Graz University of Technology

Institute for Computer Graphics and Vision

Master's Thesis

Active Monocular Localization: Towards Autonomous Monocular Exploration for Quadrotor MAVs

Christian Mostegel

Graz, Austria, August 2013

Thesis supervisors

Univ.-Prof. Dipl.-Ing. Dr. techn. Horst Bischof

Dipl.-Ing. Dr. techn. Andreas Wendel

Doubt grows with knowledge.

Johann Wolfgang von Goethe

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

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

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

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

In this thesis we approach the topic of monocular SLAM from an active perspective. The main contribution of this work is bridging the gap between passive monocular SLAM and autonomous robotic systems. While passive monocular SLAM strives to reconstruct the scene and determine the current camera pose for any given camera motion, not every camera motion is equally suited for these tasks. Thus, we propose three novel measures which allow the analysis of possible camera motions with respect to physical and visual constraints. The first measure is called ”localization quality” and allows the evaluation of the stability of the monocular localization at arbitrary virtual camera poses. This generic measure can be used with every monocular SLAM approach which is based on bundle adjustment. The purpose of the second measure, the ”point generation likelihood”, is to evaluate the chance of generating new map points from arbitrary virtual view points. It is based on our novel variable depth distribution (VDD), which provides a reasonable guess for the 3D position of yet unmapped 2D features without any prior knowledge of the scene. The third of our proposed measures allows for an evaluation of the navigational safety of holonomic aerial vehicles. We use these novel measures in a destination-based planning approach for multirotor MAVs which makes the resulting system capable of autonomous explorative navigation. In our experiments we demonstrate the effectiveness of our novel measures as well as the capabilities of the overall system. We achieve autonomous waypoint navigation with a quadrotor MAV in challenging indoor environments. Furthermore, we demonstrate that even tasks like a full 360° turn in sparsely textured environments can be achieved through our explorative navigation approach. In all experiments our system was able to maintain the visual localization at all times.

Keywords. UAV, quadrotor MAV, quadcopter, autonomous explorative navigation, collision avoidance, localization quality, monocular localization, active monocular localization, active SLAM, active visual localization

Kurzfassung

In dieser Arbeit behandeln wir das Thema monokulare "Simultane Lokalisierung und Kartenerstellung" (SLAM) von einer aktiven Perspektive. Der wichtigste Beitrag dieser Arbeit ist die Schließung der Lücke zwischen passiven monokularem SLAM und autonomen Robotik-Systemen. Während passives monokulares SLAM versucht für jede gegebene Kamerabewegung die Szene zu rekonstruieren und die aktuelle Kamerapose zu bestimmen, ist nicht jede Kamerabewegung gleichermaßen für diese Aufgaben geeignet. Um mögliche Kamerabewegungen hinsichtlich dieser Aspekte zu analysieren wurden drei neue Maße entwickelt. Das erste Maß nennt sich "localization quality" und erlaubt die Evaluierung der Stabilität der monokularen Lokalisierung für beliebige virtuelle Kameraposen. Die Aufgabe des zweiten Maßes, der "point generation likelihood", ist es abzuschätzen wie hoch die Chancen sind von einem beliebigen virtuellen Blickwinkel aus neue 3D-Punkte zu generieren. Dieses Maß basiert auf der neuentwickelten "variable depth distribution" (VDD), welche die Bestimmung der Tiefe von unkartierten 2D-Punkten auf probabilistische Weise ermöglicht. Mit unserem dritten Maß ist es möglich die Navigations-Sicherheit von holonomen Fluggeräten zu bestimmen. In dieser Arbeit vereinen wir diese drei neuen Maße in einem Zielpunkt-basiertem Planungsansatz für Multirotorflugzeuge. In unseren Experimenten zeigen wir die Effektivität unserer Maße so wie die Fähigkeiten des gesamten Systems mit einem Quadrotorflugzeug. Selbst in herausfordernden unbekannten Umgebungen ist das resultierende System fähig autonom und sicher zwischen Wegpunkten zu navigieren. Sogar ein volle 360° Drehung, welche eine der schwierigsten Aufgaben für monokulare Systeme darstellt, kann dank der erforschenden Natur unseres Ansatzes gemeistert werden. In all unseren Experimenten konnte die visuelle Lokalisierung stets aufrecht erhalten und jegliche Kollisionen vermieden werden.

Schlagwörter. Multirotorflugzeug, autonome erforschende Navigation, Kollisionsvermeidung, aktive monokulare Lokalisierung, aktive visuelle Lokalisierung

Acknowledgments

The successful completion of this thesis would not have been possible without the constructive contribution and support of many people.

First of all, I would like to thank my supervisors, Univ.-Prof. Dipl.-Ing. Dr. techn. Horst Bischof and Dipl.-Ing. Dr. techn. Andreas Wendel, for their thoughtful guidance and helpful suggestions.

Secondly, it is a pleasure to thank the Institute of Electronic Music and Acoustics of the University of Music and Performing Arts Graz, especially Dipl.-Ing. Matthias Frank, for granting us free access to their tracking laboratory which allowed for an accurate evaluation of our approach.

Thirdly, I would like to express my gratitude to my fellow student Gert Hutter who was always willing to lend me a hand for my experiments and readily shared his insight in the field of collision avoidance.

Finally, I am greatly indebted to my family and friends for their support. I owe my deepest gratitude to my loving wife, Theodora, who selflessly assisted me on every step of the way to this thesis. Thank you!

This thesis was financially supported by the Institute for Computer Graphics and Vision and the Faculty of Computer Science.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Monocular reconstruction is a field of growing interest. It allows 3D reconstruction of objects and environments at minimal cost only using a single passive camera. In order to achieve a complete reconstruction it is often necessary to acquire images from unconventional viewpoints, e.g. a bird's eye view. Images from these special viewpoints can be very cheaply acquired using Micro Aerial Vehicles (MAVs). To achieve this acquisition autonomously it is necessary that the system has knowledge of its own whereabouts (localization) and the capability to move to the desired location (navigation).

For localization most current systems rely on GPS, but in urban surroundings the signal quality is often too low to achieve an accurate localization. Other sensors, such as laser scanner or RGBD-cameras, have a very limited depth perception and increase payload of the MAV, which results in a lower mission time. Recent advances in the augmented reality (AR) community [12, 37, 54, 71] offer a different path. These works demonstrate that an accurate localization can also be achieved without any additional senors using only a single passive camera. This approach is called monocular SLAM (Simultaneous Localization And Mapping).

Unfortunately, the quality of the monocular localization strongly depends on the availability of natural landmarks, which can be tracked across a series of images. In general, it is easy to find such landmarks in regions with rich texture and high contrast, whereas a lack of texture and low contrast makes it difficult to retrieve suitable landmarks. Furthermore, it is necessary to perceive these landmarks from different viewpoints to obtain an accurate estimate of the 3D position of the landmarks. As the AR community relies on the user for the camera motion, current monocular SLAM approaches simply try make the best of the current situation and fail when the user moves the camera in a wrong way, e.g. only rotation without translation. For a human user this does not pose a severe problem, as he/she still knows where the system last worked and can backtrack the camera motion to this point. In contrast, a loss of the visual localization is a major problem for an autonomous system which has no other means to localize itself. This is especially true for unstable systems like airborne MAVs.

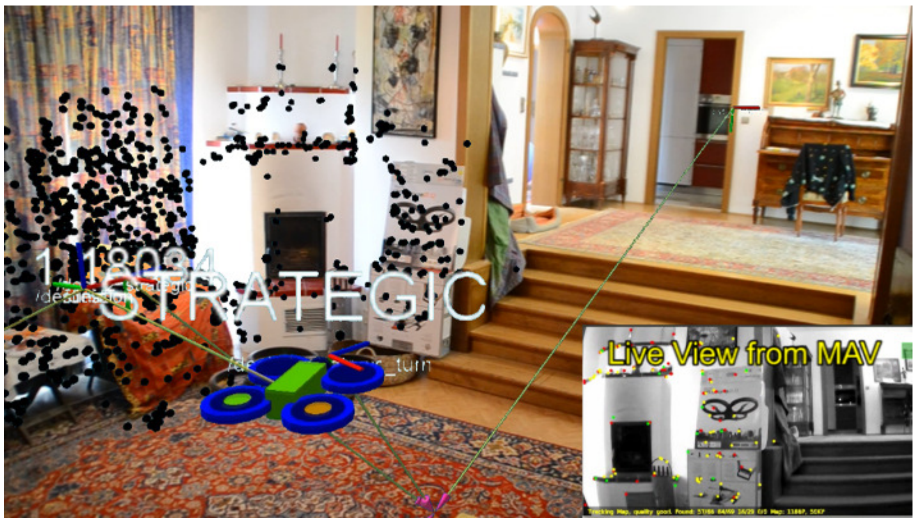


Figure 1.1: Active Visual Localization. Through the analysis of the sparse reconstruction (black dots) and actively exploring relevant parts of the scene, our system maintains the visual localization at all times during the way-point navigation.

The aim of this work is to provide means to avoid a localization loss through *active visual localization*. This means that instead of assuming that the camera moves in the correct way, we actively control the camera motion in a way that does not endanger the visual localization.

For this purpose we propose a novel measure, which we call ”localization quality”. This measure evaluates the localization stability and can be used to predict a localization loss. As this measure only depends on the geometry of the reconstruction (map), it can be calculated for arbitrary viewpoints which makes it possible to respect the need for visual localization even in the planning phase.

In this work we do not assume that the environment is known at start-up, but instead treat every flight as an exploration mission. This makes our approach very adaptive to new environments, but requires the generation of new map points for localization. To this end, we developed a way of estimating the probability distribution of potential map points in unmapped regions of the scene. We use this discrete probability distribution to estimate the "point generation likelihood" for an arbitrary viewpoint. Hence, this novel measure provides the means for an intelligent generation of new map points.

As an active component, we consider holonomic aerial vehicles in this work. Since we are aiming for an autonomous navigation system, it is necessary to avoid obstacles and unknown parts of the scene. In order to achieve navigational safety we build a probabilistic representation of the environment and use it to estimate the collision probability of the MAV.

The final contribution of this thesis is a complete active system which is capable of autonomous explorative navigation using a destination-based planning scheme. The system can autonomously detect and avoid states with a critical localization stability and generate new map points whenever necessary.

In the following chapter we discuss research topics which are related to this work. Chapter 3 outlines our novel measures and Chapter 4 explains how to combine them to an autonomous explorative navigation system. In Chapter 5 we demonstrate the effectiveness of our measures and the capabilities of our explorative navigation approach in a series of experiments. Finally, we conclude this thesis in Chapter 6 with a summary of our contributions and an outlook for future work.

Chapter 2

Related Work

Contents

This work is not limited to a single field of research, but can be regarded as a bridge between the two very active fields of monocular SLAM and autonomous robots. The first field, monocular SLAM (Simultaneous Localization And Mapping), is concerned with the reconstruction of the environment using only a single passive camera and the localization of the current camera pose in relation to the reconstructed scene. The second field, autonomous robots, is a very versatile and interdisciplinary field of research. Among other topics, it is concerned with the physical construction of robotic systems, the controlling of their actuators, ways of perceiving the environment as well as the search for the optimal plan for a specific task and its execution, i.e. navigation. Of all these topics, this work is only concerned with the last two; perception and navigation.

As this work creates the bridge between many different research topics, we outline the most common approaches of each field and relate them to our work. We start with the topic of robotic perception in Section 2.1. After discussing different perception concepts, we provide a review of current visual SLAM approaches. In Section 2.2 we use the Parallel Tracking And Mapping (PTAM) approach of Klein and Murray [37], which we used in the experimental evaluation of our work, to introduce the reader to relevant aspects of feature detection and structure from motion (SfM). After having outlined the perception model of our system, we discuss the topic of reconstruction uncertainty and relate it to current approaches of view planning in Section 2.3. The last remaining topic related to our work is navigation for multirotor Micro Aerial Vehicles (MAVs), which we discuss in Section 2.4. This topic is not only limited to the execution of motion commands, but also tackles collision avoidance and path planning.

2.1 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is a very active field of research and grants a system (passive or active) the notion of location awareness in an unknown environment. It consists of two tasks, namely localization and mapping, which are strongly interwoven. On the one hand, the system wants to construct a map using the current sensor readings and the information of the robot's current location, whereas on the other hand, the system wants to estimate the robot's current location using the current sensor readings and the already constructed map. As neither the map can be constructed without localization nor localization can be achieved without a map, the two tasks are solved simultaneously.

Pioneer work in this field of research was done by Smith et al. [66, 67] as well as Durrant-Whyte et al. [15, 16, 42]. These works formulate SLAM as a probabilistic problem based on Gaussian noise models for the robot motion as well as the sensor readings. Through this formulation the solution for both tasks, localization and mapping, can be found by maximizing the likelihood of the solution. This maximum likelihood solution can be found through Monte Carlo simulation and particle filtering as described in [51, 74]. This kind of probabilistic formulation is very general and can be adapted to all kind of range sensors.

Consequently, a range of different sensors for robotic perception have been proposed over the years. Early works [8, 41] used ultra-sonic sensors, i.e. high-frequency sonars, to acquire two-dimensional depth estimates around ground robots. Nowadays, these sensors are still in use, but mostly for one-dimensional problems such as height measurement. In general, range measurements of sonars contain a high amount of noise and have a very limited range, i.e. a few meters.

Currently very popular sensors are laser range finders, also known as laser scanners, which are a special type of lidar[1]. A lidar sensor illuminates the scene with a laser and

[1]According to the Oxford University Press [55] the word "lidar" resulted from the blending of the words "light" and "radar".

analyzes the reflected light in order to measure distances. In laser scanners the laser is reflected by a rotating mirror which results in range measurements along the plane of rotation of the mirror. The accuracy and range of such a sensor strongly depends on the power of the laser. As aerial vehicles have only a very limited amount of power available, the range of suitable laser scanners is limited to only a few meters. Compared to sonar sensors the noise of the measurements is drastically lower and the spatial resolution significantly higher. Grzonka et al. [25] as well as He et al. [29] have successfully used laser scanners for MAV localization in indoor environments. The limited range of the sensors and the fact that range readings are only provided on a single plane limits systems which exclusively use laser scanners for sensing to the operation in simple indoor environments.

Another type of lidar, namely the Time-of-Flight (ToF) camera, increases the depth perception from a single plane to a three dimensional cone. These sensors use a regular image grid similar to passive RGB-cameras, but actively determine the depth value for each pixel. Other RGBD-camera systems, like the Microsoft Kinect[2] or Asus Xtion[3], use infrared projectors for active depth measurement. Such depth-sensing camera systems have shown promising results for MAV navigation and exploration in indoor-environments [2, 52, 63]. Unfortunately, ToF- and RGBD-cameras suffer from the same problems as all other active sensors. Through the active projection of light the maximum depth perception is limited by the available power of the projected light, which restricts systems depending on this information to indoor environments.

An alternative to active sensing technologies is passive vision. Passive visual sensors have two main advantages over active ones. First of all, they have a lower power consumption as they use the available light instead of actively projecting light themselves. Secondly, the range of passive visual sensors is only limited by atmospheric influences such as the humidity of the air. This enables high range perception at a low cost, which is perfectly suited for low-power systems like MAVs. In the following paragraph we discuss the topic of visual SLAM in more detail.

Visual SLAM. One way to achieve SLAM with passive sensors is to use a stereo camera pair. In general, such stereo cameras are mounted parallel to each other with a fixed distance between the cameras, i.e. a fixed *baseline*. This camera pair can then be calibrated together and the disparity between the image pairs can be used to estimate the depth of the scene. Schauwecker et al. [61] have shown that it is possible to control an MAV based

[2]http://www.microsoft.com/en-us/kinectforwindows/
[3]http://www.asus.com/Multimedia/Xtion_PRO/

on the readings of a stereo camera and Fraundorfer et al. [23] have even succeeded in building a stereo-based MAV system which is capable of autonomous outdoor exploration. However, the maximum depth perception of the system is limited by the width of the baseline between the cameras and the resolution of the cameras. This property drastically limits the depth perception for micro aerial vehicles, which can only achieve a very limited baseline due to their own small size. If we think of the current trend towards even smaller so called "nano aerial vehicles" [5, 40], the possible baseline of camera pairs is further decreasing.

Recent advances in the Augmented Reality community [12, 37, 54, 71] have opened the door to another way of depth perception which only uses a single camera as exteroceptive sensor; i.e. monocular SLAM. The depth estimation of this approach is achieved in a fashion which is very similar to the stereo depth estimation. Instead of using two cameras at the same time, the monocular approach uses the same camera at different points in time at different positions in the world. Under the assumption of a static environment, this enables the system to adapt the baseline between image pairs to achieve an accurate depth estimate no matter how far away the objects of interest are located. A further advantage of this approach is that a single camera only consumes half the power of a stereo camera pair and also adds only half the payload to the MAV. These properties make a monocular approach very attractive for micro aerial vehicles and even more so for nano aerial vehicles.

There are currently two different ways to approach the topic of monocular SLAM; i.e. filtering [10, 12, 17] and batch optimization [37, 71]. In general, both ways model the uncertainty of the camera pose and the 3D correspondences through Gaussian distributions. The difference is how this uncertainty is used to improve the estimate of the current state (pose and measurements). On the one hand, the standard algorithm for filtering, the EKF (Extended Kalman Filter), retains only the current pose and a vector of features via mean vectors and covariance matrices. On the other hand, the standard algorithm for optimization, BA (Bundle Adjustment), keeps a subset of past poses and a vector of features as well as the interconnections between them. This whole graph is then globally optimized to minimize the reprojection error. As a global optimization technique, BA shows a cubic complexity to the number of keyframes and was therefore not considered for real-time SLAM until the introduction of Parallel Tracking and Mapping (PTAM) by Klein and Murray [37]. In their approach they separate the localization and the mapping part of SLAM and run them in two distinct threads. Thus, it becomes possible to achieve real-time localization, while using global bundle adjustment (GBA) to improve the map in the background. In order to avoid solely relying on a global optimization with cubic complexity they additionally use local bundle adjustment (LBA). Holmes et al. [30] show that LBA can be achieved in constant time by using a relative data representation.

Strasdat et al. [72] have shown that BA-based SLAM outperforms filter-based SLAM in matters of accuracy, whereas they argue that filter-based SLAM might still be the better choice in systems with a very limited computing power. Recent advances in the field of BA-based SLAM [39, 77, 79] soften this minor disadvantage in demonstrating that BA-based monocular SLAM can be achieved on devices with very limited resources, such as a mobile phone or onboard an MAV. This and the significant difference in the reconstruction accuracy motivated us to focus our research on BA-based approaches.

In this work we propose a novel measure which can be used to query the localization quality for an arbitrary viewpoint. For the calculation of this measure we only use information which is available in every BA-based SLAM approach. Consequently, the proposed measure is independent of the actual implementation and does not only work with a single SLAM package. For our experiments we have decided to use Parallel Tracking and Mapping (PTAM) [37] with some adaptations by Weiss et al. [77] and ourselves. We have chosen PTAM as it has proven to be well-suited for the task of monocular MAV controlling as a myriad of works testify [1, 18, 19, 36, 77, 80]. In the next section we use PTAM as an example to introduce the reader to selected topics of structure from motion and visual SLAM which are relevant for our work. Furthermore, we use this context to outline the most important modifications to the original version of Klein and Murray [37].

2.2 Parallel Tracking and Mapping (PTAM)

"Parallel Tracking and Mapping" is a monocular SLAM approach which was originally developed by Klein and Murray [37]. Due to its good performance and its free availability it became very popular and now exists in many different versions developed by a variety of research groups, such as [19, 39, 77, 80]. As a basis for our work we used the version of Weiss et al. [77], which is available as a ready-to-use ROS package[4].

In this section we explain the general characteristics of the PTAM approach and relate the aspects which have a large impact on our work to other relevant literature in the corresponding field. We start the section with a general overview of the approach. Then we provide a detailed survey on visual landmarks, also known as "features" or "keypoints", as they make up the foundation of the localization, the mapping as well as the optimization

[4]http://ros.org/wiki/ethzasl_ptam

scheme. We finish this section with a detailed description of the mapping and localization procedures of the PTAM approach.

2.2.1 Overview

The aim of the approach is to use a constant stream of images (frames) as input and determine from which viewpoint (camera pose) each frame was taken in relation to a known coordinate system. To achieve this it is necessary to concurrently solve the localization as well as the mapping task. This can be seen as a chicken and egg problem. Without a map the system cannot localize itself and without a proper localization it cannot build a map.

To solve this problem the camera pose of the first frame is assumed to be known. From a different perspective it could be interpreted that this frame defines the reference coordinate system. In a next step visual landmarks (features) are detected in the image. In finding the corresponding landmarks in another frame, which was taken from a slightly different viewpoint, the system estimates the relative camera motion and the 3D position of the landmarks. The scale of the reconstruction is arbitrary as it is impossible to determine the effective scale of the scene without prior knowledge. The collection of 3D landmark positions (3D points) is then regarded as ”map”.

Using the map it is now possible to track the known landmarks from frame to frame. In using the 3D information of the landmarks and their 2D location in the current frame, the approach then estimates the camera pose of the current frame in minimizing the reprojection error. This procedure implements the localization task of SLAM and has to be realized in real-time at the frame rate of the camera.

As the camera moves along it is necessary to add new landmarks to the map in order to maintain the localization. Theoretically, it would be possible to search in every frame for new landmarks. The problem with this approach is that successive frames are nearly identical to each other which would lead to an unnecessary computational burden. To overcome this problem PTAM selects only ”informative” frames which are called ”keyframes”. The selection is done in a heuristic fashion and only frames are chosen which are more than a minimum distance away from already added keyframes. To realize the necessary real-time performance of the localization procedure, a newly added keyframe is not handled right away, but instead pushed into a queue of potential keyframes, which is processed in another thread; the mapping thread.

Opposed to the localization thread, the mapping thread has no hard real-time constraints. It is no problem if it lags behind a little bit when some new keyframes are added in a short time, because it only has to keep up with the average rate of newly added keyframes. In general, this rate is far lower than the camera frame rate and depends on the speed of the camera motion.

When a new keyframe is added, the system searches for yet unmapped landmarks which are also visible in another keyframe. Similar to the initialization it is then possible to determine the 3D position of these new landmarks and add them to the map. When there are no new keyframes in the queue of the mapping thread, the system first applies local bundle adjustment and then global bundle adjustment to improve the map. Thus it optimizes the map whenever there is time left to do so. In the following subsection we will discuss the nature of the visual landmarks and how they can be detected and tracked.

