiny Images.* Andreas Wendel and Axel Pinz. In Proceedings of the Annual Workshop of the Austrian Association for Pattern Recognition, 2007.

- *Facade Segmentation from Streetside Images.* Andreas Wendel and Horst Bischof. In Proceedings of the Computer Vision Winter Workshop (CVWW), 2010.

- *Unsupervised Facade Segmentation using Repetitive Patterns.* Andreas Wendel, Michael Donoser, and Horst Bischof. In Proceedings of the German Pattern Recognition Conference (DAGM), 2010.

- *Facade Segmentation in a Multi-View Scenario.* Michal Recky, Andreas Wendel, and Franz Leberl. In Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization, and Transmission (3DIMPVT), 2011.

- *Building Facade Separation in Vertical Aerial Images.* Philipp Meixner, Andreas Wendel, Franz Leberl, and Horst Bischof. In Annals of the International Symposium on Photogrammetry and Remote Sensing (ISPRS), 2012.
- *Proceedings of the 16th Computer Vision Winter Workshop.* Andreas Wendel, Sabine Sternig, and Martin Godec. Verlag der Technischen Universität Graz, Austria, 2011.

# Bibliography

[1] Abdelkrim, N., Aouf, N., Tsourdos, A., and White, B. (2008). Robust nonlinear filtering for INS/GPS UAV localization. In *Mediterranean Conference on Control and Automation*.

[2] Achtelik, M., Achtelik, M., Weiss, S., and Siegwart, R. (2011). Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[3] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building Rome in a day. In *IEEE International Conference on Computer Vision (ICCV)*.

[4] Amenta, N. and Bern, M. (1999). Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504.

[5] Anan, C. S. and Hartley, R. I. (2008). Optimised KD-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[6] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

[7] Attali, D. and Boissonnat, J.-D. (2004). A linear bound on the complexity of the Delaunay triangulation of points on polyhedral surfaces. *Discrete & Computational Geometry*, 31(3):369–384.

[8] Bachrach, A., He, R., and Roy, N. (2009). Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228.

[9] Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A. S., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2012). Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. *International Journal on Robotics Research (IJRR)*, 31(11):1320–1343.

[10] Baillard, C., Schmid, C., Zisserman, A., and Fitzgibbon, A. (1999). Automatic line matching and 3D reconstruction of buildings from multiple views. In *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*.

[11] Bay, H., Ferrari, V., and Van Gool, L. (2005). Wide-baseline stereo matching with line segments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*.

[13] Beder, C. and Steffen, R. (2006). Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence. In *German Conference on Pattern Recognition (DAGM)*.

[14] Bellingham, J., Richards, A., and How, J. (2002). Receding horizon control of autonomous aerial vehicles. In *American Control Conference*.

[15] Bergerman, M., Amidi, O., Miller, J. R., Vallidis, N., and Dudek, T. (2007). Cascaded position and heading control of a robotic helicopter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[16] Bernier, R., Bissonnette, M., and Poitevin, P. (2005). DSA radar development report. In *Association for Unmanned Vehicle Systems International (AUVSI)*.

[17] Beyeler, A., Zufferey, J., and Floreano, D. (2009). Vision-based control of near-obstacle flight. *Autonomous Robots*, 27(3):201–219.

[18] Bills, C., Chen, J., and Saxena, A. (2011). Autonomous MAV flight in indoor environments using single image perspective cues. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[19] Bin, F., Fuchao, W., and Zhanyi, H. (2012). Robust line matching through line-point invariants. *Pattern Recognition*, 45(2):794–805.

[20] Bleser, G., Wuest, H., and Stricker, D. (2006). Online camera pose estimation in partially known and dynamic scenes. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[21] Blösch, M., Weiss, S., Scaramuzza, D., and Siegwart, R. (2010). Vision based MAV navigation in unknown and unstructured environments. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[22] Boissonnat, J.-D. and Yvinec, M. (1998). *Algorithmic Geometry*. Cambridge University Press, UK.