2.2.2 Visual Landmarks

Visual landmarks have been used for centuries for nautical localization and navigation in form of stars or light houses. The advances in recent research have widened the spectrum of possible visual landmarks drastically. We will call these "visual landmarks" from here on simply "features" to fit the general nomenclature in current literature. Basically every image structure, i.e. patterns, corners, blobs, regions etc., is well suited to be a feature if it fulfills the following three conditions: Firstly, a good feature should be repeatably detectable from different viewpoints and under varying light conditions. Secondly, the feature should have a precise location in the image and the real world. Finally, the feature should be well distinguishable from other structures in the image. Note that this definition already mixes feature detection, description and matching, as these aspects are not always easy to separate. In the following paragraph we will focus on the feature detection and neglect the third condition, which is mainly part of the feature description and matching.

Feature Detection. Theoretically, it is possible to skip the feature detection step and to directly treat each pixel or sub-pixel and its neighborhood as a feature. This approach easily leads to a vast number of features of which most cannot be unambiguously matched, e.g. all black patches in an image look very similar. This leads to a strong uncertainty in the location of the feature correspondences and a very high computational cost in matching. The main task of the feature detection is to reduce the search space as well as the location uncertainty for feature matching. For this task many types of features have been proposed, such as corners, blobs or regions. All detection procedures, no matter which type of feature they search for, analyze a set of pixels to find some sort of salient image structure. To reduce the location uncertainty, many approaches estimate 2D image coordinates for the salient image structures with sub-pixel accuracy. These 2D image coordinates are also called "keypoints", as they represent a salient feature through a single point. For corners this would correspond to the intersection between the two edges that make up the corner and for blobs it would correspond to the centroid. All feature detectors which output 2D image coordinates for the detected features are consequently called "keypoint detectors". In the following we provide an overview of the most relevant feature detection approaches for the task of localization.

The most widely used keypoint detectors are the SIFT (Scale-Invariant Feature Transform) [45] and the SURF (Speeded Up Robust Features) [3] detectors. The idea of both approaches is very similar and both make use of the Hessian Matrix and thusly the second derivatives. SIFT uses the Difference of Gaussians (DoG) to detect the characteristic scale of the blob in a scale space pyramid based on the theory of Lindeberg [44]. To separate corners and blobs from edges this approach uses the trace and determinate of the Hessian Matrix to assess the curvature of the region. SURF [3], on the other hand, can be seen as a further approximation to SIFT which uses integral images instead of image pyramids. This makes the integer approximation of the Hessian Matrix extremely fast, and the scale can be determined in using different sized masks. Matas et al.[48] proposed a completely different type of region detector. They detect Maximally Stable Extremal Regions (MSERs), which are regions of an arbitrary shape. The regions are obtained by consecutively thresholding a gray-value image and keeping track of the regions. Regions which stay the same over a large interval of thresholds are considered maximally stable. According to [50] the MSER detector achieves the highest score in most cases compared to other affine region detectors. Donoser and Bischof [13] have also proposed an efficient way to track MSERs. The biggest drawback of MSERs is that, to our best knowledge, no efficiently parallelized version exists, which leads to a considerable longer computation time than SIFT or SURF, which have both been ported to the GPU [7, 65].

Another approach to keypoint detection makes use of the first derivatives using the second moment matrix. By analyzing the eigenvalues of this matrix edges as well as corners can be detected, as edges have a large gradient in one direction and corners in more than one direction. Harris corners [27] avoid the expensive calculation of eigenvalues in analyzing the determinant and trace of second moment matrix, which made it more popular than Shi-Tomasi [64] which is based on the eigenvalues.

An approach without derivatives was proposed by Smith and Brady [68] with the SU-

SAN corners (Smallest Uni-Value Segment Assimilating Nucleus Test). This approach compares the brightness values of pixels in a circular region to the center pixel. Pixels of similar brightness form the Uni-Value Segment Assimilating Nucleus (USAN). In subtracting the size of the USAN from a geometric threshold, edges and corners can be detected (for corners the threshold is lowered). The advantages of this approach are that it is more robust to noise than approaches based on derivatives, and that it has no expensive calculations such as eigenvalues. To further speed-up the corner detection for real-time applications Rosten and Drummond have proposed "Features from Accelerated Segment Test"(FAST) [59]. Very similar to SUSAN, FAST compares the center pixel to its neighborhood, but instead of a circular area only a circle of 16 pixels is considered. To simplify the detection criterion even further, a pixel is considered a corner if a connected set of nine pixels (FAST-9/16) is either darker or brighter than a threshold which depends on the brightness of the center pixel. In order to speed up this comparison, a ternary decision tree is learned, to reduce the number of comparisons for a specific scene. To generalize this optimization for arbitrary scenes Mair et al. [47] propose the "Adaptive and Generic Accelerated Segment Test" (AGAST). They suggest to replace the specialized ternary tree with two binary trees. One tree is optimized for structured and the second for homogeneous image regions. Using dynamic programming the tree can easily be switch at the leaves of the tree, if a new region is entered.

Feature Detection in PTAM. The feature detection in PTAM happens in the localization thread which has a hard real-time constraint and has to keep up with the camera frame rate. Thus it is necessary to detect features as fast as possible. Consequently, the original PTAM [37] uses FAST-9/16 for the feature detection. Prior to the detection, PTAM builds an image pyramid with four different scale levels as FAST has no inherent way of scale detection. Then it detects FAST corners on each pyramid level. To reduce the number of responses, PTAM applies non-maximal suppression and thresholding based on the Shi-Tomasi [64] score on each pyramid level.

In our implementation we follow the example of Weiss et al. [77] and use AGAST instead of FAST as Mair et al. [47] report a significantly improved performance compared to FAST at no additional cost.

Feature Description. Feature detection approaches which were designed to establish point correspondences across images with a wide baseline, such as SIFT [45] or SURF [3], calculate a descriptor for each detected feature. These descriptors always try to balance two competing goals. On the one hand, descriptors are required to be very descriptive, as to distinguish similar features from each other. On the other hand, they should be robust against noise, illumination change and viewpoint variation, which lessens the descriptiveness of the descriptor.

The calculation of descriptors which are covariant to viewpoint changes, e.g. SIFT [45] or SURF [3], comes at a significant computational cost. To realize the extraction in real-time the algorithms have been parallelized and ported to the GPU [7, 65]. The calculation of viewpoint covariant descriptors is necessary for wide baseline matching, because they do not make any assumptions about the camera pose or the world.

As we are doing simultaneous "localization" and "mapping", we have very specific assumptions about the viewpoint relation between frames as well as a the location of features in the real world. Therefore, PTAM [37] skips the expensive calculation of a feature descriptor and uses the original gray-scale image instead. For a detected keypoint, PTAM only saves the 2D location in the image and the detection level in the image pyramid and defers the viewpoint and illumination invariance to the matching stage. In the following two subsections we will discuss the topic feature matching in the context of mapping and localization. During mapping the system only knows the spatial relation between frames and has only a limited knowledge of the scene, whereas during localization the system has very detailed knowledge of the scene but only a rough estimate of the current camera pose.

2.2.3 Mapping

In PTAM "mapping" refers to the generation of a sparse reconstruction of the scene. Through the estimation of the spatial relation between frames and known feature correspondences between them, it is possible to estimate the 3D location of the feature points in the real world. The less the system knows about the scene and the relation between the frames, the more effort it takes to extract the necessary information. In this subsection we first discuss the initialization of the map, where the knowledge of the scene and the relation between frames is at its minimum. Then we cross over to the case where the current camera pose is known in relation to the already constructed map because of the localization procedure. Finally, we discuss when and how the constructed map is optimized using bundle adjustment.

2.2.3.1 Initialization

The initialization of a 3D map with a single pair of images, without knowledge about the scene or the relative camera motion between the images, is an ill-posed problem. Even in theory it is not possible to determine the scale of the reconstruction without additional knowledge. In the next paragraph we explain how the problem of map initialization is solved in the original version of PTAM by Klein and Murray [37]. Then we cross over to other PTAM versions which were specifically adapted for the use of micro aerial vehicles and discuss the possibilities of scale estimation.

Map Initialization by Klein and Murray [37]. In the original version of PTAM the map initialization is done with two keyframes. The first keyframe is taken when the user presses a key. In this keyframe they first search for feature points and initialize 1000 tracks with these points. Then the user is required to translate the camera sideways in a smooth motion. During this motion they track the initial feature points from frame to frame. When the user hits the key for the second time, they use the tracked features as correspondences between the first frame and the current frame. With these correspondences they use the five point algorithm of Stewénius et al. [70] to estimate the essential matrix. The essential matrix defines the epipolar geometry between two images. If the calibration of the camera is known this matrix can be used to determine the relative motion of the camera between the images up to scale. After fixing the pose of the first keyframe in the origin of the map coordinate system, they triangulate map points using the tracked feature points. In embedding the essential matrix estimation in a RANSAC [21] procedure, they robustly estimate the camera pose of the second keyframe as well as the initial map. In a further step they use bundle adjustment to refine the resulting map. This leads to a map with an arbitrary scale. In the next paragraph we discuss possibilities to add further information to achieve a metric reconstruction.

Scale Estimation. The original version of PTAM by Klein and Murray [37] was developed for the purpose of augmented reality. For tracking a hand-held camera it is not necessary to reconstruct the scene in a metric scale, but it is sufficient to know the relative relation between the camera motion and the 3D feature points. Thus, they just assume that the baseline between the initial pair of keyframes is 10 cm to insert the 3D model in the AR visualization at roughly the right scale. This kind of inaccurate scale estimation might be sufficient for augmented reality, but is hardly satisfactory for robotic navigation.

If the user tells the robot to move 3 meters, the robot should move 3 meters as accurately as possible and not move a different distance after each initialization. Consequently, several research groups took different approaches in setting the scale of the reconstruction.

In general, there are two basic ways of achieving metric scale. The first possibility is to initialize the map with a rough scale estimate (similar to Klein and Murray [37]) and then correct the scale using sensors with metric information. Achtelik et al. [1] implement the scale adaption with an EKF which uses accelerometer and pressure sensor information for the scale estimation. In contrast, Engel et al.[18] use accelerometers and an ultrasonic altimeter as input for a maximum likelihood estimator of the scale. Both approaches need significant vertical motion of the MAV for a proper scale estimation.

The second possibility is using a known relation between the first pair of keyframes. This method has the advantage that no computational power or mission time needs to be wasted on scale estimation after the initialization. Wendel et al. [80] use an ARToolKitPlus marker [76] to determine the camera pose relative to the marker. This approach has some disadvantages. As the marker contains points on a single plane, this approach has problems in distinguishing rotational from translation motions if the marker is nearly parallel to the image plane. Furthermore, the depth estimation accuracy strongly depends on the space the marker occupies in the image. For a very accurate depth estimation the marker has to be prominent in the image, but in occupying a large area of the image, it is blocking a large part of the scene, which we want to reconstruct. In this work we take a very simple but effective approach to fixing the scale. Instead of having an object of a known size in the scene like Wendel et al. [80], we use an initialization platform with a known baseline. The two initial frames can then directly be treated like the images of a stereo rig. As there is no rotational motion between the frames, wide baseline feature matching can be achieved extremely fast using BRIEF - descriptors of Calonder et al. [6]. These descriptors consist of a string of binary responses to pair-wise pixel comparisons and can be matched using the Hamming distance instead of the Euclidean distance. This approach leads to a very fast initialization of the system at a metric scale.

2.2.3.2 Keyframe Selection

After a successful initialization of the map, the next question of interest is the selection criterion of the keyframes. In the original version of PTAM [37], a frame is chosen as a keyframe if the distance to all existing keyframes exceeds a fixed threshold. Weiss et al. [77] use a different heuristic. Instead of just using the metric distance to the closest keyframe, they analyze the median pixel displacement between the current frame and the closest keyframe. They add a new keyframe if the median pixel distance exceeds a fixed threshold. This approach leads to problems when the camera is steadily oscillating at the same position. In this case, more and more keyframes are added due to the pixel motion although they all contain the same information.

For our explorative approach it is important to have a cue where to look for potential map points. Thus the keyframe selection criterion is very important. First we analyze the Euclidean distance to the closest keyframe, similar to the original version of PTAM [37]. Instead of using a fixed distance threshold, we adapt the threshold depending on the median scene depth. The threshold is set in such a way that if a map point would exists perpendicular to the camera motion with a depth equal to the median scene depth, then this point should have a sufficiently large triangulation angle (0.04 rad in our experiments). Should a frame be too close to an existing keyframe it could still be valuable as it might look into an unexplored direction. Thus we also add frames if the rotational distance to all keyframes, which are within the conflicting metric distance, is sufficiently large (0.2 rad in our experiments).

2.2.3.3 Feature Matching

Due to the localization part of the approach, the pose of camera of each keyframe is already known when it is added. Additionally, the feature tracking procedure has already established some feature correspondences. As the maximum number of concurrently tracked features is limited by a fixed value, the tracking will very likely miss some of the correspondences. To maximize the number of correspondences, all existing map points are reprojected into the current keyframe and matched as described in the next paragraph.

Search for Map Correspondences. The search for correspondences of map points in a new keyframe is based on a template-based matching criterion. An existing map point is defined by a 2D location in its original keyframe as well as its detection scale. As the viewpoint might have considerably changed from the original to the current keyframe, it is necessary to warp the neighborhood of the map point. This warping, which is done with an affine transformation, tries to answer the question what the region around the map point would look like in the new keyframe. For answering this question, PTAM assumes two properties of the region around the map point. Firstly, it is assumed that the region around the point can be well approximated by a planar patch. Secondly, PTAM assumes that the normal of this patch is oriented parallel to the principle axis of the camera pose of its source keyframe. The matrix for the affine warping transformation is found by analyzing the effects of unit pixel displacements in the source keyframe on the reprojection in the new keyframe. Using the determinant of this matrix, it is also possible to find the right scale level for the template search.

After warping the region around the map point into the image plane of the new keyframe, PTAM performs a template search using 8×8 pixel patches in a fixed range around the predicted location of the map point in the new keyframe. The search is performed in analyzing the zero-mean sum of squared differences (ZMSSD) score between the warped patch and the patches of the feature points within the search radius. PTAM declares a successful match if the score of the best match is below a fixed threshold. Through this search new correspondences to existing map points can be established for map optimization.

Search for New Map Points. When the camera is moving away from its initialization pose it is necessary to add new points to ensure localization. To add new map points PTAM searches for correspondences between feature points in the new keyframe and the keyframe whose camera pose is located closest to the new keyframe. Choosing the closest keyframe has the advantage that the images are very similar to each other and no patch warping needs to be performed. The search is done using the epipolar geometry between the images. To speed up the search they do not search along the whole epipolar line for correspondences, but restrict the search to a likely depth of the map point using the mean value and the standard deviation of the scene depth. Furthermore, PTAM only searches for points on the same pyramid level as the candidate feature and rejects points which are too close to existing map points. PTAM declares a match if the ZMSSD score of the best correspondence is below a fixed threshold.

In our implementation we have added two further means of point rejection. Firstly, we do not add map points if they have feature points with a similar appearance close by. Such points are not very discriminative and will very likely be wrongly matched during tracking. To achieve this we simply analyze the ratio of the ZMSSD score between the best match and the second best match. We only accept a map point if the score of the second best match is twice as high as the score of the best match. The second rejection criterion is meant to keep the depth uncertainty of the new map points reasonably low. This is realized through enforcing a minimum triangulation angle of the map points to reject far away outliers. The theoretical background for this rejection method can be found in Section 2.3.1. Note that both rejection methods do not change the computational complexity of the problem in any way.

2.2.3.4 Map Optimization

PTAM uses a batch optimization technique known as *bundle adjustment* (BA) for the map optimization. This method minimizes the overall reprojection error of the whole map. Given a pair of a 3D map point and a corresponding 2D feature point in a keyframe, the feature point can be seen as a measurement of the 3D map point. The reprojection error of this pair is the pixel distance between the actual feature point and where the 3D point should be visible in the image according to the projection model. Instead of solving a least-squares problem, PTAM minimizes the overall reprojection error with a biweight Tukey objective function which makes the optimization more robust to outliers. The optimization is done in an iterative scheme which is called sparse Levenberg-Marquardt bundle adjustment as described in [28]. This optimization concurrently optimizes the location of the 3D map points as well as the camera poses of all keyframes. As it is an iterative optimization approach, it needs a good initialization to converge to the correct solution. In taking advantage of the sparseness of the optimization problem, the complexity of the optimization scheme can be reduced to $O(N^3)$, where N is the number of keyframes. Although the complexity is independent of the number of map points, it still comes with a large computational burden for a large set of keyframes. Klein and Murray [37] report a computation time of tens of seconds for 150 keyframes. In general, this is no problem if the system is not currently exploring new parts of the scene, as this means that the system can spend a lot of time on the optimization. On the other hand, if the system is currently exploring there will be no time to perform a global optimization. To overcome this problem, PTAM uses local bundle adjustment (LBA) prior to the global bundle adjustment (GBA). This means that, instead of using the whole set of map points and keyframes, they reduce the set to a smaller size which is relevant for the current keyframe. The set of keyframes which is to be optimized is reduced to a fixed size of 5 keyframes; the new keyframe and the four closest keyframes. The set of points is reduced to all points which are visible from this subset of keyframes. The set of all keyframes, which contain a measurement of any of the map points in the selected subset and are not already in the subset of the 5 variable keyframes, are treated as "fixed" in the optimization process. The rest of the keyframes and map points are not considered at all for the LBA. The LBA approach of Klein and Murray [37] has a worst time complexity of $O(NM)$, where N is the number of keyframes and M the number of map points. Holmes et al. [30] show that in using a relative data representation LBA can be achieved in constant time.

2.2.4 Localization

Camera localization in PTAM is achieved through the tracking of already mapped points with known 3D coordinates. For the tracking procedure PTAM uses a motion model of the camera. This means that when a new frame is received, the model assumes that the relative motion of the camera is similar to the last iteration. PTAM uses this motion propagation to initialize the camera pose of a new frame. Then it refines the pose in a coarse to fine manner. In the first step, PTAM searches for a small number of map points on the coarsest scale level. The search is done in the same manner as the search for already existing map points in the mapping procedure. This means that they warp the neighborhood of the map points in the source keyframe into the current frame and compare the resulting patches to features in a fixed distance around the "should be" locations of the map points. Then they use the resulting matches to refine the camera pose using a technique which is very similar to the map optimization. They use the same Tukey objective function to minimize the reprojection error, but this time all points treated as fixed and only the current camera pose is refined. After this coarse optimization, they search for a great number of map points (1000) and repeat the camera optimization procedure.

Localization Loss and Recovery. As PTAM is a passive localization approach it is always possible that the motion of the user causes a loss of the visual localization. While PTAM cannot predict when a loss of the localization is going to happen, it is able to detect when the localization is lost. In this case it switches to a recovery procedure based on tiny images [38]. PTAM subsamples every keyframe to a size of 40×30 pixels, blurs the image with a Gaussian kernel and subtracts the mean intensity of the image. After losing the visual localization PTAM applies the same procedure on the current frame. Then it compares this small blurry version of the current frame to the existing keyframes to find the most similar keyframe based on the SSD score. The camera pose of the current frame is first initialized with the pose of the most similar keyframe and then the camera rotation is refined using direct second-order minimization. Finally, the normal tracking procedure is reinitialized with the found pose.

In our experimental Section 5.2 we provide a detailed analysis on the topic feature detection probability, localization loss and recovery in conjunction with viewpoint changes.

In Section 3.1 we demonstrate how this information can be used to predict a loss of localization before it can occur.

2.3 View Planning

In this work we are aiming for active visual localization. In other words this means that we aim to plan the motion of the camera in such a way that we maintain the visual localization. This problem can therefore be seen as a view planning problem. Opposed to our work, the current literature in the field of view planning solely focuses on the optimization of the uncertainty and the completeness of the reconstruction. While the completeness of the reconstruction has no impact on the quality of visual localization, the reconstruction uncertainty plays an important role as we use the 3D reconstruction for the localization. As the reconstruction uncertainty directly influences the uncertainty of the localization, it is important to keep the reconstruction uncertainty as low as possible.

In this section we first discuss the topic of reconstruction uncertainty and then cross over to the topic of next-best-view (NBV) planning. We survey this field of research with regard to our work and explain how some of the ideas can be used for active visual localization.

2.3.1 Reconstruction and Localization Uncertainty

The quality of a reconstruction which was created by a structure from motion approach depends on many factors. These factors include the uncertainty of the camera calibration, the 2D image location uncertainty of feature detectors, the percentage of outliers as well as the uncertainty propagation in space and over time which can lead to a drift in the reconstruction. Not all factors have the same effect on the overall uncertainty of the reconstruction.

The uncertainty of the reconstruction in SfM originates from the imperfection of the visual system (lens distortion, sensor noise, discretization and quantization), the error in approximation of the lens distortion as well as the location inaccuracy of the response of the feature detector. The error through the lens distortion and its modeling can be summed up as the uncertainty of the camera calibration. On the one hand, the very recent research by Daftry et al. [9] shows that the quality of the camera calibration has a significant impact on the reconstruction accuracy. On the other hand, Ozog and Ostice [56] demonstrate that the explicit modeling of the camera calibration uncertainty can improve the reconstruction only if the feature detection error is very small (standard deviation of below 0.3 pixels in a laboratory) and does not lead to a significant improvement for a real world example using a SIFT [45] feature detector.

The inaccuracy of the feature detector strongly depends on the sensor noise and the image resolution. Note that the principle of feature detection and matching is a heuristic which emulates the behavior of human beings as discussed by Kanatani [34]. Consequently, it is not straight forward to find a suitable mathematical model for the uncertainty of the feature detection step. The general approach is to assume the location uncertainty follows a Gaussian distribution as the intensity variation in the neighborhood of feature points is typically unrelated [34].

Under the assumption of Gaussian noise for the feature point location, it is now possible to express the uncertainty of 2D feature points through a two-dimensional covariance matrix. As the 2D points contain noise, it is not possible to directly determine the actual location of the 3D point through the intersection of image rays. Finding the location of a 3D point from more than two views is an overdetermined problem. To find the best estimate of the 3D point location all approaches try to minimize some kind of error function. Simply solving the least squares problem of the reprojection error makes the triangulation very sensitive to wrong matches. PTAM [37] uses the Tukey biweight function to be more robust to those outliers. Alternatively, it is possible to increase the robustness of the point triangulation in embedding it in a RANSAC [21] procedure. All these approaches can be seen as maximum likelihood estimators (MLEs) for the location of a 3D point in a continuous probability distribution which represents the location uncertainty.

Beder and Steffen [4] model the 3D point uncertainty very similar to the 2D point uncertainty in assuming a 3D Gaussian distribution. Instead of intersecting image rays, they project the Gaussian distributions of the 2D feature point uncertainties and basically intersect the resulting cones to estimate the uncertainty of the 3D point. They represent the 3D point uncertainty through a covariance matrix. If one would visualize the resulting uncertainty it would correspond to an ellipsoid. An example is shown in Figure 2.1.

Beder and Steffen [4] then analyze the singular values of the covariance matrix to determine the best image pair for fixing the scale of the reconstruction. They argue that the uncertainty of a 3D point is minimal if the shape of the Gaussian distribution resembles a sphere. Therefore, they analyze the ratio between the largest and the smallest singular value of the covariance, which reflects the ratio between the largest and the smallest axis of the ellipsoid representing the distribution. As the 2D Gaussian noise distribution in

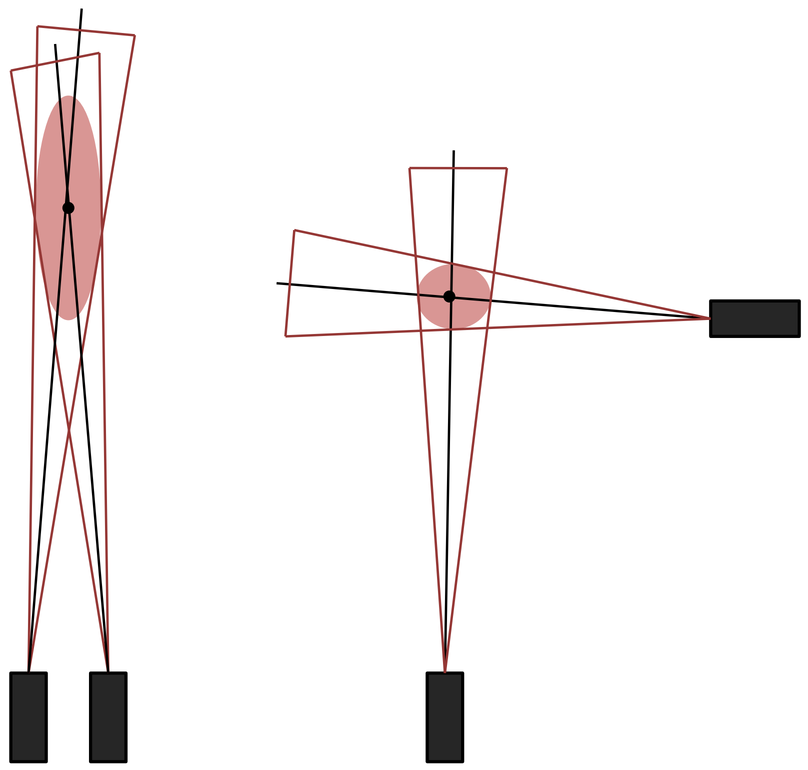


Figure 2.1: Relation between point uncertainty and triangulation angle. A small triangulation angle (left) leads to a large uncertainty along one axis, whereas a large triangulation angle(right) has a significantly smaller uncertainty. The triangles with the red lines represent the uncertainty of the projected image measurement. The red ellipses are the graphical interpretation of the covariance matrices, which are defined through the geometric constellation as well as the variance of the image measurements. The black dots represent the real world feature point, whereas the black lines represent the projections of the image measurements.

the images is assumed to be the same overall images, this measure now only depends on the geometric constellation, i.e. the triangulation angle of the 3D point. That the point uncertainty is mainly dependent on the triangulation angle was also observed by Rumpler et al. [60] in the context of stereo reconstruction.

Another way to reduce the uncertainty of a 3D point is using more measurements of this point; i.e. more views the 3D point is visible in. More measurements would mean that there a more "votes" for the 3D point location, which can be seen as more discrete samples of the underlying probability distribution. Note that a measurement only corresponds to a ray and not a point, thus care has to be taken in the selection of the images. Using images from the same view over and over again can only reduce the uncertainty of the image ray and does not necessarily lower the depth uncertainty of the 3D point. Moreover, Sugaya and Kanatani [73] have observed that there is a strong correlation between the uncertainty over the frames of a video stream. This means that taking a lot of sequential frames from the same position very likely leads to a convergence to a wrong value as the independence assumption of the samples is strongly violated. Klein and Murray [37] overcome this problem by enforcing a minimum spatial and temporal distance between the keyframes selected for the reconstruction.

Eudes and Lhuillier [20] present a way to propagate the uncertainty of the keyframe poses and the 3D points in form of covariances in the context of local bundle adjustment (LBA). They use a Taylor approximation to achieve first order error propagation. This error propagation is still computational expensive and can only be realized in real-time with an unrealistic independence assumption and parameter tuning. Furthermore, this assumption can only be used in combination with a special LBA approach of Eudes and Lhuillier [20]. They also present a more realistic (weaker) independence assumption which is more general and can be used with other LBA approaches, but cannot be computed in real-time.

As we approach the topic of localization and reconstruction from an active and explorative perspective, it is important that the uncertainty estimation of 3D points can be calculated in real-time. To achieve the real-time performance, we do not use the expensive calculation and analysis of covariance matrices, but instead use a heuristic based on the triangulation angle. This heuristic does not represent the real uncertainty of a 3D point, but can be used as a cue to determine the relative uncertainty between points.

2.3.2 Next-Best-View Planning

The research topic of Next-Best-View (NBV) planning is concerned with the improvement of a 3D reconstruction. On the one hand, this means that they aim to remove outliers and reduce the uncertainty of inliers. On the other hand, this topic is also concerned with the maximization of the completeness of the reconstruction. As inferable from the name, Next-Best-View algorithms try to find the next best viewpoint for acquiring the next image given the current reconstruction state. The definition of "best" strongly depends on the task as well as the available information. Consequently, there are a lot of different approaches to the view selection. Most of the literature is "only" concerned with the optimization of the 3D reconstruction and its completeness, whereas we want to additionally guarantee localization safety and the generation of high quality map points which allow for a high accuracy localization.

The general approach in reducing the uncertainty of the reconstruction is to use a cost function which is based on covariance matrices. Wenhardt et al. [81] analyzed which optimization criterion based on covariances is better suited for the NBV selection. They compared three different optimization criteria. The first one, the *D-optimally criterion*, is an entropy minimization based on the determinant of the covariance matrix. The second criterion, the *E-optimally criterion*, minimizes the largest eigenvalues of the covariance matrices and the third criterion, the *T-optimally criterion*, minimizes the trace of the covariance matrices, which corresponds to a minimization of the sum of eigenvalues. Their results suggest that all three criteria are equally suited for the NBV selection.

Pioneer work on the topic of active vision was done by Davison and Murray [11] fifteen years ago. Their EKF-based active vision system consisted of a pair of two movable cameras on a non-holonomic ground robot. Opposed to more recent approaches, they only use a handful of feature points for mapping and tracking. They represent the location uncertainty of the map points through covariance matrices. To decrease the uncertainty of the map points, they select map points with a high innovation gain for tracking. This means that they try to observe a map point as perpendicular as possible to the axis of the highest uncertainty of the point so as to reduce the uncertainty of this point. This criterion is strongly related to the *E-optimally criterion*.

Other approaches, such as [14, 31], do not simply optimize the reconstruction uncertainty of a sparse scene representation (point cloud), but use meshing techniques to additionally incorporate visibility constraints. Hoppe et al. [31] use the 3D mesh to reduce the space of possible camera poses and respect the point visibility via ray casting. Using this additional information they optimize the *E-optimally criterion* while considering the necessary camera overlap through counting the number of mesh triangles which are visible in neighboring cameras. Dunn and Frahm [14] also incorporate the matching probability into their optimization scheme which consists of three parts. The first part aims to reduce the reconstruction uncertainty in rewarding orthogonality between the major axis of the uncertainty and the new viewing direction. This is achieved through penalizing the norm of the product of the viewing direction vector with the eigenvector matrix. The second part uses the 3D mesh to determine the visibility of a surface patch. In analyzing the area which a surface patch occupies in the image projection, they can determine whether a patch is visible at all as well as estimating how likely it is for a patch to be recognized. The third part also incorporates the surface texture saliency by measuring the entropy of the autocorrelation. These three parts are then combined to a single function so that all three criteria can be jointly optimized.

Another approach by Trummer et al. [75] uses an extended version of the *E-optimally criterion*. In assuming a spherical camera motion they can find the shortest path to the optimal solution in a closed form. The drawback of this formulation is that it only allows the optimization of one point at a time.

Haner and Heyden [26] propose a more general approach. They concurrently minimize the camera path as well as the reconstruction uncertainty. In their cost function they multiply the trace of the covariance matrix (*T-optimally criterion* although they call it *A-optimally*) with a weighted function of the camera path. In their view planning algorithm for visual SLAM, they initialize the camera trajectory with a discretely sampled linear interpolation between the current pose and the target camera pose. Then they use their cost function in a Levenberg-Marquardt optimization scheme to find the optimal path. The complexity of the path optimization is $O(MN)$, where M is the number of discrete samples on the path and N the number of considered feature points. They report that the optimization is not yet suited for real-time applications without limiting the number of considered feature points. Note that this approach does not consider the topics of collision avoidance, point visibility or localization quality.

All of the NBV criteria, which have been proposed up to the present, do not quite fit our purpose as the main aim of our work is not the improvement of the reconstruction but the stability of the visual localization. As the localization accuracy depends on the reconstruction uncertainty, we incorporate a term which penalizes a high point uncertainty in our optimization criterion. As we aim for real-time performance with thousands of map points and camera positions, we do not base this term on the expensive analysis of covariance matrices, but use a heuristic based on the point triangulation angle only. Inspired by the work of Dunn and Frahm [14], we also incorporate the feature recognition probability depending on the viewpoint. This probability can be computed extremely fast without an expensive meshing step through analysis of the geometric constellation of the feature observation as described in Section 3.1.

2.4 Common Navigation Approaches for Monocular Micro Aerial Vehicles

The idea of an autonomous robot which operates only with a single camera has not just recently appeared. Over the last twenty years many researchers have approached the topic from various sides. The early work in this field [43, 53, 69] mainly focuses on the detection and avoidance of obstacles using optical flow and insect-inspired motion patterns. Very recent work by Ross et al.[58] has demonstrated that a reactive controlling behavior for MAV flight in cluttered environments can be learned using AI training techniques.

The problem of systems which rely solely on reactive controlling is that they have a very limited location awareness. This means that they can see that an obstacle is in front of them, but they do not know e.g. in which part of the forest they are or where they came from. Simply using inertial sensors for localization is insufficient as these sensors are in general very inaccurate, which leads to a drastic drift in the pose estimation. To allow for a drift-free localization other exteroceptive sensors are necessary.

One approach is to use an external high-frequency and high-accuracy tracking system. Using such a state estimation, impressive results of aggressive flight have been achieved [46, 49]. However, this kind of state estimation limits these approaches to the laboratories, which is not what we aim for in this work.

A more general approach is to equip the MAV with additional sensors and perform SLAM for pose estimation. On the one hand, additional sensors can be used for a better localization, on the other hand, they increase the payload on the MAV. Opposed to ground robots, MAVs have a very limited maximum payload and also a very limited time frame for operation. An increase of the payload directly leads to a decrease of the effective operation time. In this work we aim to achieve autonomous MAV navigation with a minimum of exteroceptive sensors. This means that we only use a single light-weight low-cost passive camera to achieve visual localization.

Another important difference between multirotor MAVs and ground robots is that an airborne MAV is an unstable system. This means that even for a "simple" task as "not moving", i.e. hovering, it needs permanent pose feedback and controlling efforts. Thus a lot of current research focuses on the topic of monocular servoing [1, 18, 19, 36, 78, 80]. These works demonstrate that MAV controlling can be achieved with a single passive camera, although the pose estimates contain significantly more noise and arrive far less frequently than the pose estimates of external tracking systems.

All the monocular MAV controllers mentioned above can be seen as absolute pose controllers. This means that they are aware of the current camera pose with respect to a world coordinate system and aim to move the MAV towards a predefined pose in this coordinate system. Such a controller enables the flight on arbitrary linear trajectories. In this work we make use of this trait in our destination-based local planning scheme described in Chapter 4. Instead of sending any destination to controller, we send the optimal reachable destination in the local neighborhood. For our experiments we use the controller of Wendel et al. [80], but any other pose controller would work just as fine. In our experiments we demonstrate that our planning approach is very resilient against navigational imperfection.

2.4.1 Path-Planning and Collision Avoidance

The topics of collision avoidance and path-planning have always been closely related. In path-planning one seeks to find a path on which the robot can move as safely and as fast as possible towards a destination.

In dynamic environments it is not sufficient to only consider collision avoidance in the planning phase, e.g. a path which was available earlier can now be blocked by a person. Therefore a lot of research is concerned with the topic of reactive obstacle avoidance for monocular systems [43, 53, 58, 69]. The very recent work of Ross et al. [58] is concerned with "teaching" a monocular MAV to "learn" the right reactions to avoid obstacles. Using state-of-the-art learning techniques they achieve impressive results for the flight in cluttered environments.

The drawback of reactive collision avoidance with monocular systems is that it limits the "allowed" motion patterns to a single direction, which is in this case "forward". Unfortunately, a simple forward motion is insufficient to generate sufficiently large triangulation angles for accurate monocular SLAM. This leads to two competing requirements. On the one hand, the safest way to move is forward as the camera points in this direction and thereby allows obstacle detection, on the other hand, an accurate map can only be constructed if the MAV motion contains a large component perpendicular to this direction.

As obstacle avoidance is not the main goal of this work but more a necessity for experimental evaluation, we simply assume a static environment. This assumption makes it possible to remember the location of obstacles and also free space, which drastically improves the agility of the monocular MAV as it can now fully use all the available degrees of freedom and is not limited to a narrow motion cone.

Scene Representation. For ground robots the most popular representation of the environment is a two-dimensional probabilistic map. For this representation many efficient local as well as global planners are available as open source[5]. Such a 2D representation has also been used for indoor mapping with MAVs [25, 62]. The problem with this representation is that it is based on the assumption that the world only consists of horizontal ground planes and vertical walls. This means that even objects which are typically found in indoor environments, such as chairs or tables, cannot be represented properly. The advantage of this approach is that mapping and planning can be achieved with freely available and well tested approaches, but by using this representation the only advantage of MAVs over ground robots, i.e. the increased mobility, is unnecessarily reduced.

So as to use the full capabilities of an MAV a 3D scene representation is essential. Shen et al. [63] demonstrate that autonomous indoor exploration with a quadrotor MAV in multi-level indoor environments can be achieved with a probabilistic 3D map representation. Opposed to our work, they soften the problem in using additional sensors, i.e. an active RGBD-camera and a laser scanner, for the exploration task.

A probabilistic 3D map has the disadvantage of having a memory consumption which is cubic to the scene size and that it does not inherently provide support for monocular SLAM. However, this probabilistic representation has one major advantage over other more efficient representations, such as a point cloud or a surface mesh. This advantage is the explicit representation of free and unknown space. For a safe navigation it is of utmost importance to reduce the MAV motion to areas in the scene which are known to be unoccupied; i.e. free space. Simply avoiding known obstacles can easily lead to collisions with obstacle in unknown parts of the scene. In order to ensure a safe navigation for our MAV, we decided to use a probabilistic map alongside with the sparse point cloud generated by PTAM [37]. In our experiments we used the efficient probabilistic map implementation of Hornung et al. [32], which is based on an octree representation and keeps the memory consumption minimal through representing unknown space only implicitly via null pointers in the tree.

Path-Planning. Path-planning in 3D is a very complex problem. If one would directly treat the volumetric 3D representation as a graph-optimization problem, the maximum size of the map would be very limited. Even a small map of 10 m in each dimension with a coarse resolution of 10 cm would lead to 10^6 nodes and approximately $14 \cdot 10^6$ bidirectional edges (allowing diagonal transition). To reduce the graph to a feasible size current

[5]http://www.ros.org/wiki/navigation

approaches, such as [33, 63], use randomly sampled Probabilistic Roadmaps (PRMs) [24] or some sort of Rapidly-exploring Random Trees (RRTs) [35]. A suitable and collision free path can then be extracted using standard graph search algorithms like A* or D*.

The approaches mentioned above aim to find a collision free path for MAVs. In this work we augment the problem one step further. As we are restricting our system to a localization based on a single passive camera, it is crucial not only to plan the three-dimensional trajectory of the MAV with respect to collision avoidance, but additionally consider the field of view of the camera at all times. Although graph-based path-planning methods could be extended to the required four-dimensions (3D position + yaw), several reasons motivated us to take a different approach. While a collision check in a tree-based probabilistic map can be calculated very fast ($O(d)$ where d is the tree depth), the evaluation of the visual localization requires the evaluation of the projection of all map points. Thus, the runtime of a simple validity check of a vertex in the graph now becomes linear to the number of map points. Furthermore, the evaluation of the visual localization has not only to be calculated for each vertex in the graph but also each edge as we have to ensure localization safety at every point in time. As a multirotor MAV is an unstable system, unforeseen disturbances like turbulences or network lags can always cause the MAV to stray significantly from the planned path. In such a case the whole planning procedure has to be repeated which can hardly be achieved with a graph-based approach in real-time. For these reasons we decided to take a local planning approach, which can be calculated very fast and thus can be easily repeated. Our proposed adaptive destination-based planning scheme is described in Chapter 4.

2.5 Summary

In this chapter we discussed several fields of research of which all are related to this work. The myriad of related research topics shows the complexity of the problem at hand. In the next chapter we outline three novel measures; the localization quality, the point generation likelihood, and the collision probability. These three measures are the main contribution of this work and can be seen as the pillars of the bridge between passive monocular SLAM and active monocular localization.

Chapter 3

Quality Measures for Active Monocular Localization

Contents

Active monocular localization is a combined expression with a deeper meaning than the sum of its terms. Nevertheless, we will revisit each term on its own for a better clarification of the combined meaning. The meaning of ”localization” can be simply described as knowing *where* something is in relation to a known structure. In our case, this means knowing the pose of the camera and the MAV in relation to a predefined origin. ”Monocular” states that the only exteroceptive sensor that is used for this localization is a single camera. The ”active” component basically means to ”actively” choose a pose to improve the localization, while performing a certain task,which is in our case reaching a destination pose. But this raises the question: What pose is the best to choose?
One of the main contributions of this work is providing an answer to exactly this question. In this chapter, we concern ourselves with how to measure the quality of a given camera/MAV pose, in order to find the ”best” pose available.

The task of active monocular localization has two competing aspects. The first aspect is to keep the overall state of the system safe and sound. On the one hand, the system should respect the physical availability and safety of a pose and stay away from any objects and unknown environments. On the other hand, it is necessary to keep the monocular localization as accurate and stable as possible to ensure a safe navigation.

While the first aspect makes the system stay at a safe and well-known location with a good localization, the second aspect is concerned with achieving a higher level goal. In our case, the main goal is to reach a predefined destination pose. However, simply moving towards the destination pose on the shortest possible path will very likely lead to a loss of the visual localization, as most of the scene is still unexplored after the initialization. Consequently, it is necessary to explore the scene up to an extent that allows for a navigation without localization loss.

Satisfying all aspects concurrently is a non-trivial task, for which this work proposes a solution. This chapter focuses on how to measure the competing aspects, whereas Chapter 4 describes how to combine those measures to an autonomous explorative navigation system.

The remainder of this chapter is structured as follows. Section 3.1 describes a novel measure, called localization quality, which respects the uncertainty as well as stability of the monocular localization for an arbitrary camera pose. Section 3.2 introduces a novel measure for the likelihood to generate new points from an arbitrary camera pose with the aim of ensuring localization safety through the generation of new useful map points. Section 3.3 outlines a way to estimate the safety of an MAV pose, through the estimation of the collision probability using a probabilistic occupancy grid. Finally, Section 3.4 closes this chapter in summarizing the three proposed measures.

3.1 Localization Quality

Monocular localization has mainly been developed by and for the Augmented Reality community. Consequently, most approaches such as PTAM [37], DTAM [54], MonoSLAM [12], ScaViSLAM [71] follow a passive localization approach. In other words, this means that the user can move through the scene and the MonoSLAM system tries to find the best possible localization. The AR community can hardly restrict the motion of the user to valid motions, so there was no urgent need to predict the probability of a localization loss, although the user could also benefit from this information. In contrast, it is clearly possible to restrict the allowed motions of the active component in the robotics domain.

Most approaches such as [12, 71] simply assume that sufficient information is available for localization. The closest measure to a localization failure prediction was proposed by Klein and Murray [37]. They keep track of the ratio between the number of successfully tracked points and the number of points considered for tracking. Sadly, this measure only provides a very limited prediction scope as it is solely based on the current tracking information. In their implementation (PTAM) the tracking quality is represented by three states; namely good, dodgy and bad. The states are only separated by fixed thresholds of the previously mentioned ratio. In practice, the tracking quality tends to switch from good to bad within a split second without giving any prior warning.

To widen the prediction scope, we design a measure which is independent of the actual tracking state and can be computed for arbitrary virtual or real camera poses. In order to avoid the expensive calculation of the reconstruction uncertainty based on covariances [4, 22], we base this novel measure solely on geometric properties of the reconstructed scene. All that is needed is a structure that we call a "linked point cloud".

A linked point cloud consists of the set of 3D points (map points), the set of camera poses (keyframes) and the information which point was observed from which camera (links). Basing our novel measure exclusively on the geometric constellation of the linked point cloud results in another appealing property aside from a fast computation. All the information needed for the construction of a linked point cloud is available in every point-based MonoSLAM framework, as this information is essential for the 3D point generation and optimization. Consequently, it is possible to use the measure in combination with nearly all currently available MonoSLAM frameworks, such as PTAM [37] or ScaViSLAM [71].

We call this novel measure "localization quality" as it represents the inverse likelihood of losing the visual localization. The localization quality measure itself is based on two other novel measures. The "geometric point quality" measure approximates the reliability and localization uncertainty of a point in the linked point cloud, whereas "point recognition probability" measure represents the likelihood of recognizing such a point from a given viewpoint. The following subsections will explain the necessary steps to calculate the proposed measures as well as how to combine them in more detail.