[23] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239.

[24] Bry, A., Bachrach, A., and Roy, N. (2012). State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[25] Burgard, W., Fox, D., Hennig, D., and Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids. In *National Conference on Artificial Intelligence*.

[26] Chen, S. Y., Li, Y. F., Zhang, J. W., and Wang, W. L. (2008). *Active Sensor Planning for Multiview Vision Tasks*. Springer-Verlag.

[27] Choset, H., Lynch, K. M., Hutchinson, S., and Kantor, G. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

[28] Collins, R. T. (1996). A space-sweep approach to true multi-image matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[29] Conroy, J., Gremillion, G., Ranganathan, B., and Humbert, J. (2009). Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous Robots*, 27(3):189–198.

[30] Conway, A. R. (1995). *Autonomous control of an unstable model helicopter using carrier phase GPS only*. PhD Thesis, Stanford University, USA.

[31] Cramer, M. (2010). The DGPF-Test on Digital Airborne Camera Evaluation - Overview and Test Design. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 2010(2):73–82.

[32] Crowley, J. (1989). World modeling and position estimation for a mobile robot using ultra-sonic ranging. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[33] Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research (IJRR)*, 27(6):647–665.

[34] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *ACM SIGGRAPH*.

[35] David, P., DeMenthon, D. F., Duraiswami, R., and Samet, H. (2002). SoftPOSIT: Simultaneous Pose and Correspondence Determination. In *European Conference on Computer Vision (ECCV)*.

[36] Davies, E. (1997). *Machine vision: Theory, algorithms, practicalities*. Morgan Kaufmann.

[37] Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *IEEE International Conference on Computer Vision (ICCV)*.

[38] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067.

[39] Dementhon, D. F. and Davis, L. S. (1995). Model-Based Object Pose in 25 Lines of Code. *International Journal of Computer Vision (IJCV)*, 15(1):123–141.

[40] DeSouza, G. N. and Kak, A. C. (2002). Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(2):237–267.

[41] Dey, D., Geyer, C., Singh, S., and Digioia, M. (2011). A cascaded method to detect aircraft in video imagery. *International Journal on Robotics Research (IJRR)*, 30(12):1527–1540.

[42] Dey, D., Liu, T. Y., Sofman, B., and Bagnell, J. A. (2012). Efficient optimization of control libraries. In *AAAI Conference on Artificial Intelligence*.

[43] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

[44] Dissanayake, G., Newman, P. M., Clark, S., Durrant-Whyte, H., and Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

[45] Doyle, J., Francis, B., and Tannenbaum, A. (1990). *Feedback Control Theory*. Macmillan Publishing Company.

[46] Drummond, T. and Cipolla, R. (1999). Visual Tracking and Control Using Lie Algebras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[47] Drummond, T. and Cipolla, R. (2002). Real-Time Visual Tracking of Complex Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):932–946.

[48] Dunn, E., van den Berg, J. P., and Frahm, J.-M. (2009). Developing visual sensing strategies through next best view planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[49] Durrant-Whyte, H. (1988). Uncertain geometry in robotics. *IEEE Transactions on Robotics and Automation*, 4(1):23–31.

[50] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.

[51] Eade, E. and Drummond, T. (2007). Monocular SLAM as a graph of coalesced observations. In *IEEE International Conference on Computer Vision (ICCV)*.

[52] Eisenbeiss, H. (2004). A mini unmanned aerial vehicle (UAV): System overview and image acquisition. In *ISPRS International Workshop on Processing and Visualization using High-Resolution Imagery*.

[53] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[54] Fard, M. G., Pena-Mora, F., and Savarese, S. (2011). Monitoring changes of 3D building elements from unordered photo collections. In *IEEE International Conference on Computer Vision (ICCV) Workshops*.