3.1.1 Geometric Point Quality

The main purpose of the geometric point quality measure is to provide an estimation of the reliability and location uncertainty of a point in the linked point cloud structure. This can be seen as the estimation of the "reconstruction uncertainty", which is a very active field of research. In the next paragraph we shortly summarize the general approach of modeling the uncertainty of 3D points as well as its relation to the point triangulation angle. A thorough survey of current literature in this field can be found in Section 2.3.1.

In general, the location uncertainty of a point can be well modeled with the assumption of a Gaussian distribution around the location of the detected 2D image correspondence of a 3D point. For the point projection task, this means that a projection ray is then accompanied by a cone-like probability distribution. To model the actual location uncertainty of the 3D point, one can use a covariance matrix, which represents the multivariate Gaussian probability density function of the intersection. The extent of the uncertainty is mainly dependent on the uncertainty of the 2D image correspondences and their relative triangulation angle.

As we want to avoid the expensive calculation and analysis of covariance matrices, we only use the point triangulation angle for our geometric point quality measure. The idea of our formulation is to reward large triangulation angles as they potentially decrease the location uncertainty of the 3D points.

Furthermore, we incorporate the probability of a point being an outlier into our quality measure. Outliers originate from wrongly matched keypoints, which produce 3D points without a real world correspondence, i.e. a point in thin air. Every SfM approach suffers to some degree from outliers even if outlier rejection is applied at every step of the reconstruction chain. It was observed that a high percentage of bad outliers, i.e. map points that are far away from any object, has only very few image correspondences. An example is shown in Figure 3.1. We integrate this observation in a heuristic which penalizes points with too few image correspondences.

The remaining part of this subsection will provide the mathematical formulation of the geometric point quality measure.

Penalizing the Point Uncertainty. For a fast estimation of the location uncertainty of a point, we neglect the uncertainty of the individual image measurements and solely consider the relative triangulation angles of a 3D map point. The proposed angle dependent quality q_α of a map point is based on the maximum value of all relative triangulation angles, as it has the biggest influence in limiting the point uncertainty. The following formulation keeps the computational cost very low, while still penalizing points with a high localization uncertainty.

First of all, let the relative triangulation angle $\alpha_f(i,j)$ between a 3D map point f and two keyframes i and j be defined as

$$\alpha_f(i,j) = acos(\frac{v_{fc_i} \cdot v_{fc_j}}{\|v_{fc_i}\| \cdot \|v_{fc_j}\|}), \tag{3.1}$$

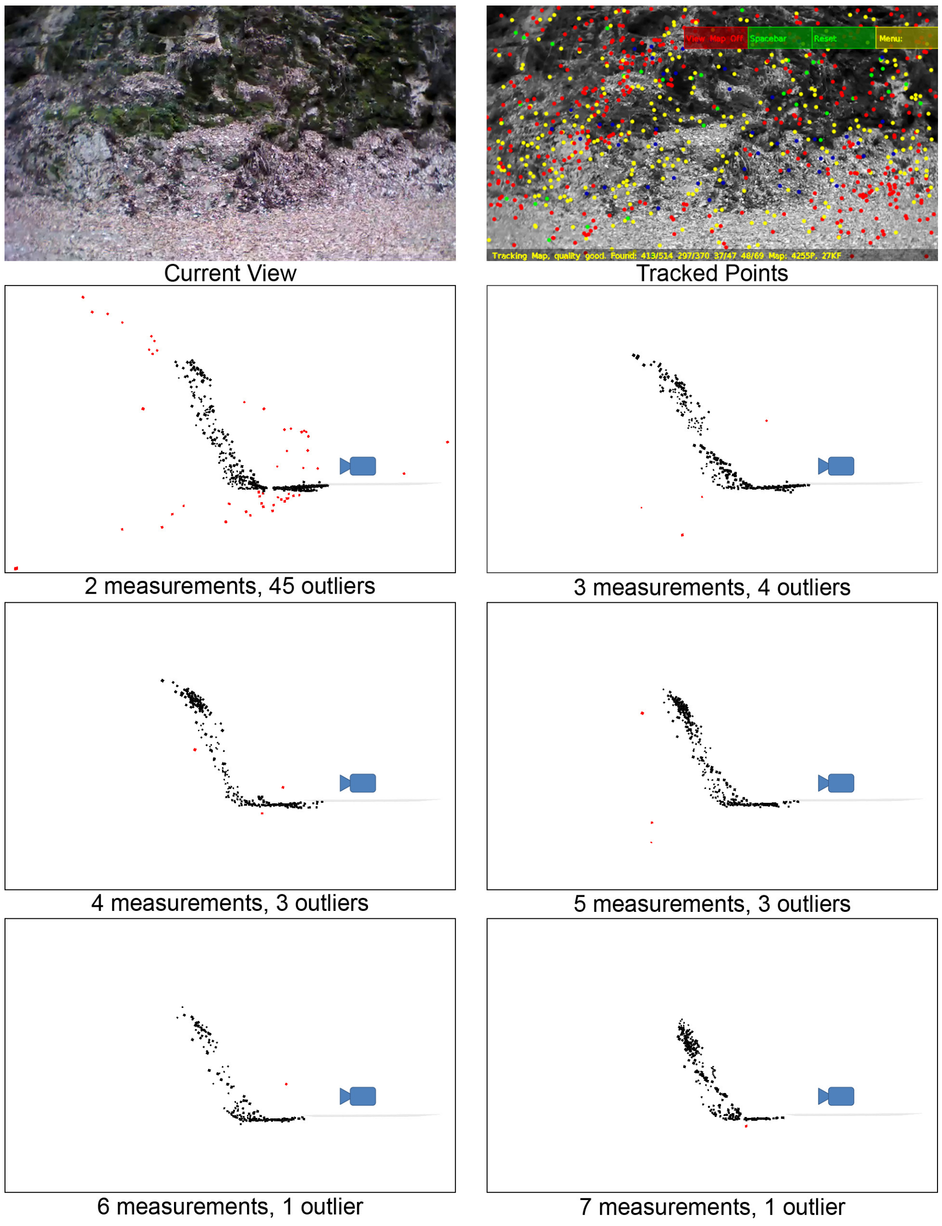


Figure 3.1: Example of the relation between outliers and the number of image measurements in a natural scene. Top left: Original image from the MAV of a natural wall with vegetation. Top right: The same image with tracked points from PTAM [37]. Row 2-4: The point cloud reconstructed from 28 keyframes viewed from the side. The red points are severe outliers, without any real world correspondence, whereas the black points can be seen as valid map points. The blue object symbolizes the camera of the MAV at its current location facing the wall. Each reconstruction figure only shows points with a specific number of keyframe measurements (links).

where v_{fc_i} and v_{fc_j} are the vectors from the currently concerned 3D map point f to the camera centers c_i and c_j of the keyframes i and j.

Then we only consider the maximum angle between a point f and its keyframes, as it has the biggest influence in limiting the point uncertainty

$$\alpha_{max} = max\{\alpha_f(i,j) : i,j \in K_f\}, \tag{3.2}$$

where K_f is the set of keyframes, which contain a measurement of point f.
Thus, we can define the triangulation angle dependent point quality q_α as

$$q_\alpha = \begin{cases} \alpha_{max}/\alpha_{cap} & \text{if } \alpha_{max} \leq \alpha_{cap} \\ 1 & \text{otherwise} \end{cases}, \tag{3.3}$$

where α_{max} is the maximum angle, which is scaled and capped by the angle α_{cap}. The limitation of the angle through α_{cap} has two main reasons. Firstly, it is necessary to scale the quality between 0 and 1. Secondly, it is wise to limit the power of the angle rewards and penalties, as it was observed that the largest triangulation angle is often found on outliers which appear very close to the camera. The parameter α_{cap} basically defines which triangulation angle can be regarded as "sufficiently large". In matters of uncertainty this can be interpreted as a "sufficiently low" point uncertainty.

Note that the value of α_{cap} does not have to be defined by hand, but can be inferred at start up. In our experiments we set this parameter in a "what you see is what you get" manner. To infer the value of α_{cap} we analyze the linked point cloud when we start our autonomous mission. First, we push the maximum triangulation angle of each point into an array and sort the array according to these angles in an ascending order. In order to be robust to outliers, we do not set α_{cap} equal to the maximum value of this array, but instead use the n^{th} element. This n^{th} element is selected very similar to the median value. The only difference is the instead of choosing the element at $\frac{1}{2} \cdot N$, where N is the size of the array, we choose the element at $\frac{3}{4} \cdot N$. In other words this means that we disregard the quarter of the array with the largest triangulation angles. This automatic inference renders a manual parameter adaption to new scenes unnecessary and helps the system to adapt to the present flight scenario.

Respecting the Outlier Probability. An image measurement of a map point can be seen as a proof of its existence. Hence, the probability of a point being an outlier decreases with its number of linked keyframes. The proposed quality q_{out} aims to penalize map points that have a low number of links as they have insufficient proof of their existence. Thus, we define the outlier penalizing point quality q_{out} as

$$q_{out} = \begin{cases} 1 & \text{if } |K_f| \geq N_g \\ |K_f|/N_g & \text{otherwise} \end{cases}, \tag{3.4}$$

where $|K_f|$ is the total size of the keyframe set of point f. N_g represents a "good" minimum size for the set for which the outlier probability is sufficiently low.

The Geometric Point Quality. Finally, we define the overall geometric point quality as a product of the angle dependent and the outlier penalizing quality measures. This leads to the resulting quality q_f of a 3D map point f as

$$q_f = q_\alpha \cdot q_{out}. \tag{3.5}$$

Through this definition a point can only get a high score if its maximum triangulation angle is sufficiently large and it has a sufficient number of image measurements to attest its existence. An example of the two dimensional quality function for $N_g = 4$ is illustrated in Figure 3.2.

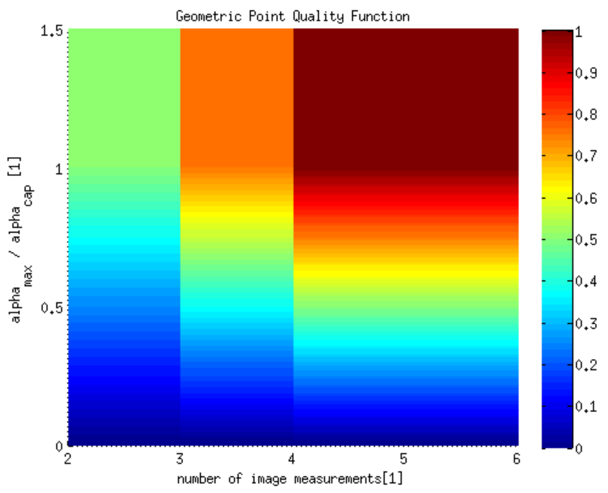


Figure 3.2: Example of the geometric point quality function q_f for $N_g = 4$ as in Equation 3.5. The function itself is color coded and dependent on the number of linked keyframes $|K_f|$ (horizontal axis) as well as the maximum relative triangulation angle α_{max} (vertical axis) of a 3D map point f.

3.1.2 Point Recognition Probability

For monocular tracking and localization it is very important that already generated 3D map points can be repeatedly recognized over a series of images. Consequently, it is important to know if a certain map point will be recognizable from a certain viewpoint. For instance, consider a richly textured wall which is represented in our reconstruction through a notable amount of coplanar map points. Let these map points all be generated from the same pair of keyframes. If we simply translate the camera a few centimeters from the original position, then nearly all map points will be recognized. In contrast, if the camera is moved farther, this can significantly change the viewing direction as well as the distance to the map points. This changes the appearance of the associated real world features of the map points in the 2D image, which results in a decrease in the number of recognized map points.

Note that the recognition probability strongly depends on the type of the used features. For our experiments we set the parameters of our model to fit the feature detection system of PTAM [37]. In our model, we respect the scale difference as well as the angle variation from the original viewpoint of the key frame to the viewpoint of an arbitrary query frame.

Viewing Angle Dependency. Let us assume that the world can be approximated by planar patches. Then the projection of a patch in the image contains a maximum of information if the principle axis of the camera is oriented parallel the patch normal. If we increase the relative angle between those two vectors, i.e. change the viewing angle, this means that area of projection shrinks. Hence, the larger the relative angle grows the less information is contained in the image. When the relative angle reaches 90° or more, the image contains no information about the patch.

Thus, it is plain that there exists a relation between the viewing angle and the point recognition probability for every detection approach. The evaluation of Mikolajczyk et al. [50] on affine region detectors suggests that this relation has nearly linear properties between 20° and 60°. Our own experiments with feature detection system of PTAM [37] in Section 5.2 support this notion. Therefore, we model the recognition probability with a piecewise linear function. We define the viewing angle dependent probability function $p_\alpha(k, i)$ between a keyframe k and an arbitrary query frame i for a map point f

as

$$p_\alpha = \begin{cases} 1 & \text{if } \alpha_f(k,i) < \alpha_0 \\ 1 - \frac{\alpha_f(k,i)-\alpha_0}{\alpha_1-\alpha_0} & \text{if } \alpha_f(k,i) \geq \alpha_0 \text{ and } \alpha_f(k,i) < \alpha_1 \ , \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where $\alpha_f(k,i)$ is the angle between the keyframe k and a query frame i as defined in Equation 3.1, and α_0 and α_1 are constants which can be determined experimentally.

Scale Dependency. Changing the distance to a map point inherently changes the scale of the associated feature of the map point in the image. Respecting scale is more complex than the angle, as most systems (including PTAM [37]) are based on an image pyramid. Thus, one has to respect at which scale the original feature was detected to know at which scale the recognition probability starts to decay. For instance, if the feature of a map point is extracted at the finest scale (the original image), it is possible to detect the map point on a coarser level if the camera is moved closer to the feature. On the other hand, if the camera is moved farther away it is not possible to detect the same map point on a finer scale level as there is no such level. Consequently, there is a strong difference in the resilience to scale changes depending on the pyramid level the map point was originally detected on.

Similar to the viewing angle dependent case, we use a piecewise linear function to approximate the recognition probability observed in our experiments. In contrast to the angular case, the shape of this function is very similar to the silhouette of a hill with a flat top. Furthermore, the parameters of this function vary for each possible scale level.

Let the relative scale change of a feature point f between a keyframe k and an arbitrary query frame i be defined as

$$s(k,i) = \frac{d(f,i)}{d(f,k)}, \tag{3.7}$$

where $d(f,i)$ and $d(f,k)$ are distances from the feature point f to the camera centers of the frames i and k respectively. Then the recognition probability of a map point f depends on the scale level l on which the associated feature was originally extracted in keyframe k and the relative scale change from keyframe k to the query frame i. Thus, we define the scale dependent recognition probability as

$$p_s = \begin{cases} \frac{s(k,i)-S_{0,l}}{S_{1,l}-S_{0,l}} & \text{if } s(k,i) \geq S_{0,l} \text{ and } s(k,i) \leq S_{1,l} \\ 1 & \text{if } s(k,i) > S_{1,l} \text{ and } s(k,i) < S_{2,l} \\ 1 - \frac{s(k,i)-S_{2,l}}{S_{3,l}-S_{2,l}} & \text{if } s(k,i) \geq S_{2,l} \text{ and } s(k,i) \leq S_{3,l} \\ 0 & \textit{otherwise} \end{cases}, \tag{3.8}$$

where $S_{*,l}$ are experimentally determined constants of the extraction scale level l of the concerned map point f.

Point Recognition Probability. The final recognition probability is a two dimensional function which depends on the relative angle and scale difference between the extraction keyframe of a map point and a query frame. An illustration of the resulting probability function can be found in Figure 3.3.

The overall recognition probability p_f of a map point f can then be simply defined as

$$p_f = p_\alpha \cdot p_s. \tag{3.9}$$

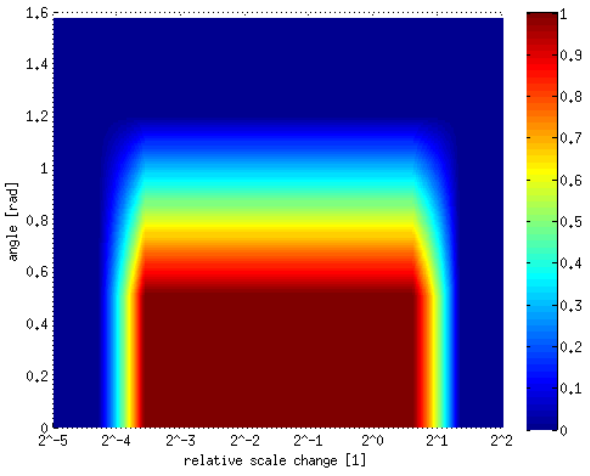


Figure 3.3: Recognition probability function for map points detected on the finest pyramid level. The recognition probability is color-coded and depends on the viewing angle (vertical axis) and change in scale (horizontal logarithmic axis).

3.1.3 Localization Quality Measure

The main aim of the localization quality measure is to prohibit the system from losing the visual localization. In that sense, it is very important that the measure does not only represent the inverse localization uncertainty, but also incorporates the localization robustness of a given pose. Our proposed measure is designed to express the robustness against three important disturbances: First of all, the measure incorporates the feature recognition probability with respect to viewpoint changes, as discussed in the previous subsection. A second relevant disturbance is unpredictable occlusion. Although the task of this work is the navigation in a static environment, a system which is very sensitive to occlusion can only be used under laboratory conditions. Under normal conditions, e.g. in urban areas, there will always be motion in the scene, in terms of people, cars, or trains. Consequently, it is always possible that parts of the scene will become unpredictably occluded during the execution. Finally, it is crucial for our autonomous system to respect navigational imperfection. Even if the MAV has reached a pose, it will never stay at the exact same position, but always move slightly, as it is a highly dynamic system and even holding a position in the air is an instable state. To respect all three disturbances as well as the position uncertainty of the 3D map points, we propose a grid based measure. The main purpose of the grid is to group and reduce the data for further processing. On the one hand we want to maximize the data reduction, on the other hand it is necessary to retain sufficient location information for our further processing steps. In our experiments, we set the side length of the grid to 8 bins in each direction. Empirically, we found that this size is the smallest power of two which still retains sufficient location information. An example of the 8x8 grid is shown in Figure 3.4.

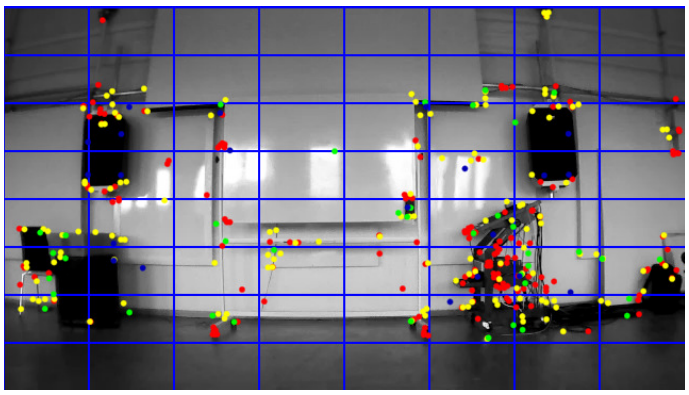


Figure 3.4: Image with tracked map points. Blue lines represent the borders of the 8x8 image bins, which are used to simplify the data.

Data Reduction. Firstly, we project the already mapped points back into the virtual image of the query frame. Then we sum the information of all map points inside a bin b to a single score $s_0(b)$

$$s_0(b) = \sum_{f \in b} p_f \cdot q_f, \tag{3.10}$$

where p_f is the point recognition probability of a point f as defined in Equation 3.9 and q_f is the point quality as defined in Equation 3.5. Note that by this definition, the score of a bin with only one high quality point can be equal to the score of a bin containing a lot of low quality points. As not to over-reward huge amounts of points in a bin, we limit the maximum quality of a bin to $s_{0,max}$. This value, very much like α_{cap}, can be automatically set at start up. When we receive the command that starts our autonomous mission, we analyze the bin scores for the current camera position. Then we use the maximum score of all analyzed bins as $s_{0,max}$. Using the maximum score, we can define the normalized quality q_b of a bin b as

$$q_b = \begin{cases} \frac{s_0(b)}{s_{0,max}} & \text{if } s_0(b) < s_{0,max} \\ 1 & \text{otherwise} \end{cases}. \tag{3.11}$$

Localization Uncertainty Reduction. The general approach for the monocular localization uses the reprojection error to optimize the camera pose estimate. Similar to the depth uncertainty of a 3D point, the location uncertainty of a camera pose increases if the relative angles between the camera and its linked 3D points decreases. As the relative angles get smaller the spatial distance between the 2D image correspondences of the 3D points declines as well. This leads to an increasing influence of the 2D location uncertainty and consequently an increased depth uncertainty of the pose estimate.

To ensure that the influence of the 2D uncertainty is kept minimal, we want to reward large distances between the image correspondences of the 3D map points. Hence, we define the relative score of a bin not only on its own value, but in relation to the other bins and their geometric constellation.

Each bin b has a distance $d_{max}(b)$, which is defined as the maximum possible distance from this bin to any other bin. Using this measure we then define the relative score $s_r(x)$ of a bin x from the total set of bins B as

$$s_r(x) = q_x \cdot \max_{y \in B} \{ q_y \cdot \frac{d(x,y)}{d_{max}(x)} \}, \tag{3.12}$$

where $d(x,y)$ is the relative distance between the bins x and y. Through this formulation a bin can only score high, if it has a good quality score itself and there exists another bin with a high quality which is located far-away in the image. In averaging over all bins we can then obtain what we call level-0-quality

$$q_0 = \frac{1}{|B|} \cdot \sum_{x \in B} s_r(x). \tag{3.13}$$

Note that through averaging this measure rewards views with well distributed image correspondences as these views are very robust to unpredictable occlusion.

Robust Grid Pyramid. The level-0-quality measure is a very good basis but has two major drawbacks. Firstly, to achieve a high score nearly all bins have to contain features, which in urban surroundings will hardly ever be the case and does not necessarily have to increase the localization quality. Secondly, it does not incorporate the robustness against the positioning error in any sense.