[55] Farr, T. G., Rosen, P. A., Caro, E., Crippen, R., Duren, R., Hensley, S., Kobrick, M., Paller, M., Rodriguez, E., Roth, L., et al. (2007). The shuttle radar topography mission. *Reviews of Geophysics*, 45(2).

[56] Faugeras, O. D. and Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508.

[57] Ferguson, D. and Stentz, A. (2006). Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics*, 23(2):79–101.

[58] Fischler, M. A. and Bolles, R. C. (1981). RANdom SAmple Consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[59] Frahm, J.-M., Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building rome on a cloudless day. In *European Conference on Computer Vision (ECCV)*.

[60] Fraser, C. S. (1984). Network design considerations for non-topographic photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 50:1115–1125.

[61] Fraundorfer, F., Lionel, H., Honegger, D., Lee, G., Meier, L., Tanskanen, P., and Pollefeys, M. (2012). Vision-based autonomous mapping and exploration using a quadrotor MAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[62] Früh, C. and Zakhor, A. (2003). Constructing 3D city models by merging ground-based and airborne views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[63] Fukunaga, K. and Narendra, P. M. (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, C-24(7):750–753.

[64] Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Towards internet-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[65] Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376.

[66] Gallup, D., Frahm, J.-M., and Pollefeys, M. (2010). A heightmap model for efficient 3D reconstruction from street-level video. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

[67] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *ACM SIGGRAPH*.

[68] Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221.

[69] Gösele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *IEEE International Conference on Computer Vision (ICCV)*.

[70] Graber, G., Pock, T., and Bischof, H. (2011). Online 3d reconstruction using convex optimization. In *IEEE International Conference on Computer Vision (ICCV) Workshops*.

[71] Gruber, L., Gauglitz, S., Ventura, J., Zollmann, S., Huber, M., Schlegel, M., Klinker, G., Schmalstieg, D., and Höllerer, T. (2010). The city of sights: Design, construction, and measurement of an augmented reality stage set. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[72] Guizzo, E. (2011). Robots with their heads in the clouds. *IEEE Spectrum*, 48(3):16–18.

[73] Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.-M., Hirzinger, G., and Rus, D. (2007). Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[74] Haala, N. and Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580.

[75] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics (JFR)*.

[76] Haralick, R. M., Lee, C., Ottenberg, K., and Nölle, M. (1991). Analysis and solutions of the three point perspective pose estimation problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[77] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*.

[78] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[79] Hartley, R. I. and Sturm, P. F. (1997). Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157.

[80] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition.

[81] He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[82] Helgason, S. (1999). *The Radon Transform*, volume 5. Birkhauser.

[83] Hilton, A., Stoddart, A. J., Illingworth, J., and Windeatt, T. (1996). Reliable surface reconstruction from multiple range images. In *European Conference on Computer Vision (ECCV)*.

[84] Hinterstoisser, S., Benhimane, S., and Navab, N. (2007). N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation. In *IEEE International Conference on Computer Vision (ICCV)*.

[85] Hirschmüller, H. (2006). Stereo vision in structured environments by consistent semi-global matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[86] Hofer, M., Wendel, A., and Bischof, H. (2013). Line-based 3D reconstruction of wiry objects. In *Computer Vision Winter Workshop (CVWW)*.

[87] Hoppe, C., Klopschitz, M., Rumpler, M., Wendel, A., Kluckner, S., Bischof, H., and Reitmayr, G. (2012a). Online feedback for structure-from-motion image acquisition. In *British Machine Vision Conference (BMVC)*.

[88] Hoppe, C., Wendel, A., Zollmann, S., Pirker, K., Irschara, A., Bischof, H., and Kluckner, S. (2012b). Photogrammetric camera network design for micro aerial vehicles. In *Computer Vision Winter Workshop (CVWW)*.

[89] Horn, M. and Dourdoumas, N. (2004). *Regelungstechnik*. Pearson Studium.

[90] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

[91] Hrabar, S. and Sukhatme, G. (2009). Vision–based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431–452.

[92] Hrabar, S., Sukhatme, G. S., Corke, P., Usher, K., and Roberts, J. (2005). Combined optic-flow and stereo-based navigation of urban canyons for a UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[93] Huang, Y. and Gupta, K. (2004). A delaunay triangulation based node connection strategy for probabilistic roadmap planners. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[94] Huber, P. J. (1964). Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics*, 35(1):73–101.

[95] Huber, P. J. (1981). *Robust Statistics*. John Wiley and Sons Inc.

[96] Huttenlocher, D., Klanderman, G., and Rucklidge, W. (1993). Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(9):850–863.

[97] Hyunwoo, K. and Sukhan, L. (2010). A novel line matching method based on intersection context. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[98] Irschara, A., Kaufmann, V., Klopschitz, M., Bischof, H., and Leberl, F. (2010). Towards fully automatic photogrammetric reconstruction using digital images taken from UAVs. In *Proceedings of the ISPRS Symposium, 100 Years ISPRS - Advancing Remote Sensing Science*.

[99] Irschara, A., Rumpler, M., Meixner, P., Pock, T., and Bischof, H. (2012). Efficient and globally optimal multi view dense matching for aerial images. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*.

[100] Irschara, A., Zach, C., and Bischof, H. (2007). Towards wiki-based dense city modeling. In *Workshop on Virtual Representations and Modeling of Large-scale environments (VRML)*.

[101] Irschara, A., Zach, C., Frahm, J. M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[102] Jain, A., Kurz, C., Thormaehlen, T., and Seidel, H. (2010). Exploiting global connectivity constraints for reconstruction of 3d line segments from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[103] Kaminsky, R. S., Snavely, N., Seitz, S. M., and Szeliski, R. (2009). Alignment of 3D point clouds to overhead images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[104] Kaplan, E. D. and Hegarty, C. J. (2006). *Understanding GPS: Principles and Applications*. Artech House Publishers.

[105] Kavraki, L. and Latombe, J. C. (1998). *Probabilistic Roadmaps for Robot Path Planning*. John Wiley and Sons Ltd.

[106] Kavraki, L., Svestka, P., Latombe, J. C., and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[107] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*.

[108] Kemp, C. (2006). *Visual Control of a Miniature Quad-Rotor Helicopter*. PhD thesis, Churchill College, University of Cambridge.

[109] Kempter, T., Wendel, A., and Bischof, H. (2012). Online model-based multi-scale pose estimation. In *Computer Vision Winter Workshop (CVWW)*.

[110] Khaleghi, B., Baklouti, M., and Karray, F. (2009). SILT: Scale-invariant line transform. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*.

[111] Kim, J., Kang, M.-S., and Park, S. (2010). Accurate Modeling and Robust Hovering Control for a Quad-rotor VTOL Aircraft. *Journal of Intelligent and Robotic Systems*, 57(1-4):9–26.

[112] Klein, G. and Drummond, T. (2003). Robust Visual Tracking for Non-Instrumented Augmented Reality. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[113] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[114] Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision (ECCV)*.

[115] Klein, G. and Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[116] Klopschitz, M., Irschara, A., Reitmayr, G., and Schmalstieg, D. (2010). Robust incremental structure from motion. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

[117] Kluckner, S., Birchbauer, J. A., Windisch, C., Hoppe, C., Irschara, A., Wendel, A., Zollmann, S., Reitmayr, G., and Bischof, H. (2011). Construction site monitoring from highly-overlapping MAV images. In *IEEE International Conference on Advanced Video- and Signal-based Surveillance (AVSS)*.

[118] Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[119] Koenig, S. and Likhachev, M. (2002). Improved fast replanning for robot navigation in unknown terrain. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[120] Konolige, K. and Bowman, J. (2009). Towards lifelong visual maps. In *International Conference on Intelligent Robots and Systems (IROS)*.