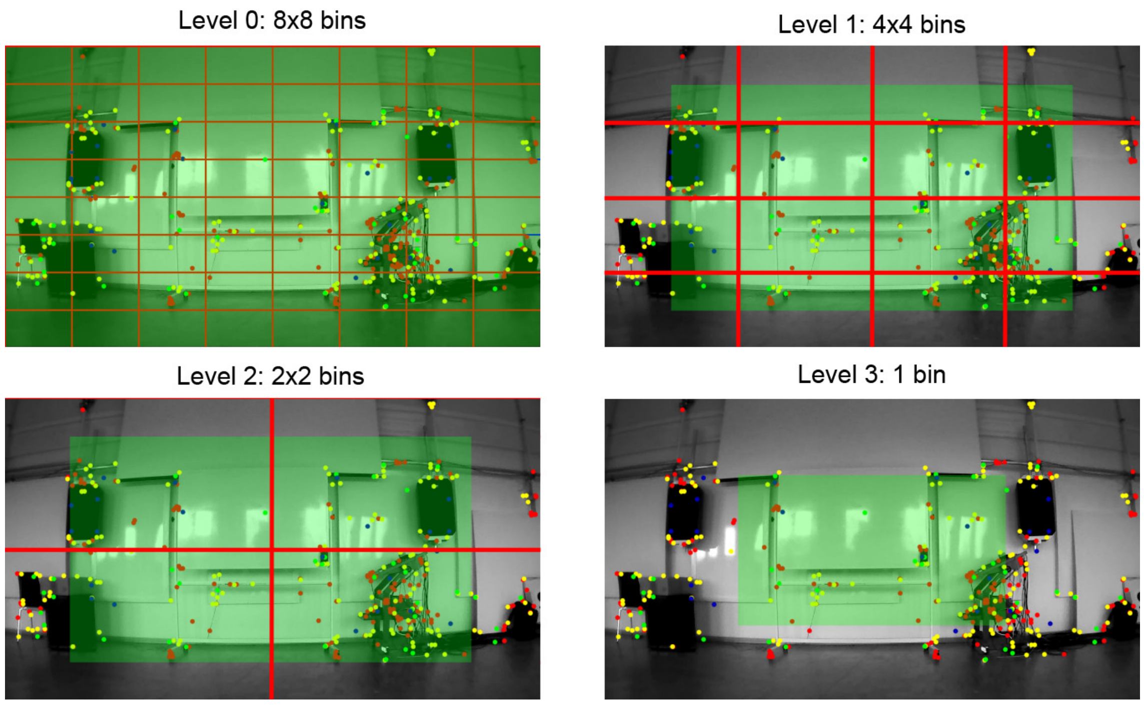


Figure 3.5: "Robust" grid pyramid. The red lines mark the borders of the image bins. The green rectangles mark the areas which are considered for the bin calculation. The higher levels (1 to 3) are calculated from the maximum value of the bins below.

To overcome this problem, we build a "robust" pyramid of the 8x8 bin set as depicted in Figure 3.5. The pyramid is based on maximum values rather than the mean. This leads to a better score for scenes that have good features, which are evenly distributed in the image, but do not cover all of the scene. Furthermore, we disregard the outer regions of the image in the higher pyramid levels. Thus, the localization quality becomes more robust to pose displacements, as they are more likely to survive translational and rotational navigation errors. For our experiments, we have chosen to disregard the outmost ring of bins on level 1 and 2 and an additional ring on level 3. The scores of each pyramid level are then averaged and each level-x-quality represents the localization quality with respect to a different level of feature density and distribution as well as robustness. The final score can be simply obtained by averaging all levels as

$$q_{localization} = \frac{1}{4} \cdot \sum_{i=0}^{3} q_i. \tag{3.14}$$

3.2 Point Generation Likelihood

Another topic which is neglected in current passive MonoSLAM approaches is the fact that not every scene or viewpoint is equally suited for the map generation. Yet again, the current approaches originate mainly from the AR domain and rely on a human user to move the camera in such a way that the system has enough information to produce a reasonable pose estimation and an estimate of the current environment. Most present MonoSLAM approaches rely on some sort of salient features to find correspondences and to be able to perform structure from motion. No matter which salient feature is concerned, the scene as well as the visual projection of the scene (the image) have to contain enough information. Information is not found in equality and repetition, but in change and variation. Especially, this means that large homogeneous regions always pose a problem for feature matching and localization, as they only contain information at their borders. If these homogeneous regions get too large or the camera gets too close, no borders are visible and the image does not contain any information which could be used for map generation or localization. In human-built environments homogeneous regions appear everywhere, e.g. uniformly colored walls or floors. But they are also present in natural environments in the shape of a blue sky, a field covered with snow or just a low textured region in the shade. In order not to lose the visual localization, it has to be ensured that the camera only adopts poses which contain enough useful information.

In this work, we do not assume that all relevant 3D map points are known from the start, as this would require the user to explore the scene by hand in advance and would consume a considerable amount of time for every new scene. Quite contrary, we treat every "mission" as an explorative task. This means that after a short initialization of the system, only a small fraction of the scene is known. Consequently, new points have to be generated, in order to reach the destination without losing the visual localization. As mentioned above, homogeneous regions do not contain enough information for visual localization. This results in the possibility that some paths cannot be taken to reach a destination, or worse, that some destinations cannot be reached without losing the visual localization. In order to find a suitable path to a destination, and determine if along this path enough new points can be generated, it is crucial to have an estimate where to look for new points.

In the general structure from motion approach, the first step is to detect salient 2D feature points, which are used in a later step to establish correspondences across multiple images. We propose to make use of these 2D feature points, as well as the already mapped 3D points, to locate unmapped areas which are well-suited to create new map points. For the generation of potential 3D points two main steps are necessary.

First, one has to separate already mapped areas from the unmapped ones. Given a new keyframe, we found that it is not sufficient to simply find correspondences to other images, and treat all unmatched feature points as potential new points. This approach leads to numerous potential points in regions which are very rich in texture, even if they have already been mapped very thoroughly. To overcome this problem, we propose to use a binning approach to reject points in already mapped areas. Our approach rejects all potential points of a bin if the bin contains more than a threshold t reprojected map points. An example can be seen in Figure 3.6.

Secondly, it is necessary to get a depth estimate for the new potential points, in order to find a suitable viewpoint for the point generation. Without prior knowledge about the topology of the scene, the best guess for the depth of unmapped parts of the scene is given by the already mapped points. One approach would be to simply take the mean or the median depth value of all currently visible map points as a guess for yet unmapped feature points in the image. But in assigning a single depth value to all feature points in the unmapped parts of the image, the system becomes incapable to handle multi depth scenes. For instance, imagine a scenario where 50 percent of the map points lie on a prominent foreground object with a mean depth of 2 meters and the other 50 percent are part of a far away background with a depth of 20 meters. Then this simple approach would

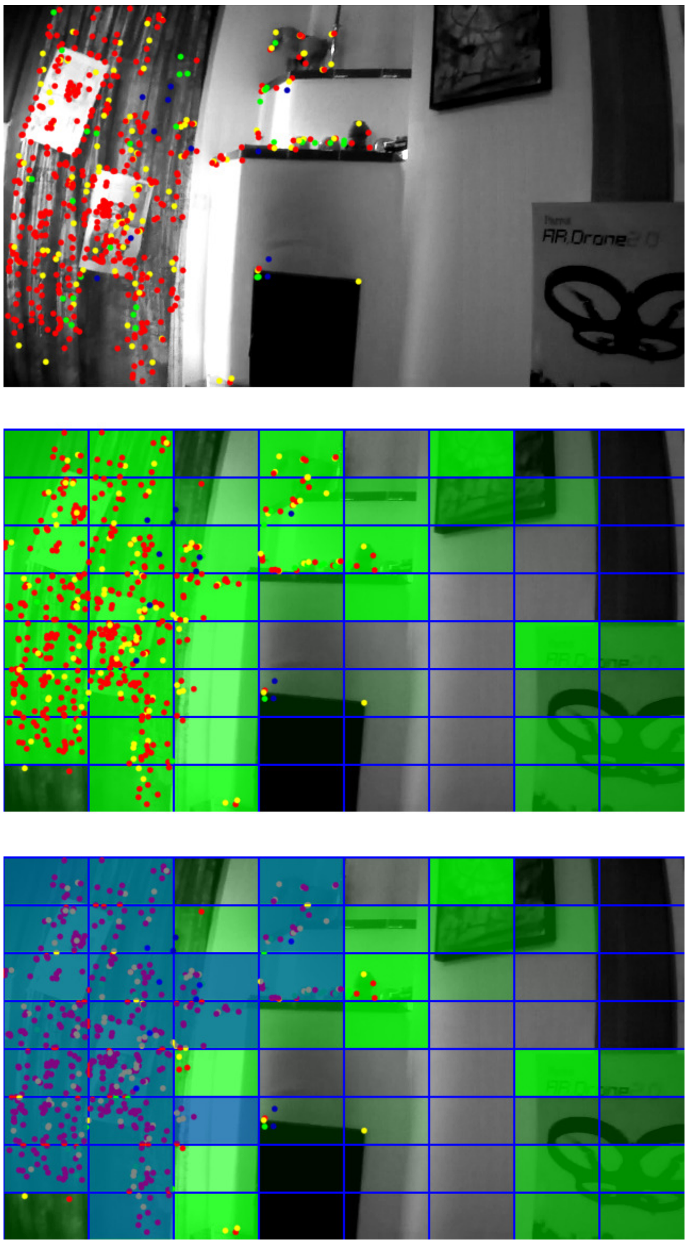


Figure 3.6: 2D feature point rejection scheme for the potential point map generation. Top: Original image from PTAM [37], where the colored dots represent tracked 3D points. Middle: The original image overlain with an 8×8 grid and green fields. The intensity of the green color corresponds to the number of feature points located in the bin. Note that homogeneous regions do not contain any information and therefore no feature points. Bottom: The blue overlays represent bins for which no potential 3D points are projected, as the bin already contains enough map points.

either search for new map points in a depth of 11 meters for the mean value or in a depth of 2 or 20 meters for the median value. The mean depth approach would search for points in thin air 9 meters away from any object, whereas the median depth approach would disregard either the fore- or the background. Both approaches do not use the available information to its full extent.

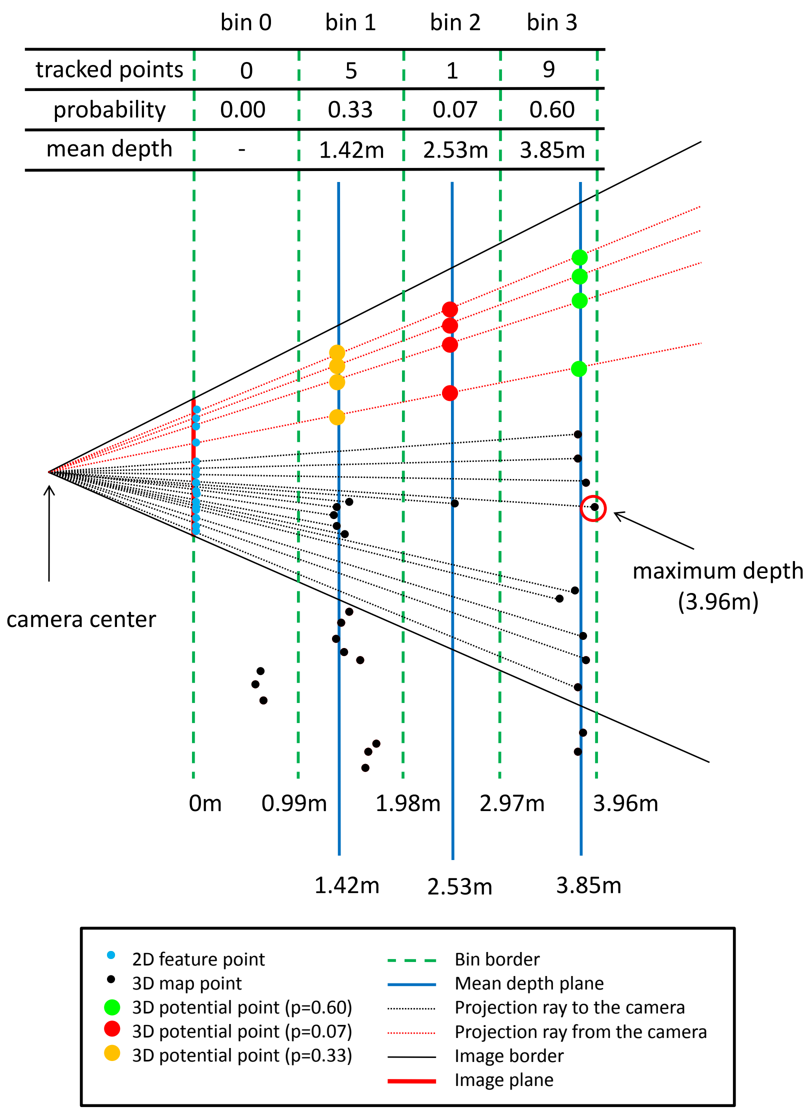
As a solution for this problem, we use the available depth information of the currently tracked map points to estimate a depth probability distribution of the current view. To keep the computational effort manageable, we use a discrete depth approximation scheme. First of all, we generate a histogram of the depth values. Unlike normal histograms, this one has a fixed number of bins, but adapts the value range to the current scene. The maximum value range is set equal to the maximum depth value of the tracked points. Then the thresholds of the histogram bins are set so that they split the maximum range into equal parts. Thus, the histogram automatically adapts to the present scene. To avoid a degeneration of the histogram through far-away outliers, we insert only map points with a sufficiently large triangulation angle.

Then we calculate a probability for each histogram bin according to the percentage of points it contains. Furthermore, we assign a depth value to each bin, which is equal to the mean depth of all points contained in the bin. We call this collection of mean depth and probability values ”variable depth distribution” (VDD). A detailed schematic example of the VDD construction process can be found in Figure 3.7.

Using the set of 2D feature points, the VDD and the pose of the keyframe camera, one can generate a map of potential 3D points. Opposed to a normal point cloud, a point in this map does not only hold information about the location of the point but also its probability of existence. This is achieved through projecting n 3D points for each 2D feature point, where n is the number of bins in the VDD with a point occurrence greater than zero. A real world example can be seen in Figure 3.8.

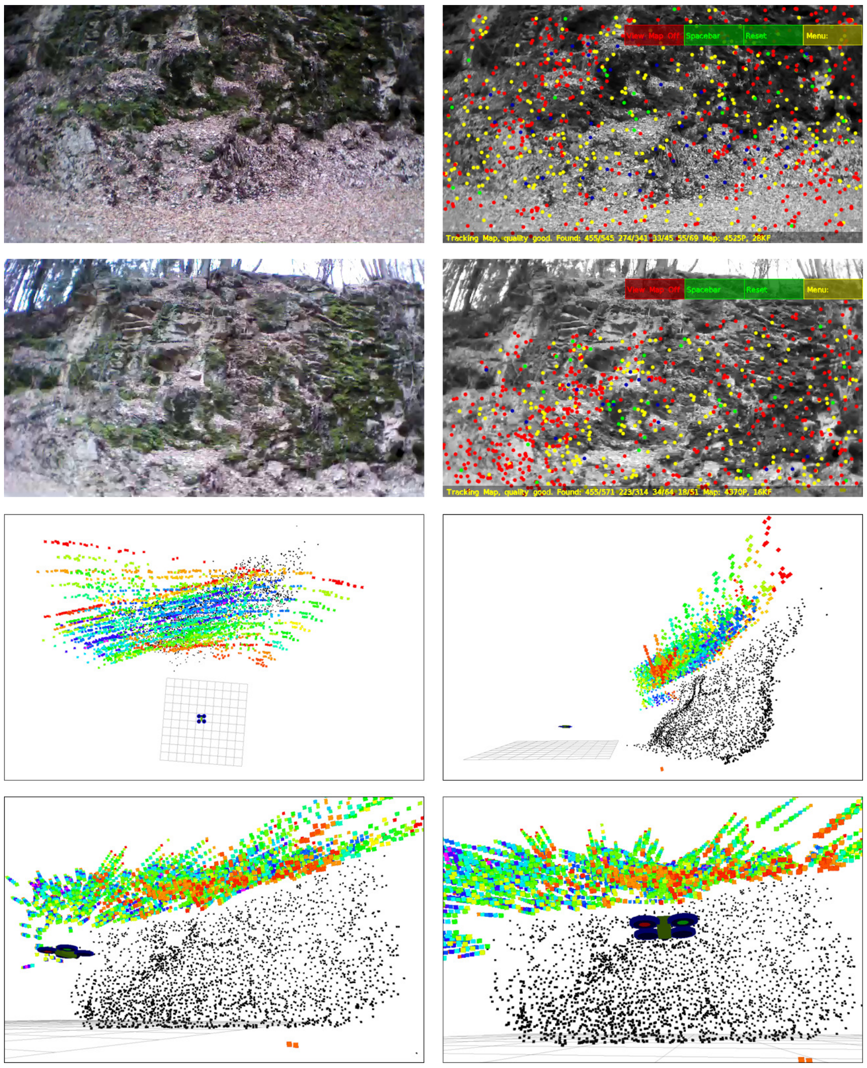
The MonoSLAM framework used for our experiments (PTAM [37]), detects thousands of feature points per frame, if the scene is richly textured. Using all these points would be very costly and result in a very large potential point cloud. Empirically we found that a small random subset of points ($n = 100$) is sufficient to represent areas which are well-suited for point generation. This keeps the amount of potential points reasonably low, while still carrying sufficient information.

During the execution, the potential point map is always kept up to date. Firstly, every new keyframe adds new potential points to the map. Secondly, after each change of the



	bin 0	bin 1	bin 2	bin 3
tracked points	0	5	1	9
probability	0.00	0.33	0.07	0.60
mean depth	-	1.42m	2.53m	3.85m

Figure 3.7: Potential point map generation. **Construction of the variable depth distribution (VDD):** *Step 1:* Of all currently tracked 3D map points find the point with the maximum depth value in relation to the current image plane and set the maximum value range to this value. *Step 2:* Split the depth value range into k equal parts (here: $k = 4$, experiments: $k = 20$). *Step 3:* Assign each tracked 3D map point to one of the k bins according to its depth value. *Step 4:* For each bin calculate the probability of a map point being in this bin. *Step 5:* Calculate the mean depth of all points contained in a bin. **Potential point projection scheme:** *Step 1:* Find suitable 2D feature points. *Step 2:* For each suitable feature point project n potential points. n is the number of bins with a point occurrence greater than zero (here: $n = 3$).



Tracking Map, quality good. Found: 455/545 274/341 33/45 55/69 Map: 4525P, 28KF

Tracking Map, quality good. Found: 455/571 223/314 34/64 18/51 Map: 4370P, 16KF

Figure 3.8: Example of the potential point generation for a natural scene. 1st row: On the left, earlier view of the MAV and, on the right, tracked points of PTAM [37] for the same view. 2nd row: On the left, current view of the MAV and, on the right, tracked points of PTAM [37] for the same view. Note that the upper part of the image is not yet mapped, therefore potential points will be generated for the unmapped part. 3rd and 4th row: Simulated scene from different viewpoints. The black points are already mapped feature points, whereas the colored points are potential points. The color symbolizes the probability of a point being at this location (red stands for a low probability and blue for a high one). Note that for each potential 2D feature point, multiple potential 3D points on a common ray are projected. These potential points are projected on planes parallel to the keyframe they originate from, where the depth of these planes is defined through the VDD.

actual point map, previously inserted potential points can be rejected by reevaluating the bin based proximity criterion depicted in Figure 3.6. Finally, one can use additional information about empty space to further reject potential points. One possible source of extra information is a probabilistic volumetric map, which can be efficiently constructed with the information of the linked point cloud with a framework such as OctoMap [32].

Point Generation Likelihood. The purpose of the point generation likelihood is to evaluate which camera pose of a given set of real or virtual poses has the greatest chance of generating new useful map points. In order to calculate the score, we make use of the probability of existence of the potential points, which is defined through the VDD, as well as the triangulation angle.

We define the score $s_{alpha}(f_p)$ of a potential 3D point f_p based on its triangulation angle between its source keyframe k and the query frame i as

$$s_{alpha}(f_p) = \begin{cases} 0 & \text{if } \alpha_f(k,i) < \alpha_{min} \\ \frac{\alpha_f(k,i)}{\alpha_0} & \text{if } \alpha_f(k,i) \geq \alpha_{min} \text{ and } \alpha_f(k,i) < \alpha_0 \\ 1 - \frac{\alpha_f(k,i)-\alpha_0}{\alpha_1-\alpha_0} & \text{if } \alpha_f(k,i) \geq \alpha_0 \text{ and } \alpha_f(k,i) < \alpha_1 \\ 0 & \text{otherwise} \end{cases}, \tag{3.15}$$

where $\alpha_f(k,i)$ is the triangulation angle as defined in Equation 3.1, and α_{min}, α_0 and α_1 are predefined constants. α_{min} is automatically defined through the properties of the used MonoSLAM approach, whereas α_0 and α_1 are the exact same values as in Equation 3.6 and represent the angle dependent recognition probability. Through this formulation we reward large triangulation angles, as they can improve the quality of the resulting points, up to the point where the feature recognition probability starts to decay.

The score of a single potential point f_p can then be defined as

$$s_{f_p} = s_{alpha}(f_p) \cdot p_{f_p}, \tag{3.16}$$

where p_{f_p} is the probability of existence defined through the related VDD.

Very similar to the calculation of the localization quality, we disregard potential points which lie too close to the image margin ($< \frac{1}{8}$ on each side). This leads to the following formulation for the point generation likelihood score $s_{q,gen}$ for a given query pose q,

$$s_{q,gen} = \sum_{f_p \in F_q} s_{f_p}, \tag{3.17}$$

where F_q is the set of all relevant potential points, which have a 2D projection within the image frame of the query pose q minus the mentioned margin. Note that there is no need to normalize this score, as it is only necessary to find the camera pose with the maximum point generation score in a given set of poses. The matter of how to sample this set of possible camera poses as well as the topic of which subset of potential points is currently relevant for the desired motion direction is described in Section 4.3.

3.3 Collision Probability

For every autonomous system, it is important to achieve navigational safety. To this end, it is crucial to detect obstacles and avoid states which could endanger the environment, the robot, or simply the mission success. Firstly, this section explains how the information of the linked point cloud is used to construct a volumetric occupancy representation of the scene, and secondly, it outlines a way to use this information to estimate the probability of a collision.