[121] Kuffner, J. J. and LaValle, S. M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[122] Kushleyev, A., Mellinger, D., and Kumar, V. (2012). Towards a swarm of agile micro quadrotors. In *Robotics: Science and Systems (RSS)*.

[123] Kyrki, V. and Kragic, D. (2011). Tracking Rigid Objects Using Integration of Model-Based and Model-Free Cues. *Machine Vision and Applications*, 22(1):323–335.

[124] Labatut, P., Pons, J., and Keriven, R. (2007). Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision (ICCV)*.

[125] Labrosse, F. (2006). The visual compass: Performance and limitations of an appearance-based method. *Journal of Field Robotics*, 23(10):913–941.

[126] LaValle, S. M. (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Technical report, Carnegie Mellon University.

[127] Leberl, F., Irschara, A., Pock, T., Meixner, P., Gruber, M., Scholz, S., and Wiechert, A. (2010). Point clouds: Lidar versus 3-D vision. *Photogrammetric Engineering and Remote Sensing*, 76(10):1123–1134.

[128] Leberl, F., Meixner, P., Wendel, A., and Irschara, A. (2012). Automated photogrammetry for three-dimensional models of urban spaces. *Optical Engineering*, 51(2).

[129] Lee, D., Merrell, P., Wei, Z., and Nelson, B. (2010). Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance. *Machine Vision and Applications*, 21(3):229–240.

[130] Lee, W., Kim, K., and Woo, W. (2009). Mobile phone-based 3d modeling framework for instant interaction. In *IEEE International Conference on Computer Vision (ICCV) Workshops*.

[131] Leonard, J. J. and Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382.

[132] Li, Y., Snavely, N., and Huttenlocher, D. P. (2010). Location recognition using prioritized feature matching. In *European Conference on Computer Vision (ECCV)*.

[133] Lim, H., Sinha, S. N., Cohen, M. F., and Uyttendaele, M. (2012). Real-time image-based 6-dof localization in large-scale environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[134] Low, K. (2004). *Linear least-squares optimization for point-to-plane ICP surface registration*. Technical report, TR04-004, University of North Carolina.

[135] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110.

[136] Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349.

[137] Lupashin, S., Schoellig, A., Sherback, M., and D'Andrea, R. (2010). A simple learning strategy for high-speed quadrocopter multi-flips. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[138] Matthews, L., Ishikawa, T., and Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):810–815.

[139] Maurer, M., Rumpler, M., Wendel, A., Hoppe, C., Irschara, A., and Bischof, H. (2012a). Geo-referenced 3D reconstruction: Fusing public geographic data and aerial imagery. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[140] Maurer, M., Wendel, A., and Bischof, H. (2012b). Airborne inspection using single-camera interleaved imagery. In *Computer Vision Winter Workshop (CVWW)*.

[141] Meagher, D. (1982). Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147.

[142] Mei, C., Sibley, G., Cummins, M., Newman, P., and Reid, I. (2009). A constant–time efficient stereo SLAM system. In *British Machine Vision Conference (BMVC)*.

[143] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[144] Michels, J., Saxena, A., and Ng, A. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *International Conference on Machine Learning (ICML)*.

[145] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *International Journal of Computer Vision (IJCV)*, 65:43–72.

[146] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI National Conference on Artificial Intelligence*.

[147] Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[148] Moreno-Noguer, F., Lepetit, V., and Fua, P. (2008). Pose Priors for Simultaneously Solving Alignment and Correspondence. In *European Conference on Computer Vision (ECCV)*.

[149] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). 3D reconstruction of complex structures with bundle adjustment: An incremental approach. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[150] Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP)*.

[151] Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[152] Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., and Fitzgibbon, A. (2011a). KinectFusion: Real-time dense surface mapping and tracking. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[153] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011b). DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision (ICCV)*.

[154] Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–770.

[155] Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual Odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[156] Nistér, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[157] Olague, G. (2002). Automated photogrammetric network design using genetic algorithms. *Photogrammetric Engineering and Remote Sensing*, 68(5):423–432.

[158] Pan, Q., Reitmayr, G., and Drummond, T. (2009). ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *British Machine Vision Conference (BMVC)*.

[159] Pix4D Aerial Image Processing Software (2013). `http://www.pix4d.com`.

[160] Pollefeys, M., Nister, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision (IJCV)*, 78(2-3):143–167.

[161] Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *International Journal of Computer Vision (IJCV)*, 59(3):207–232.

[162] Pomerleau, D. (1989). Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document.

[163] Przybilla, H.-J. and Wester-Ebbinghaus, W. (1979). Bildflug mit ferngelenktem Kleinflugzeug. *Bildmessung und Luftbildwesen, Zeitschrift für Photogrammetrie und Fernerkundung*, 47(5):137–142.

[164] Raguram, R. and Frahm, J.-M. (2011). RECON: Scale-adaptive robust estimation via residual consensus. In *IEEE International Conference on Computer Vision (ICCV)*.

[165] Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256.

[166] Ratliff, N., Bradley, D., Bagnell, J. A., and Chestnutt, J. (2006). Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems (NIPS)*.

[167] Reitinger, B., Zach, C., and Schmalstieg, D. (2007). Augmented reality scouting for interactive 3d reconstruction. In *IEEE Virtual Reality (VR)*.

[168] Reitmayr, G. and Drummond, T. (2006). Going Out: Robust Model-Based Tracking for Outdoor Augmented Reality. In *International Symposium on Mixed and Augmented Reality (ISMAR)*.

[169] Rhemann, C., Hosni, A., Bleyer, M., Rother, C., and Gelautz, M. (2011). Fast cost-volume filtering for visual correspondence and beyond. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[170] Ross, S. and Bagnell, J. (2010). Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[171] Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[172] Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., and Hebert, M. (2013). Learning monocular reactive UAV control in cluttered natural environments. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[173] Rosten, E. and Drummond, T. (2006). Machine learning for high–speed corner detection. In *European Conference on Computer Vision (ECCV)*.

[174] Rumpler, M., Irschara, A., Wendel, A., and Bischof, H. (2012). Rapid 3D city model approximation from publicly available geographic data sources and georeferenced aerial images. In *Computer Vision Winter Workshop (CVWW)*.

[175] Rumpler, M., Wendel, A., and Bischof, H. (2013). Probabilistic Range Image Integration for DSM and True-Orthophoto Generation. In *Scandinavian Conference on Image Analysis (SCIA)*.

[176] Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *IEEE International Conference on Computer Vision (ICCV)*.

[177] Sattler, T., Weyand, T., Leibe, B., and Kobbelt, L. (2012). Image retrieval for image-based localization revisited. In *British Machine Vision Conference (BMVC)*.

[178] Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242.

[179] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42.

[180] Scherer, S., Singh, S., Chamberlain, L., and Saripalli, S. (2007). Flying fast and low among obstacles. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[181] Schindler, G., Brown, M., and Szeliski, R. (2007). City-scale location recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[182] Schindler, G., Krishnamurthy, P., and Dellaert, F. (2006). Line-based structure from motion for urban environments. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

[183] Schmid, K., Hirschmüller, H., Doemel, A., Grixa, I., Suppa, M., and Hirzinger, G. (2012). View planning for multi-view stereo 3d reconstruction using an autonomous multicopter. *Journal of Intelligent Robotic Systems*, 65:309–323.

[184] Shuster, M. D. (1993). A Survey of Attitude Representations. *Journal of the Astronautical Sciences*, 41(4):439–517.

[185] Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision (ICCV)*.

[186] Snavely, N., Seitz, S., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. In *ACM SIGGRAPH*.