3.3.1 Obstacle Representation

Linked point clouds, as used in this work, are very efficient in matters of memory consumption, but are ill-suited for the task of obstacle avoidance as points by definition have no volume, area, or length. One might argue that it is sufficient to check for the closest point and decide from this information alone if a position is dangerous or safe. This approach would have two major drawbacks. Firstly, outliers are always present in structure from motion approaches, even if outlier rejection is applied at every step of the reconstruction chain. Consequently, the lack of robustness against this disturbance is a major problem. Imagine an empty corridor, where the robot should simply move in a straight line along the corridor, but it cannot or is not allowed to, because a single outlier is blocking its way. Secondly, given only this representation, it is impossible to determine which parts of the scene are still "unknown" and which parts are known to be "empty". For an autonomous exploration task, it is vital to memorize this "empty space", so that the robot can safely use this space for further navigation and it is not limited to a user defined space. In contrast, simply avoiding known feature points would be very unsafe, as the robot might collide with an object in an unexplored part of the scene. Both problems can be solved by using a probabilistic volumetric occupancy map.

A volumetric map is a discrete representation, where the world is represented by a 3D grid of cubes, which are also called voxels. Each voxel has an occupancy value which represents the probability of this voxel being part of an object. The occupancy value is defined by all measurements which influence this voxel. In our case, the probability of being part of an object is determined by the linked point cloud and the current position of the MAV.

The only way to increase the occupancy value of a voxel is the measurement of an object. In our case these measurements are the 3D map points of the point cloud, where each point inside a voxel makes it more likely for this voxel to be part of an object. On the other hand, there are two ways to "vote" for a voxel to be empty. Firstly, each voxel in a direct line from the camera to the 3D feature point has to be empty, as otherwise the measurement could not have been obtained. This can be done for all point-camera pairs in the linked point cloud as well as for the current camera pose and the tracked points. Secondly, the space which the MAV traversed has to be empty, as otherwise it could not have passed. The resulting occupancy map can be used to predict the probability of a collision for an arbitrary MAV pose. The remainder of this chapter explains how the collision probability can be modeled to reflect the flight properties of a quadrotor MAV.

3.3.2 Collision Probability

For aerial vehicles such as a quadcopter, one cannot assume a perfect execution of motion commands, not even in the task of simply holding a position. The quadcopter will always oscillate around the desired location as it is a highly dynamic system, even if one does not consider the possibilities of external disturbances such as wind. This behavior has many reasons such as the noise on the pose estimation. Even the rotors of the MAV are sufficient to cause severe turbulences, especially when the MAV is flying very close to the ground. Consequently, it is necessary to keep a safety distance to all obstacles, in order to compensate for the imperfection of the navigational system.

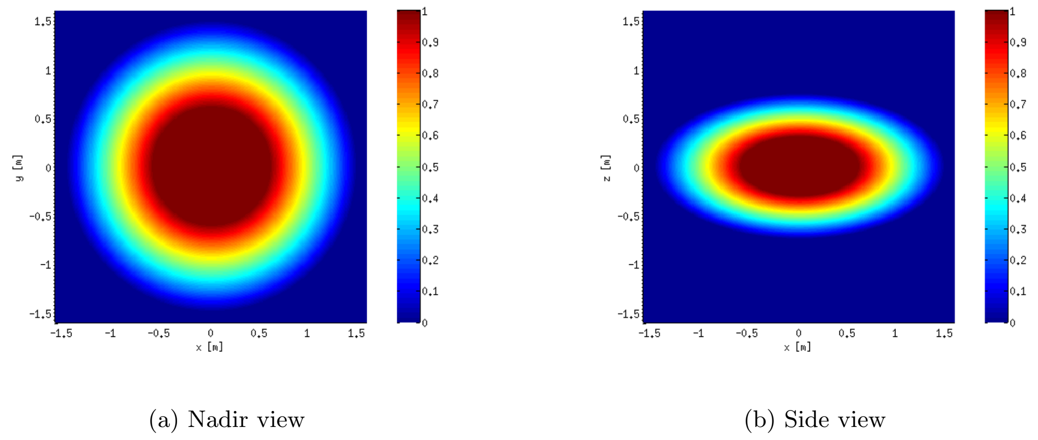
To this end, it is necessary to define the concept of an obstacle in our volumetric representation. In the probabilistic map each voxel is assigned an occupancy value, which represents the probability of this voxel being part of an object. This information is then used, in the general approach, to threshold the map. A voxel can be either classified as "free" if it is below the threshold, as "occupied" if it is above and "unknown" if there has not yet been any information about this specific voxel.

This raises the topic of how to treat "unknown space". The safest way would clearly be to treat each unknown voxel as occupied. In our experiments, we have noticed that this approach is very restrictive, limiting the navigational space of the MAV basically to its initialization space, as using a single camera and sparse features only leads to spatially unevenly sampled rays, which results in a lot of unknown voxels. In order to enlarge the navigational space, we decided to relax the unknown space criterion and treat unknown space and occupied space separately at first.

Handling Unknown Space. Respecting the nature of the volumetric occupancy representation, we decided to consider the unknown space in a cuboidal region around the MAV to ensure its navigational safety. The rotation of the cuboid is not dependent on the MAV rotation, but is aligned with the world coordinate system to speed up the queries. We propose the usage of two cuboids: a tight cuboid with a hard constraint and a larger cuboid with a soft constraint. The collision probability of the tight cuboid is set to the maximum occupancy value of all contained voxels. In contrast, the collision probability of the larger cuboid represents the mean occupancy value of all contained voxels. Thus, we define the collision probability of the unknown space $p_{coll|unknown}$ as the maximum of the two cuboids. In setting the threshold of the accepted collision probability appropriately, it is possible to restrict the navigational space of the MAV to parts of the scene which are mostly known and thus avoid large unknown areas which potentially contain obstacles.

Handling Obstacles. Following the general approach with occupancy maps, we consider voxels above a predefined threshold as part of solid objects. Due to the imperfection in the navigational execution, it is necessary to keep a certain safety distance to obstacles. One approach to achieve the safety distance is to set a hard distance threshold. The main disadvantage of this approach is its sensibility to noise. We have observed great problems if the MAV was very close to the limit, as the pose estimation is always varying a little bit depending on the random selection of map points for tracking at each frame. This led to the behavior of jumping back and forth across the threshold, locking the MAV in place.

To overcome this problem, we added a linear function, which provides a smooth transition between zero collision probability and the obstacle threshold. Opposed to ground robots, MAVs have to consider obstacles in the three dimensional space. Once again, the obvious approach of using simply the Euclidean distance and a threshold is not well-suited for quadrotor MAVs. It was experimentally observed (Section 5.3) that the pose variance in the vertical direction is significantly smaller than the variance parallel to the surface. Consequently, a spherical buffer region does not represent the pose covariance very well. Thus, we decided to use an ellipsoidal safety distance as it is displayed in Figure 3.9.



(a) Nadir view (b) Side view

Figure 3.9: Collision probability field surrounding the MAV from two different views.

Note that this representation does not come with additional costs, but can still be computed the same way as the spherical measure. The only difference is the usage of an ellipsoidal distance measure instead of the Euclidean. This distance measure $d \in \mathbb{R}$ can be obtained by simply scaling the vertical dimension z. This leads to the following formulation.

Let the vector between two points $\boldsymbol{p_1}, \boldsymbol{p_2} \in \mathbb{R}^3$ in the three-dimensional ellipsoidal space be defined as

$$\boldsymbol{v_{12}} = [1, 1, s_{vert}]^T \bullet (\boldsymbol{p_2} - \boldsymbol{p_1}), \tag{3.18}$$

where s_{vert} is the scaling factor for the vertical dimension and $\bullet$ is the symbol for the point wise product operator. As for the Euclidean space, the distance can then be calculated as the square root of the dot product of the relative vector v_{12} with itself

$$d_{12} = \sqrt{\boldsymbol{v_{12}^T} \cdot \boldsymbol{v_{12}}}. \tag{3.19}$$

Using this metric, we then define the collision probability with known objects based on the distance from the MAV center to the closest "occupied" labeled voxel $d_{closest}$ as

$$p_{coll|obstacle} = \begin{cases} 0 & \text{if } d_{closest} > d_{max} \\ \frac{d_{max}-d_{closest}}{d_{max}-d_{min}} & \text{if } d_{closest} \leq d_{max} \text{ and } d_{closest} > d_{min} \\ 1 & \text{otherwise} \end{cases}, \tag{3.20}$$

where d_{min} and d_{max} are constants which represent the navigational uncertainty of the system.

Finally, the collision probability of obstacles $p_{coll|obstacle}$ and unknown space $p_{coll|unknown}$ are then combined by taking the maximum of both values, to obtain the overall collision probability p_{coll}.

3.4 Summary

In this chapter we have outlined three measures which play a vital role in achieving our task of active visual localization and autonomous explorative navigation.

Our first novel measure, the localization quality, can be used to avoid camera poses which are very likely to cause a loss of the visual localization. It incorporates not only the localization uncertainty, but also respects the viewpoint dependent recognition probability of the 3D map points. Furthermore, it implicitly represents the resilience of the camera pose to rotational and translational positioning errors in using a robust grid pyramid.

The second presented novelty, the point generation likelihood, estimates the chance of generating new 3D points in yet unmapped regions of scene from an arbitrary viewpoint. This is achieved through calculating the variable depth distribution (VDD) of the tracked map points of a keyframe. The VDD is then used to discretely sample the probability distribution of potential map points in 3D. This approach can estimate the point generation likelihood without any prior knowledge about the topology of the scene.

Finally, we presented a way of estimating the collision probability for a given MAV pose. This approach is based on a volumetric representation of the scene which only requires the linked point cloud for its construction. The collision probability measure does not only respect the distance to already known objects, but also explicitly handles unknown areas close to the MAV.

This chapter has shown how to measure the most important aspects for active visual localization, whereas the next chapter describes how these measures can be combined to an autonomous explorative navigation system which requires no other exteroceptive sensor but a single camera.

Chapter 4

Autonomous Explorative Navigation

Contents

Every autonomous robotic system has to solve two main problems. Firstly, the system has to find a general plan to achieve a predefined goal. Secondly, the system has to interact with the world to realize this plan. In our case of active visual localization these two problems are strongly interwoven.

In this work the task of the system is to reach an arbitrary destination pose without losing the visual localization. One of the properties that sets this work apart from other navigation approaches is that at the time of initialization the system knows only a very small portion of its environment. Consequently is the system, even in theory, unable to decide whether or not a destination pose can be reached directly after the initialization. This leads to some kind of chicken and egg problem. On the one hand, the system has to move in order to devise a reasonable plan, whereas on the other hand the system cannot move without a plan. To make this task even harder, it is not only necessary to avoid collisions during the navigation, but additionally it is crucial to maintain the visual localization at all times.

In this chapter we describe our solution for this challenging problem by making use of our novel measures described in the previous chapter. We commence the description of our approach, which we call ”autonomous explorative navigation”, with an explanation of its basic navigation scheme in Section 4.1. This is followed by description of our modular system architecture in Section 4.2. In Section 4.3 we then explain how we logically combine our novel measures in a local planning scheme to an autonomous navigation system. We conclude this chapter with a summary of our autonomous explorative navigation approach in Section 4.4.

4.1 Destination-Based Navigation

The most popular approach of controlling a ground robot is to send a motion command in the shape of a vector of velocities. The ground robot then executes the command with an uncertainty which depends on the robot type as well as the type and topology of the floor. Unfortunately, this is totally infeasible for multirotor MAVs.

The main reason for this is that a multirotor MAV, opposed to a ground robot, is a highly dynamic and unstable system. Consequently, it is necessary to use a controller to achieve even a simple task such as hovering. For this reason, most multirotor MAVs, such as the AR.Drone or the AscTec Pelican, are equipped with onboard attitude and height controllers to facilitate the controlling interface. Opposed to ground robots, multirotor MAVs are even with these low-level controllers still not controllable by velocities, but rather take a vector of target angles for the attitude control as input (except for the vertical direction). Sending such a motion command does not result in a fixed velocity of the MAV but in a nonconstant acceleration. Even if one would thoroughly model the complex system of the MAV, it would still be very hard to predict the exact MAV pose in the future due to the uncertainties of the onboard controllers and external influences such as wind.

Luckily, recent advances in the field of visual servoing [1, 18, 19, 36, 78, 80] present a new way of controlling a multirotor MAV. These ”high-level” controllers make it possible to define a destination pose in world coordinates instead of a vector of angles and velocities. Through this type of controlling it is now possible to navigate the MAV along piece-wise linear trajectories.

In this work we make use of this destination-based type of navigation. To allow for a seamless realization of our plans, our planning logic does not generate arbitrary paths, but instead generates destination poses. During the planning process our approach takes the resulting trajectory to a destination pose into account, but the visual controller never has to consider anything but a single destination pose at every point in time.

This type of navigation has one major advantage. The fact that we only have to provide the visual controller with new destination poses instead of motion commands drastically softens the real-time constraint of the planning logic. As the controller takes over the task of outputting motion commands multiple times per seconds, it is now possible to spend a significantly longer time on finding the best plan available. The resulting plans can then be transmitted to the controller as simple destination poses. In the following section we will outline all modules of our system architecture and describe the communication between them.

4.2 System Architecture

The system architecture proposed in this work has several appealing properties. Aside from providing the necessary functionality for a working system, the design offers a high degree of modularity. In total our system consists of five independent modules as depicted in Figure 4.1. In our implementation each module runs in its own process and they communicate via messages using the ROS framework[6]. The advantage of this approach is two-fold. Firstly, in using multiple processes we can make use of physical parallelism provided by currently available CPUs. Secondly, the modularity divides the functionality in independent parts. This makes it possible to replace modules with minimal effort. For example, if in the near future we want to use a different type of MAV we can simply replace the MAV driver module and do not necessarily have to alter the other modules. The same is true for any of the other modules. In the following paragraphs we will shortly outline each module.

MAV. The MAV module consists of the physical multirotor MAV itself as well as the corresponding driver. The MAV has to be equipped with a camera and be able to stream the image data to the base station, so that this data is available for the other modules. Further, we assume that the MAV can be controlled by a four dimensional motion command, e.g. (*yaw*/*pitch*/*roll*/*thrust*). The IMU data can be seen as a supplementary feature of the MAV as all our modules are fully functional without this additional information. The only difference is that without the IMU data the system has no way to recover in the case of a loss of the visual localization. In our experiments we used a Parrot AR.Drone 2.0 as robotic component.

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

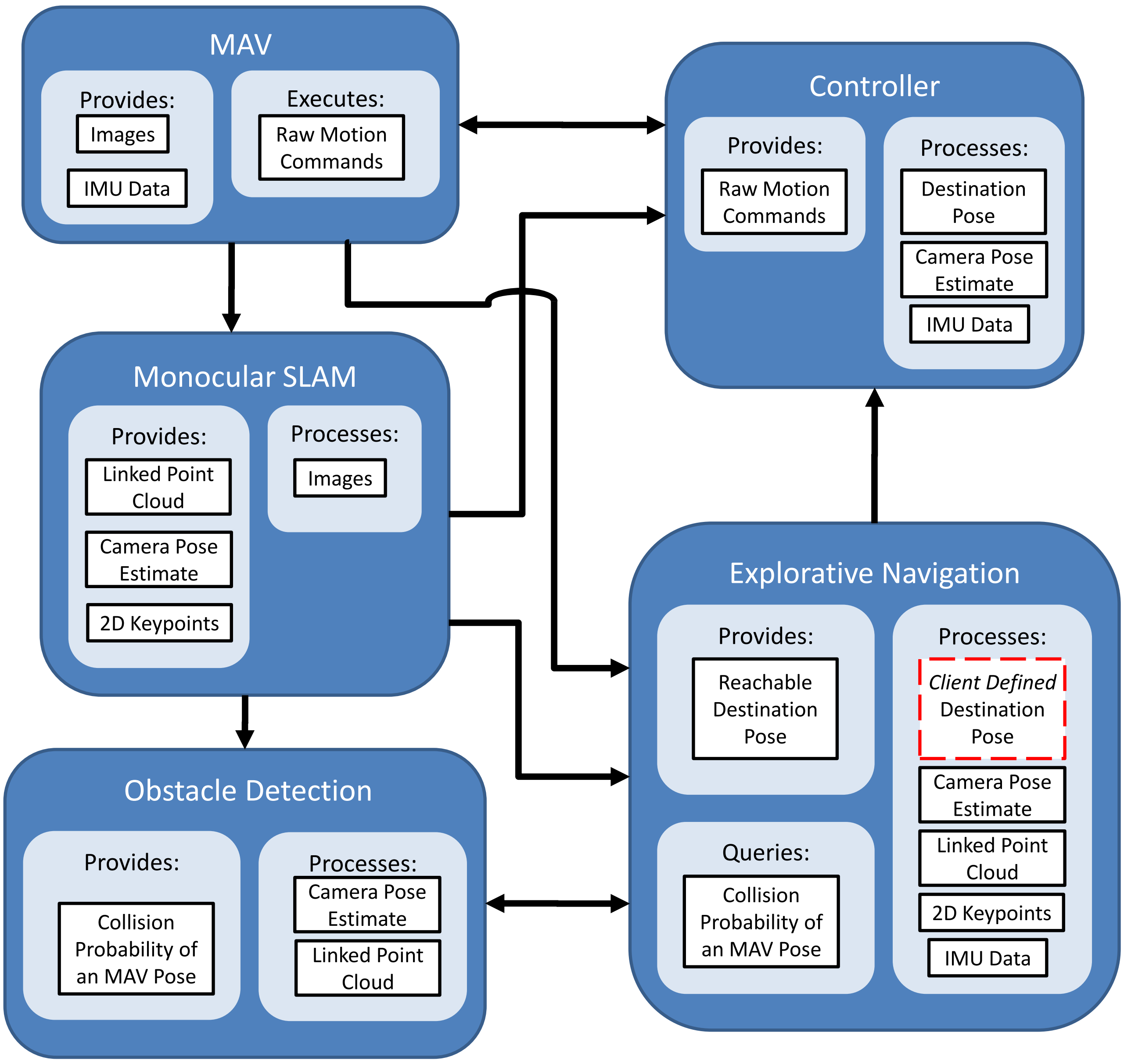


Figure 4.1: System architecture. Each block presents a module of our overall system and the arrows visualize the information flow between them. The only information which is not generated by our system is the "*client defined* destination pose" which is marked with a red dashed border and defines the high-level goal of the system.

Monocular SLAM. The purpose of the monocular SLAM module is two-fold. Firstly, it should create a sparse reconstruction of the environment based on keypoints and, secondly, it should estimate the current pose of the camera relative to the sparse reconstruction. To achieve this task the module should not require anything but the raw images from the camera. In our experiments, we used a slightly modified version of PTAM [37] which provides all the necessary data.

Obstacle Detection. We assume that the obstacle detection module can create a sufficiently accurate estimation of free and occupied space only based on the information of the linked point cloud. Based on this data only, it should provide the functionality which enables other modules to query the collision probability of an arbitrary MAV pose. The obstacle detection module in our experiments uses the Octomap framework [32] for the generation of a volumetric representation of the scene.

Controller. The purpose of the controller is to send raw motion commands to the MAV such that the MAV moves towards a destination pose. As a multirotor MAV is an unstable system, the controller needs a high frequency pose estimate as input. In our experiments we used the fuzzy control logic of Katusic [36] as control module.

Explorative Navigation. This module contains the entire "high-level" logic of the system. The module takes arbitrary *client defined* destination poses as input which define the high-level goal of the system. During the navigation phase the logic evaluates all available data in order to find the "best" possible move for the current situation. This move is then transmitted to the controller in form of a reachable destination pose. In the following section we provide a detailed description of how our system evaluates the current state of the world and uses our novel measures to find the next best possible move in a local planning approach.

4.3 Planning Logic

For most ground robots the path planning task can be reduced to a simple two-dimensional problem for which it is possible to find an optimal solution in an acceptable amount of time. In contrast, a monocular MAV system has to consider at least two more dimensions; the vertical dimension and the horizontal rotation (yaw). To make the problem even more complex, the visual localization adds an additional reprojectional requirement. This means that it is not only necessary to consider the pose of the MAV with respect to obstacles and unknown parts of the scene, but additionally consider the projections of our virtual map at any point in time. The complexity of the problem makes it computationally infeasible to find a globally optimal solution for the path-planning problem with currently available algorithms within a real-time setting such as ours. Additionally, the fact that most of the scene is still unexplored after start up, motivated us to take a local planning approach rather than a global one.

During the autonomous navigation our system has to fulfill a range of requirements simultaneously. First of all, the MAV should close the distance to a predefined destination pose. Secondly, it should keep a safe distance to any obstacles to avoid collisions. Additionally, the system is required to maintain the visual localization at all times. Without the localization the system is basically "blind" and can neither properly avoid obstacles nor reach the destination pose. In order to ensure a localization-safe navigation it might be necessary to generate new useful map points. To ensure that all requirements are fulfilled simultaneously we designed our operation logic in a hierarchical fashion. Our system can be seen as a state machine, but each state has its own position in the hierarchy depending on its importance for the overall safety of the system as shown in Figure 4.2. Further on, we will call the "states" of the state machine "modes" to avoid a confusion with the term "system safety state" which is concerned with the physical state of the MAV in the real world.

We can split the space of the system modes into two logical parts; the modes of explorative navigation and emergency modes. The modes of the first type want to achieve a high-level goal and do not have any hard real-time constraints. In contrast, if the system enters an emergency mode this means that some of the basic needs of the system are violated and need immediate attention. While we respect the basic needs of the system for collision avoidance and localization during our planning stage, external influences can always lead to an unpredictable violation of these needs. Therefore we designed our system in such a way that a mode of high importance can interrupt a mode with a lower importance at any point in time.