[187] Snavely, N., Seitz, S. M., and Szeliski, R. S. (2008a). Modeling the world from internet photo collections. *International Journal of Computer Vision (IJCV)*, 80(2):189–210.

[188] Snavely, N., Seitz, S. M., and Szeliski, R. S. (2008b). Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[189] Sontag, E. D. (1998). *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer Verlag, 2nd edition.

[190] Srinivasan, M. V. (2011). Visual control of navigation in insects and its relevance for robotics. *Current Opinion in Neurobiology*, 21(4):535–543.

[191] Stentz, A. (1995). The focussed D* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[192] Stentz, A. and Hebert, M. (1995). A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2):127–145.

[193] Strasdat, H., Montiel, J., and Davison, A. (2010a). Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems (RSS)*.

[194] Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010b). Real-time monocular SLAM: Why filter? In *IEEE International Conference on Robotics and Vision (ICRA)*.

[195] Strecha, C., Pylvaenaeinen, T., and Fua, P. (2010). Dynamic and scalable large scale image reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[196] Strecha, C., von Hansen, W., Van Gool, L., Fua, P., and Thoennessen, U. (2008). On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[197] Stühmer, J., Gumhold, S., and Cremers, D. (2010). Real-time dense geometry from a handheld camera. In *German Conference on Pattern Recognition (DAGM)*.

[198] Szeliski, R. (2006). Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2(1):1–104.

[199] Thorpe, C., Hebert, M. H., Kanade, T., and Shafer, S. A. (1988). Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 10(3):362–373.

[200] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*, volume 1. MIT Press.

[201] Thrun, S., Fox, D., and Burgard, W. (1998). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1):29–53.

[202] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692.

[203] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment – A modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–375. Springer Verlag.

[204] Trummer, M., Munkelt, C., and Denzler, J. (2010). Online next-best-view planning for accuracy optimization using an extended e-criterion. In *International Conference on Pattern Recognition (ICPR)*.

[205] Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. In *ACM SIGGRAPH*.

[206] Uden, M. and Zipf, A. (2013). Open building models: Towards a platform for crowdsourcing virtual 3d cities. In *Progress and New Trends in 3D Geoinformation Sciences*, Lecture Notes in Geoinformation and Cartography, pages 299–314. Springer Verlag.

[207] Unger, C., Wahl, E., Sturm, P., and Ilic, S. (2010). Probabilistic disparity fusion for real-time motion-stereo. In *Asian Conference on Computer Vision (ACCV)*.

[208] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y., Singh, S., Snider, J., Stentz, A., Whittaker, W., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466.

[209] Valgren, C. and Lilienthal, A. J. (2010). SIFT, SURF & Seasons: Appearance-based long-term localization in outdoor environments. *Robotics and Autonomous Systems*, 58(2):149–156.

[210] Vandapel, N., Kuffner, J., and Amidi, O. (2005). Planning 3D path networks in unstructured environments. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[211] Varadarajan, V. S. (1984). *Lie Groups, Lie Algebras and Their Representations*. Springer Verlag.

[212] Vázquez, P.-P., Feixas, M., Sbert, M., and Heidrich, W. (2001). Viewpoint selection using viewpoint entropy. In *Vision Modeling and Visualization Conference*.

[213] Vogiatzis, G., Torr, P., and Cipolla, R. (2005). Multi-view stereo via volumetric graph-cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[214] Von Gioi, R., Jakubowicz, J., Morel, J.-M., and Randall, G. (2010). LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(4):722–732.

[215] Wagner, D. and Schmalstieg, D. (2007). ARToolKitPlus for Pose Tracking on Mobile Devices. In *Computer Vision Winter Workshop (CVWW)*.

[216] Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., and van de Molengraft, R. (2011). RoboEarth. *IEEE Robotics & Automation Magazine*, 18(2):69–82.

[217] Watanabe, K., Yoshihata, Y., Iwatani, Y., and Hashimoto, K. (2008). Image-Based Visual PID Control of a Micro Helicopter Using a Stationary Camera. *Advanced Robotics*, 22(2-3):381–393.

[218] Weiss, S., Achtelik, M., Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). Intuitive 3D maps for MAV terrain exploration and obstacle avoidance. 61(1-4):473–493.

[219] Wendel, A., Hoppe, C., Bischof, H., and Leberl, F. (2012a). Automatic fusion of partial reconstructions. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*.

[220] Wendel, A., Irschara, A., and Bischof, H. (2011a). Automatic alignment of 3D reconstructions using a digital surface model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[221] Wendel, A., Irschara, A., and Bischof, H. (2011b). Natural landmark-based monocular localization for MAVs. In *IEEE International Conference on Robotics and Vision (ICRA)*.

[222] Wendel, A., Maurer, M., and Bischof, H. (2012b). Visual landmark-based localization for MAVs using incremental feature updates. In *Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT)*.

[223] Wendel, A., Maurer, M., Graber, G., Pock, T., and Bischof, H. (2012c). Dense Reconstruction On-the-Fly. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[224] Wendel, A., Maurer, M., Katusic, M., and Bischof, H. (2012d). Fuzzy visual servoing for micro aerial vehicles. In *Austrian Robotics Workshop (ARW)*.

[225] Weng, K. W. and Abidin, M. S. B. (2006). Design and Control of a Quad-Rotor Flying Robot For Aerial Surveillance. In *Proceedings 4th Student Conference on Research and Development*.

[226] Wenhardt, S., Deutsch, B., Angelopoulou, E., and Niemann, H. (2007). Active visual object reconstruction using D-, E-, and T-optimal next best views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[227] Werlberger, M., Pock, T., and Bischof, H. (2010). Motion estimation with non-local total variation regularization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[228] Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., and Bischof, H. (2009). Anisotropic Huber-L1 optical flow. In *British Machine Vision Conference (BMVC)*.

[229] Wheeler, M., Sato, Y., and Ikeuchi, K. (1998). Consensus surfaces for modeling 3d objects from multiple range images. In *IEEE International Conference on Computer Vision (ICCV)*.

[230] Wilhelms, J. and Gelder, A. V. (1992). Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227.

[231] Wu, B., Ooi, T., and He, Z. (2004). Perceiving distance accurately by a directional process of integrating ground information. *Nature*, 428(6978):73–77.

[232] Wu, C. (2007). SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/∼ccwu/siftgpu.

[233] Yun, B., Peng, K., and Chen, B. M. (2007). Enhancement of GPS signals for automatic control of a UAV helicopter system. In *International Conference on Control and Automation (ICCA)*.

[234] Zach, C. (2008). Fast and high quality fusion of depth maps. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

[235] Zach, C., Pock, T., and Bischof, H. (2007). A globally optimal algorithm for robust TV-L1 range image integration. In *IEEE International Conference on Computer Vision (ICCV)*.

[236] Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8(3):338–353.

[237] Zamir, A. R. and Shah, M. (2010). Accurate image localization based on google maps street view. In *European Conference on Computer Vision (ECCV)*.

[238] Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision (IJCV)*, 13(2):119–152.

[239] Zhang, Z. (2000). A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(11):1330–1334.

[240] Zhao, W., Nister, D., and Hsu, S. (2005). Alignment of continuous video onto 3D point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(8):1305–1318.

[241] Zhiheng, W., Fuchao, W., and Zhanyi, H. (2009). MSLD: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953.

[242] Zhu, Z. W., Oskiper, T., Samarasekera, S., Kumar, R., and Sawhney, H. S. (2008). Real-time global localization with a pre-built visual landmark database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[243] Zufferey, J.-C., Beyeler, A., and Floreano, D. (2010). Autonomous flight at low altitude with vision-based collision avoidance and GPS-based path following. In *IEEE International Conference on Robotics and Vision (ICRA)*.