4.3.1 Explorative Navigation Logic

The purpose of our explorative navigation module is to safely reach a predefined destination pose without losing the visual localization. Who or what defines these so-called *client defined* destinations does not matter. In our experiments the destinations were provided

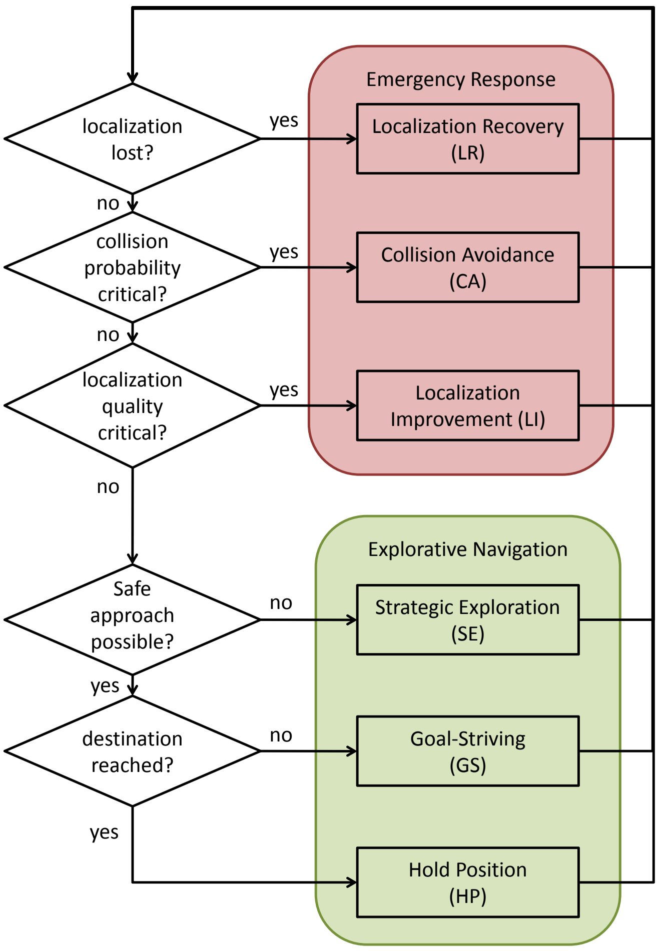


Figure 4.2: Diagram of the basic system logic and the corresponding system modes. The system modes can be categorized into two types of planning logic; the emergency response logic and the explorative navigation logic. If the system enters an emergency mode this means that one of the basic needs of the system is not satisfied and needs immediate attention. The order in the decision tree can be seen as a ranking of the modes by their importance. This order defines which mode can be interrupted by another mode. E.g. the *Localization Recovery* mode has the highest importance and cannot be interrupted at all, whereas the *Hold Position* mode can be interrupted by any of the other modes.

by the human user, but they could as easily originate from a next-best view algorithm.

As our system uses the definition of a destination pose to control the motion of the MAV, the straight-forward approach would be to simply send the *client defined* destination to the controller. This might even work if this destination is good-natured, the scene has been sufficiently explored prior to the mission and the scene does not contain any obstacles. In any other case, this approach will result in a mission failure in the best case and damaged property or persons in the worst case. Consequently, we do not take any of the three properties above for granted.

To ensure localizational as well as navigational safety, our system generates additional destinations which enable us to navigate on arbitrary piece-wise linear trajectories towards a *client defined* destination. These system generated destinations do not only prevent the MAV from entering dangerous situations, but also ensure that the MAV has sufficient free space for a safe navigation and enough map points for a stable localization.

In the remainder of this subsection we explain how we generate temporary destinations to realize an autonomous explorative navigation system. We structure our explanation according to the three system modes of the explorative navigation and move from the mode with the lowest priority to the mode with the highest. We show under which circumstances a certain mode is entered, explain the purpose of each mode and how we fulfill this purpose through the intelligent generation of reachable destinations.

4.3.1.1 Hold Position (HP)

The HP mode is the most basic mode of our system. We only enter this mode after successfully reaching a destination. The purpose of the HP mode is to keep the MAV safely hovering at the destination pose. On the one hand, this mode gives the system a purpose after completing the mission and, on the other hand, it relaxes the real-time constraint for planning as it ensures that the MAV stays at the destination pose while the system is generating a new plan. As the MAV is already at the desired location when this mode is entered, the original destination is send to the controller. Note that if for some reason the MAV strays too far from the destination, e.g. through the influence of wind or a human pushing the MAV, this causes the system to leave this mode and invoke our goal-striving navigation.

4.3.1.2 Goal-Striving (GS)

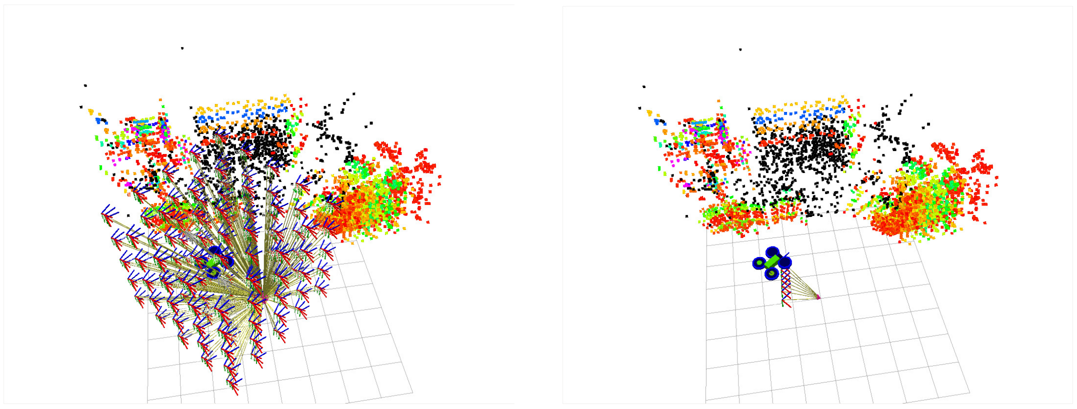
The main purpose of the GS mode is to close the distance to a *client defined* destination and thus complete the mission. Therefore, this mode holds the basic planning logic of our approach.

The goal of our approach is to find a safe path towards a *client defined* destination on which the visual localization is always maintained. As these *client defined* destinations are in general not directly reachable, it is necessary to generate, what we call, *intermediate* destinations. These *intermediate* destinations can be regarded as some kind of safe step stones towards a *client defined* destination and enable us to navigate on arbitrary piece-wise linear trajectories. During the planning phase we analyze the current state of the world and choose the best available pose as an *intermediate* destination. This pose is then transmitted to the controller for execution. During the execution phase it is very likely that new map points will be generated, which may lead to the case that a different destination is now the better choice. For this reason we do not wait for an *intermediate* pose to be reached before looking for a new destination, but replan after a fixed amount of time (10 seconds in our experiments).

This leads to the question: What is the best pose to choose? The answer to this question strongly depends on what someone wants the MAV to achieve. In our case, we want the system to move safely towards a destination pose while avoiding a loss of the visual localization. This means that, aside from two safety aspects, the best pose to choose is clearly the one that minimizes the distance to the *client defined* destination. The distance between two points in a three dimensional space is well defined. In our case however, we additionally have to consider the rotation of the MAV (yaw). Mathematically, this additional value cannot be simply treated as a fourth dimension. Aside from a unit mismatch between radiant and meter, we also have to be aware that the rotation is only defined on the interval $]-\pi, \pi]$ whereas all other dimensions are defined on $]-\infty, \infty[$.

In our case, it is not necessary to know the actual distance value in meter or radiant, what we are really interested in is the cost to perform a certain motion. If we know the difference in the navigational speed between a rotational motion and a linear motion, we can use the ratio to transform them into a common four dimensional cost space. In this cost space we can simply calculate the Euclidean distance between two poses. Given a fixed set of possible destination poses, we can now decide which of those poses is the "closest" to the *client defined* destination. The next paragraph describes how we reduce the set of possible destination poses to a manageable size.

Reducing the Search Space. In a first step of our local planning scheme, we aim to reduce the four-dimensional continuous search space to a fixed sized set of destination poses. First of all, we check if the *client defined* destination itself is reachable without any problems. In this case there is no need for any further efforts as we already found the optimal solution. In any other case, we sample MAV poses in a fixed interval around the current MAV pose. In our experiments, we sample the poses in a regular grid with intervals of 30 cm up to a distance of 1.2 m and the rotation in steps of 0.2 rad up to 0.6 rad. This leads to a total number of 5103 sampled poses. In Figure 4.3a we display an example of the sampling scheme with a reduced number of poses for visualization. Additionally to this set of regularly sampled poses we add the closest reachable pose on a direct trajectory to the *client defined* destination to the set of destination poses. Note that all possible destination poses should be located sufficiently far away from the current MAV pose. If they are sampled too close to the MAV, this can lead to a tediously slow navigation speed when the MAV comes close to losing the visual localization.



(a) Regularly sampled possible destination poses.

(b) Regularly sampled trajectory.

Figure 4.3: Sampling scheme for local path planning. (a) In a first step we regularly sample possible MAV poses around the current MAV pose. (b) After finding the best available destination pose for a certain task, we also sample the trajectory to the best destination pose to check for a violation of our safety constraints along the path. The black dots are already mapped 3D points, whereas the colored dots represent potential map points. Note that in this figure we actually displayed the sampled camera poses rather than the MAV poses, because they are more important for visual path-planning than MAV poses.

Finding the Optimal Destination. In the next step we have to find the optimal destination in the given discrete set of destination poses. Therefore, we first evaluate the estimated collision probability of each pose in the set and use it to reject dangerous poses with a fixed threshold. Then we estimate the localization quality for the remaining set of poses and reject poses with a poor localization quality in a similar fashion. Thus, we receive a subset of safe poses with a good localization quality. In this subset the optimal pose for the task of moving towards a destination, is the pose which minimizes the distance to this destination.

Up to this point we neglected one important property of the destination poses; the reachability. This means even if we found the optimal destination in the previous step, it still might be that this destination is not reachable in the sense of collision avoidance or localization maintenance. We deferred the evaluation of this property up to this point, because it does not come very cheaply. In order to maintain the visual localization it is necessary to check whether the localization quality is high enough at each point along the trajectory to the destination. To this end, we sample the estimated MAV trajectory in short intervals (10cm in our experiments) and evaluate the localization quality as well as the collision probability for each sampled pose. An example of this trajectory sampling scheme is shown in Figure 4.3b. If both values are far from critical for each sampled pose along the trajectory, we have found the optimal reachable destination. In the opposite case, we retreat to the previous step, select the next best destination and check this reachability constraint again. This procedure is repeated until we find a pose, or until no more useful poses are in the set of possible destinations. In our experiments we treat a destination pose as useful if its collision probability is low enough, the localization quality high enough, and it is closer to the *client defined* destination than the current MAV pose. If we are unable to find a useful destination, our system switches into its explorative mode so as to explore unknown parts of the scene which might be used for a localization-safe navigation.

4.3.1.3 Strategic Exploration (SE)

As our only exteroceptive sensor is a single camera, the field of perception of our MAV is significantly smaller compared to other robotic setups which are often equipped with laser scanners. Furthermore, our monocular approach needs to generate a reasonable baseline between the keyframes to receive an accurate depth estimation through structure from motion. The aim of this system mode is to overcome these limitations through the generation of *strategic* destinations which provide the necessary baseline to generate new useful map points.

In our approach we consider two types of *strategic* destinations. The purpose of the first type is to ensure that enough map points are available for the further navigation through the active generation of new map points. The second type, on the other hand, is not concerned with localization-safety but with the navigational safety. Its purpose is to ensure that the way towards a *client defined* destination is free of obstacles in actively carving free space. The following two paragraphs will explain the two types of destinations in more detail.

Active Map Generation. In general, the monocular mapping system (PTAM [37]) generates new map points passively. This means that while the MAV is moving, the system automatically builds a map through the triangulation of feature points. In order to get a reasonable depth estimate for the map points, it is crucial that the images used for the triangulation were taken from different viewpoints to provide a sufficiently large triangulation angle. The closer the viewpoints are to each other, the smaller the triangulation angle becomes and the higher the depth uncertainty of the resulting map point rises. This means that especially a pure rotational motion, which does not provide any triangulation angle at all, is ill-suited for the map generation. Through our local planning scheme we are able to detect situations in which it is necessary to generate new ”useful” map points. In the following we describe how we actively search for map points in relevant regions of the scene.

As described in the previous chapter, we estimate a distribution of potential map points additional to the normal map points. Using this distribution of potential map points we are able to calculate the likelihood of generating new points from an arbitrary MAV pose. But to ensure that we generate ”useful” new map points it is necessary to decide in which direction it is necessary to generate new map points.

For this purpose, we first simulate the motion of the MAV if we would directly send the *client defined* destination to the controller. Then we estimate at which point along this trajectory the visual localization will very likely get lost. We use the pose at this point to determine which subset of the potential points is relevant for the further navigation. This means that we only considered potential points, whose projections lie inside the image frame of this pose. To avoid being overly restrictive we enlarge the image frame for the reprojection task by a constant factor k_e (we used $k_e = 1.2$ in our experiments), which leads to a greater field of perception. This strategy leads to a ”useful” subset of potential points. The purpose of generating new useful map points can now be fulfilled by finding a pose which maximizes the point generation likelihood of this ”useful” subset.

To find the optimal destination pose we use the same local planning approach as in the *Goal-Striving* mode with a minor change. Instead of the distance to the *client defined* destination, we now optimize the point generation likelihood of the destination poses while still respecting the localization quality and the collision probability.

Of course, there remains the possibility that the ”useful” subset is empty if there are simply no suitable features available on the direct path to the *client defined* destination. In this case we force the MAV from this path by sending it on an exploration mission by using the whole set of potential points instead of a subset.

In practice, it is not always necessary to reach this *strategic* pose entirely, because the system also generates new map points on the way to this pose. As these points might already be enough for a localization safe navigation, we limit the validity of the *strategic* destination. In our case, this means that the system tries to reach this pose for a considerable amount of time, but after failing to do so in the given time frame, it can ignore this pose and return to its previous task.

Active Free Space Carving. To ensure the navigational safety of our system, we only allow the MAV to pass through parts of the scene which very likely do not contain any obstacles; i.e. free space. In order to grant our system enough free space to navigate towards a *client defined* destination, we use a simple but effective strategy. After receiving a *client defined* destination we want the MAV to look in the direction of the newly received destination. This procedure reflects very much the behavior of a human in an unknown environment. If you tell someone to move to a certain position in e.g. a corridor, normally the person will first take a look towards the specified position before actually starting to move. In our case we make use of this behavior to ensure that no obstacles are blocking our way. To achieve this, we define a destination pose at the current xyz-position of the MAV, which is rotated such that the camera is directly looking at the *client defined* destination. Then we treat this if this newly generated destination as a *client defined* destination, which invokes our *Goal-Striving* navigation. While moving towards this newly generated destination, the system automatically generates map points in the direction of the *client defined* destination through our local planning scheme and the active map generation described in the previous paragraph. The visibility information of the newly generated map points can be used to carve a corridor of free space in the direction of the *client defined* destination. The MAV can then use this free space for a safe navigation.

4.3.2 Emergency Response Logic

Although we respect the matter of navigational safety and localization maintenance in our planning phase, it is highly likely that due to external disturbances some of the safety constraints will be violated during the execution. In order to avert imminent danger, it is important to react as fast as possible to an approach to a dangerous situation. The more time is wasted before reacting, the more likely it becomes that either the mission fails or, even worse, a collision might occur. Therefore, we check the safety constraints in a fixed interval and interrupt the explorative navigation logic in the case of a violation. To resolve the safety problems we generate temporary *emergency* destinations. Opposed to the other types of destinations, the purpose of an *emergency* destination is not so much to pull the MAV towards a certain destination but rather to drag it away from a dangerous situation. When the system state is sufficiently safe again, we resume with the explorative navigation.

In this work we distinguish in total three emergency scenarios, where each scenario is linked to a specific emergency mode.

4.3.2.1 Collision Avoidance (CA)

In this first emergency scenario the MAV is physically in a dangerous situation. This means that it moved either too close to a known obstacle or unknown parts of the scene which might contain obstacles. To detect such situations we use our novel collision probability measure. In our experiments we treat this physical safety constraint as violated if the collision probability rises above a value of 0.4. In this case the system enters the *Collision Avoidance* mode and tries to secure the physical position of the MAV, while still maintaining the visual localization.

As it is crucial in such a situation to react immediately, we do not invoke a complex planning procedure, but instead choose the closest keyframe pose which is sufficiently safe (collision probability < 0.3 in our experiments). This strategy has two nice properties. First of all, it is very fast because we reduce the search space to the set of keyframes. Secondly, the motion towards our *emergency* destination is very likely to be safe, not only with respect to collisions, but also in regard to localization maintenance. As we sample keyframes in fixed intervals, choosing the closest safe keyframe pose as a destination will, more probably than not, cause the MAV to retreat a short distance along the path that led to this emergency situation. In practice, this emergency behavior has prevailed over other more sophisticated and "in theory" safer approaches, because the reaction speed seems to be the most important factor for collision avoidance.

4.3.2.2 Localization Improvement (LI)

In this emergency scenario the system enters a state with an increased chance of losing the visual localization. Although a loss of the localization does not directly endanger the MAV or its physical surroundings, it basically renders our system "blind". To detect such a situation we use our novel localization quality measure, which was specifically designed for this purpose.

In the case that localization quality reaches a critical level (below 0.35 in our experiments), our system enters the *Localization Improvement* mode. In this mode the system strives to improve the stability of the localization while it concurrently takes care to keep the collision probability reasonably low.

As it is not necessary to prevent an imminent collision, it is possible to invest a bit more time in the destination generation. But to keep the computational time reasonably low, we once again restrict the search space to the current set of keyframe poses. Opposed to the *Collision Avoidance* mode, we do not only respect the collision probability at the destination, but additionally ensure that the collision probability is reasonably low along the whole simulated MAV trajectory. Thus, we choose the closest keyframe pose which has a sufficiently high localization quality (> 0.4 in our experiments) and is safely reachable without any *intermediate* destinations.

4.3.2.3 Localization Recovery (LR)

Losing the visual localization can be seen as the worst case scenario for our system. In this case the system is absolutely "blind" and can only estimate its own motion with its inertial sensors. Unfortunately, the inertial sensors are in general quite inaccurate and are prone to drift over time. Consequently, we can trust the pose estimate with inertial sensors only for a short period of time (10 seconds in our experiments).

Our system considers the visual localization lost if the time since the last update has exceeded a certain threshold (1 second in our experiments). Under normal conditions our planning scheme prevents this scenario from occurring, but in a real world experiment external disturbances and network lags can always move the MAV far from its desired location.

In this scenario we do not to waste any valuable time and immediately set the closest keyframe pose as a destination. Then we assume that inertial sensors are accurate enough for this short time navigation so that the system is able to restore the visual localization. If the system is not able to recover in the given time frame, we simply send hover commands to the MAV as we can no longer trust our pose estimate due to the considerable drift of the inertial sensors. In this case a human operator can manually return the MAV to a safe location where the visual localization can be restored.

4.4 Summary

In this chapter we have presented our novel explorative navigation approach. The proposed navigation scheme does not only respect the physical constraints with regard to collision avoidance, but also strives to maintain the visual localization at all times. Considering the visual localization during planning is a very complex task, which requires the system to respect the reprojection of the already mapped points along a continuous four dimensional trajectory. With our local planning approach we present a way to drastically reduce the search space which enables visual path planning within the tight boundaries of a real-time setting. Opposed to conventional navigation approaches which assume the localization to be given at all times, our approach is able to detect the need for a localization improvement using our localization quality measure. Through evaluation of the point generation likelihood it is then possible to explore yet unknown parts of the scene. In simulating the motion of the MAV we make sure that we only explore parts of the scene which are relevant for further navigation. This leads to a navigation approach which is fully capable to decide which parts of the scene need to be explored to ensure the visual localization. The resulting navigation system can autonomously move through arbitrary unknown environments while avoiding collisions and maintaining the visual localization at all times.

In the next chapter we provide the details of our experimental setup and experimentally prove the capabilities of our monocular navigation approach.

Chapter 5

Experiments

Contents

Every complex robotic system is a combination of many hardware and software modules, which can have many abstraction layers. Consequently, the performance of the overall system strongly depends on the performance of each module of the system. Thus, we decided to analyze the most important parts of the system individually with the least possible number of dependencies on other modules. After outlining our experimental system setup, we present our experiments in four parts.

We start off our experiments with a thorough analysis of the viewpoint dependency of the localization system in Section 5.2. More specifically, the section analyzes the feature recognition probability of the PTAM system [37] in relation to viewing angle and scale changes. In this experiment we demonstrate that, even under ideal settings, the feature recognition probability declines if the difference between the original and the current viewpoint increases. This information can then be used to improve the estimation of the localization quality as described in Section 3.1.2.

The second experiment described in Section 5.3 analyzes the performance of the visual controller of Katusic et al. [36], which was used as the navigation module in all our experiments. In this experiment we use a set of simple trajectories to infer information about the navigational performance of the system. We do not only analyze the response time of the system, but also investigate the deviation from the ideal trajectory. Through this experiment we demonstrate that an ellipsoidal metric as proposed in this work leads to a more realistic model for the collision probability than a simple Euclidean metric. Furthermore, we describe how the newly obtained knowledge about the navigational imperfection of the system can be used to improve the safety and performance of the overall system.

In our third experiment (Section 5.4) we evaluate the performance of the localization quality measure in a controlled environment. Through this experiment, we demonstrate that this novel measure is well suited for detecting states which are very likely to result in a loss of the visual localization.

Finally, Section 5.5 analyzes the performance of the overall system in two challenging scenarios. In the first scenario we show that our monocular system can autonomously perform a full 360° turn in a barely textured scene. Through this experiment we do not only demonstrate that our localization quality measure can be used to prevent states which might result in a loss of the visual localization, but also show that our point generation likelihood measure can be used to generate new useful map points in an intelligent fashion. In the second scenario we let the MAV autonomously navigate in a complex and narrow scene which contains multiple floor levels. In this challenging scene we show that the system is able autonomously find a safe path in an unknown environment and can even manage a safe flight across a narrow staircase without losing the visual localization. Aside from simply proving our concept in these challenging scenes, we also use this experiment to evaluate the localization and mapping performance of our system.

5.1 System Setup

Our general system setup consists merely of three components as shown in Figure 5.1. We can classify the three parts into an active component (MAV), a computational component (notebook) and a human interface (notebook and/or gamepad). In our experiments we used a Parrot AR.Drone2.0 as an active component, but for our implementation the type of MAV hardly matters as long as it fulfills a few requirements. Firstly, it should be a multirotor MAV which can be controlled with a four-dimensional vector (e.g. velocities in three dimensions and one rotational axis). Secondly, the MAV should have the ability to communicate with a base station such as a notebook. Thirdly, the MAV should be equipped with inertial sensors and a camera. Last but not least, our system requires the IMU data and the camera images to be streamed in real-time to the base station for control.

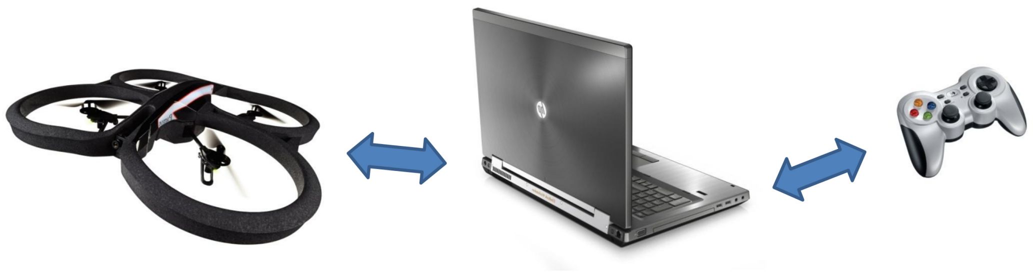


Figure 5.1: General system setup. The setup consists of three main parts: The most important part is the MAV, for which we used a Parrot AR.Drone2.0 in our experiments. The second part is a preferably mobile computer which is able to establish a Wi-Fi connection to the MAV. The last part of our setup is a gamepad, which allows the user to take over in the case of an emergency.

In our experiments we used a notebook of the type *HP EliteBook 8570w* as the computational unit. It consists of a 2.4 GHz quadcore processor of the type Intel CORE i7, a memory module of 8GB DDR3 RAM, a 750GB HDD and an NVIDIA Quadro K2000M graphics card with 2GB RAM. Of course this component can be replaced by any CUDA-capable system with comparable computational power.

The last component is a human interface, such as a keyboard or a gamepad. The main purpose of this component is to allow the user to intercept the autonomous execution in the event of an emergency. In our experiments we used a wireless gamepad to offer the user an increased degree of mobility during the execution opposed to a bulky keyboard.

5.1.1 AR.Drone 2.0

All our experiments were conducted with an unmodified model of the Parrot AR.Drone 2.0 as active system component. It is a quadrotor MAV which was primarily developed as a toy to be used in combination with a smart phone. Due to the availability of an SDK, the drone can be controlled by any device which can establish a Wi-Fi-connection. Via this wireless link the AR.Drone receives the control commands and streams back the images and IMU data.

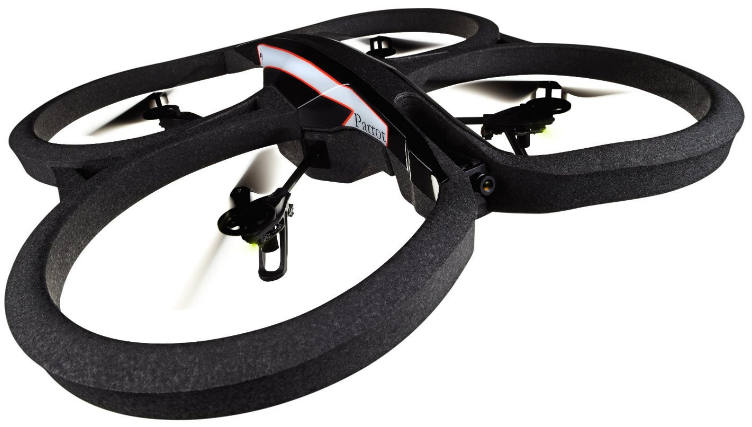


Figure 5.2: Parrot AR.Drone 2.0 with its protective indoor hull.

On the one hand, using a Wi-Fi connection for controlling the MAV has the advantage of being independent of the platform of the controlling device. On the other hand, a Wi-Fi connection often suffers from unpredictable lag spikes. For stabilization, the AR.Drone is equipped with an onboard Linux system and a set of internal and external sensors [57]. Using this data, the onboard controller can keep the MAV in a hovering position in the case of a communication loss and also compensate wind to a certain extend. Although the hovering abilities are useful in the case of a connection loss, the MAV pose significantly drifts while hovering.

As internal sensors the AR.Drone provides a gyroscope, an accelerometer, a magnetometer and a pressure sensor. Additionally, the AR.Drone is equipped with ultrasound sensors for the height estimation, and a vertical camera, which is used for ground speed measurement [57]. Finally, the AR.Drone has a forward looking HD camera, which is the only exteroceptive sensor used by our implementation.

Camera Details. The main camera, which is looking in the forward direction, has a maximum resolution of 1280×720 px and a maximum frame rate of 30 fps. In our experiments we used a smaller resolution of 640×360 px at a frame rate of 10 fps. Note that, in the case of the AR.Drone2.0, using a smaller image size does not necessarily lead to less information in the image. In order to sustain a resolution of 1280×720 px at 30 fps over a Wi-Fi connection, the AR.Drone2.0 is forced to use a very high compression rate, which leads to an image with clearly visible compression artifacts.

The main camera is a fish-eye camera with a diagonal field of view of 92° [57]. For our experiments we also measured the field of view along the central axes of the camera. We found that along the horizontal axis the camera's perception is limited to a field of view of approximately 60°, and to 40° along the vertical axis.

5.1.2 Vicon

For our evaluation, we did not solely rely on the pose estimate of PTAM [37], but also used a Vicon tracking system[7]. The tracking setup consists of 15 M-Series cameras, which are mounted along the wall of a non-rectangular 120 m^2 room. The effective tracking volume of this setup is approximately 5×5×3 meters.

Each vicon camera is equipped with an array of infra-red (IR) LEDs. The light emitted by the cameras is used to track small reflective balls in the tracking volume. To estimate the location as well as the rotation of a rigid object, it is necessary to use at least three reflective balls. To increase the robustness against self-occlusion we decided to use seven reflective balls, as depicted in Figure 5.3. The Vicon system measures the pose of the MAV with a frequency of approximately 120 Hz.

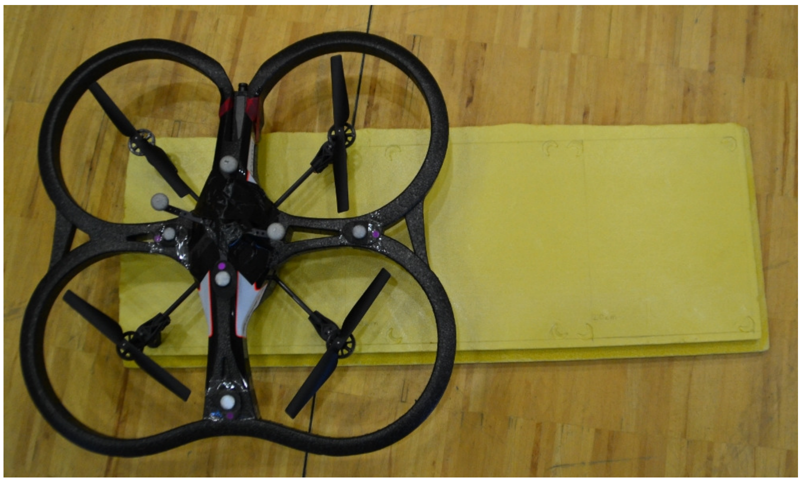


Figure 5.3: Parrot AR.Drone 2.0 with its protective indoor hull and seven reflective balls for tracking with the Vicon system.

To determine the accuracy of the setup, we analyzed the noise of the system over 5 seconds, while the MAV remained motionless on the floor. The results are shown in Table 5.1. In summary, the system has a location uncertainty of approximately 0.1 mm and a rotation uncertainty of approximately 0.001 rad.

[7]http://www.vicon.com

	x	y	z	yaw	pitch	roll
unit	mm	mm	mm	rad	rad	rad
STD	0.08401	0.07375	0.06245	$0.9811 \cdot 10^{-3}$	$0.7252 \cdot 10^{-3}$	$0.5199 \cdot 10^{-3}$
$max - min$	0.6242	0.4847	0.3914	$0.5426 \cdot 10^{-2}$	$0.4026 \cdot 10^{-2}$	$0.3142 \cdot 10^{-2}$

Table 5.1: Accuracy of the Vicon tracking setup. Standard deviation and the difference between maximum and minimum value of 589 pose measurements, which were recorded over 5 seconds, while the MAV was standing completely still on the floor.

5.2 Determining the Point Recognition Probability

The main purpose of this work is the design of a system which does not lose the visual localization while navigating towards a destination pose. In our case, the localization is considered lost if the system is unable to determine its current position through tracking the already mapped point cloud. For this purpose it is beneficial to know from which viewpoint it is possible to perceive and recognize a map point.

In order to get a reliable guess whether or not a map point is visible from an arbitrary viewpoint, it is crucial to know how a viewpoint change influences the recognition probability of features. This chapter describes the analysis of the point recognition probability for the PTAM [37] system, which we used for all our experiments.

The next paragraph will shortly outline the most important properties of the PTAM tracking procedure to clarify the scope of this experiment, which is described in the subsequent paragraph.

PTAM Tracking Procedure. In PTAM each map point has an associated source keyframe, which is the first keyframe this map point was observed in, and a 2D location within this keyframe. Furthermore, the system records on which scale the map point was detected. For tracking a map point the system uses the relative transform between its source keyframe and the estimate of the current camera pose to determine what the patch around the map point should look like in the current image. To this aim, PTAM assumes that the surface normal of a map point is parallel to the optical axis of the camera pose of its source keyframe and applies an affine warping transform. After creating a suitable template patch, the system searches for the map point within a fixed range from its predicted image location. A patch is considered found if the zero-mean SSD score is beneath a predefined threshold.

Experimental Scope. In general, it is not possible to find the effective point recognition probability for all possible scenes and scenarios as the assumptions made by PTAM can be violated in many ways. Therefore, we only aim to set an upper bound on the point recognition probability in this experiment. To determine this upper bound, which we use to set the parameters of our model (see Section 3.1.2) adequately, we do not violate the assumptions made by the PTAM system. First of all, we restrict the scene to a single well textured plane, which means that each 3D map point can be well represented by an image patch. Secondly, we set the orientation of the this textured plane parallel to the image plane for the initialization, which leads to a perfect surface normal estimate for the map points. Note that in a real world example these two assumptions will be severely violated.

For our analysis, we split the whole spectrum of possible perspective transformations into two parts which directly fit our model. First of all, we analyze the influence on the point recognition probability of viewing the same map points from an increasingly steep angle. Then we conduct the same experiment for varying the distance between the map points and the camera, which results in a variation of the feature scale.

Note that for the following two experiments we only used the initial pair of keyframes and disabled any further keyframe generation as well as the outlier removal procedure of the PTAM system.

5.2.1 Angle Dependency

Let us consider a planar 3D patch which is located parallel to the image plane. If then the patch is rotated around e.g. the vertical axis, the area of the projection of the patch on the image plane decays, and with this also the possible information about the patch. After a patch rotation of 90 degrees, the image does not contain any information about the patch. Let us assume that every object can be approximated through a set of planar patches. Under this assumption, we want to analyze how the degree of off-plane rotation of patches influences the probability of recognition.

Experimental Procedure. Instead of actually rotating the feature target in the real world, we decided to rotate the camera around the object at a fixed distance, which leads to the same image projection. We initialize the PTAM system parallel to a planar feature target with the size 0.5 m x 1 m, where the first keyframe is taken central to the feature target and the second keyframe shifted 20 cm to the right. Then we move the camera back to the central position. From this position we move the camera to the left in steps

5 degrees along an arc with a distance of 2 m to the center of the feature target. At each step we record the number of tracked points versus the number of possibly tracked map points. The procedure is depicted in Figure 5.4 for clarification. After the loss of tracking the camera is moved back along the same arc to analyze the recovery behavior of the system. To obtain reliable results the procedure was repeated seven times.

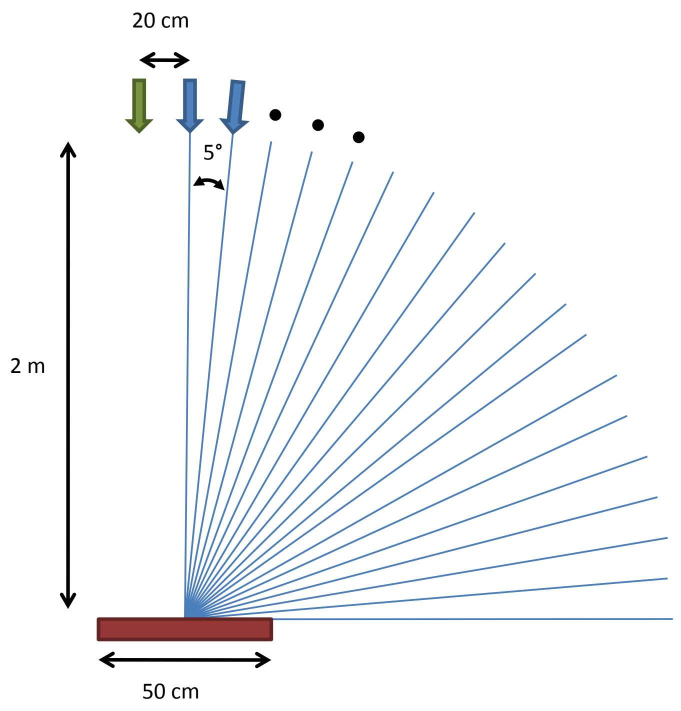


Figure 5.4: Experimental procedure to evaluate the impact of angle variation on the tracking performance. The red box symbolizes the feature target, whereas the arrows symbolize the camera of the MAV. The first key frame is taken from the position of the central arrow, the second keyframe from the position of the green arrow. Then the camera is moved back to the central position. From this position we move the camera in steps 5 degrees (blue lines) along an arc with a distance of 2 m to the center of the feature target. At each step we record the number of tracked points versus the number of possibly tracked map points.

Results. For the evaluation of the angle dependency of the point recognition probability, we analyzed the averaged percentage of tracked points over the map points that could have been tracked. The overall results can be found in Figure 5.5. The remainder of this paragraph will outline the most important findings of this experiment.

The outcome of the experiment shows that after the initialization the tracking performance stays nearly constant up to 30° with a very low decline of approximately 0.3 percent per degree. Above 30° the steepness of the decline increases drastically and hits its maximum above 60° with nearly 3 percent per degree. The average angle at the point of a localization loss is 71.875° and above 80 degrees the tracking failed in all seven runs.

After the loss of localization the system cannot recover until on average 30.714°. For a further decrease of the angle the probability of recovery increases further, and reaches 100 percent at 20°. Note that after a successful recovery the number of tracked points is approximately the same as if the localization would not have been lost.

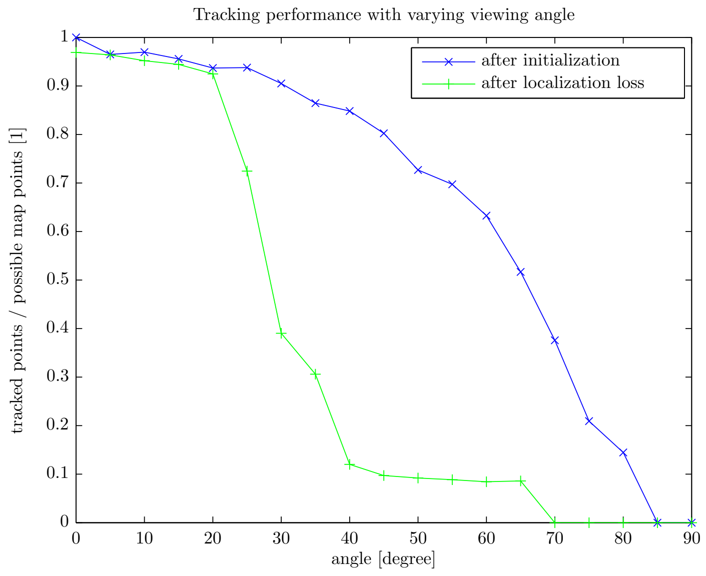


Figure 5.5: Point recognition probability with varying viewing angle. After initialization (blue line) the angle between the normal of the feature plane and the optical axis is subsequently increased until the localization is lost. After the loss of localization (green line) the angle is decreased again until the initial pose is reached.

5.2.2 Scale Dependency

Scale invariance in PTAM is achieved through an image pyramid. The system uses four pyramid levels, where each higher level has half the side lengths of the previous level. We will call the level containing the original image "level 0" and the smallest resolution of the image "level 3".

Using an image pyramid the system can only adapt the scale within the layers of the pyramid. This means that features, which are detected on different pyramid levels, show different properties in the recognition probability in relation to scale changes. Consequently, we decided to analyze the features of each layer individually by only using the features of one layer at a time for tracking. Unfortunately, it was not possible to achieve a successful pose estimation solely based on features of the coarsest pyramid level. Because of the very small size of the image at this level (80×45 pixels), we could not extract minimum number of features that PTAM needs for successful tracking. Therefore, we neglect this layer in our experiments, which does not harm the evaluation overly much as this layer can be extrapolated from the other three layers.

We split the evaluation into two parts, one where we move the camera closer to the feature target and one where we move farther away.

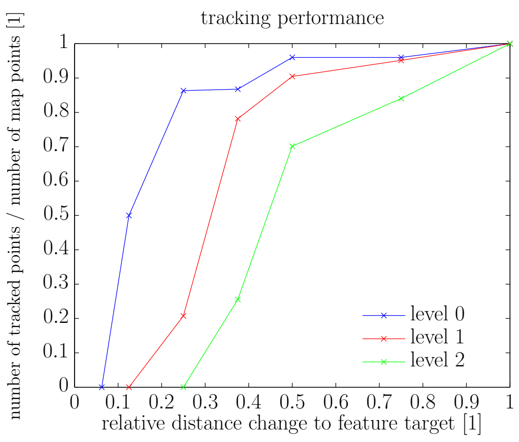
Experimental Procedure. For both parts of the experiment we used a planar feature target with a size of 1m x 0.5m, and initialized the point cloud with two key frames, which have an image plane parallel to the target. For the experiment where the camera is moved closer to the target, we used an initial distance to the target of 80cm and a keyframe baseline of 8cm, whereas for the second experiment we used an initial distance of 20cm and a keyframe baseline of 2cm.

Results. In this experiment we analyzed the relation between scale changes and the point recognition probability. Similar to the previous experiment, each part of the experiment was repeated four times. The averaged results can be found in Figure 5.6.

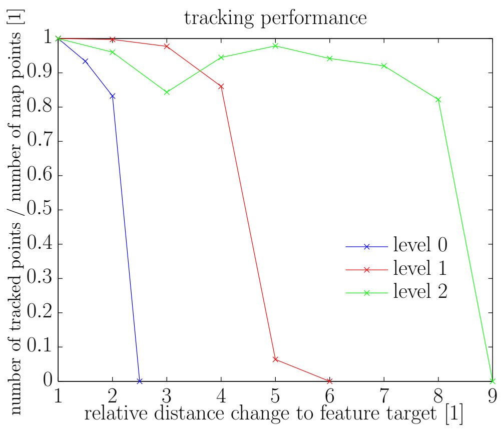
For the first experiment, we decreased the distance to feature target. In this case the scale of the features grows finer inverse to the distance change. In moving closer features start to disappear at the edges of the image and only features in the central part of the image remain for detection. For this experiment, we used our projection model to determine, how many points should be visible for a given camera pose. We use this information to get comparable results to the next case.

For the second experiment, we increased the distance to feature target. As apparent from Figure 5.6b, each layer has a different capability to adapt to the scale changes. Features, which were detected on a higher level of the image pyramid, can be detected on lower levels if the camera is moved farther away. The lowest level of the pyramid can still adapt to scale changes up to the factor 2, where still 80 percent of the features can be recognized, although it has no other layer it can turn to. The other levels show a similar behavior. The capability to adapt to scale changes between adjacent levels appears to be approximately 2, which directly corresponds to the information reduction between the pyramid levels. For features detected on the highest pyramid level, which could not be analyzed directly, this means that the system can handle scale changes up to a factor of 16.

Overall, the approach appears to be less robust against moving closer than moving farther away. In contrary to increasing the distance, the system has troubles even finding the feature points on the highest / coarsest pyramid level, much less extending further. This behavior could be explained with the very limited size of the coarsest level (80×45 pixels). As PTAM matches patches of the size 8×8 pixels and only considers patches which are fully visible in the image, many features at the margins of the image are only partly visible and are therefore not detected.



(a) Decreasing distance to the feature target.



(b) Increasing distance to the feature target.

Figure 5.6: Point recognition probability with varying distance to feature target.

5.2.3 Conclusion

The aim of this series of experiments was to evaluate the influence of viewpoint changes on the point recognition probability of the tracking system. We evaluated the ideal case which means that all keyframes have an image plane parallel to the plane of the planar target. Note that in general environments the surface can have an arbitrary topology and a single image patch could even contain projections of overlapping objects with very different depth values. Nevertheless, an evaluation of the ideal case makes sense to set an upper bound on the system performance.

For the angle dependency, we found that the recognition probability stays nearly constant up to an angle of 30° and that the average angle at the point of a localization loss is 71.875°.

For the scale dependency, we discovered a significant difference between increasing and decreasing the scale. It appears that the PTAM system can handle a decrease of the scale (moving farther away) much better than increasing the scale (moving closer). While a decrease can be handle up to a factor of 2 without using other pyramid levels, an increase can only be handle up to a factor of approximately $\frac{4}{3}$. Furthermore, the system appears to be unable to detect points of the coarsest pyramid level. The most likely cause for this discrepancy seems to be that, in moving closer, regions around many keypoints become only partly visible in the image, which does not happen in the other case.

As described in Section 3.1.2, we model our feature recognition probability using piecewise linear functions to speed up the calculation. Without any prior knowledge about the scene, we recommend to set the constants of the model such that they are clearly beneath the upper bound values displayed in Figures 5.5 and 5.6. For our experiments we set the values $\alpha_0 = 0.5$ rad and $\alpha_1 = 1.2$ rad for the angle dependency as in Equation 3.6. For the scale dependency we chose an equal relative spacing between the layers. The actually used values can be found in Table 5.2.

pyramid level $l \rightarrow$	0	1	2	3
$S_{0,l}$	0.1	0.2	0.4	0.8
$S_{1,l}$	0.2	0.4	0.8	1.0
$S_{2,l}$	1.5	3.0	6.0	12.0
$S_{3,l}$	2.5	5.0	10.0	20.0

Table 5.2: Chosen values for the scale dependency model in Equation 3.8.

5.3 Navigational Imperfection

Every dynamic robotic system suffers from an imperfect execution. The extent of this imperfection depends on many factors such as the manufacturing accuracy of the hardware, the level of detail of the system model or environmental influences. In this work we aim to design an autonomous explorative navigation system. For every autonomous system it is crucial to keep a safety distance to obstacles and unknown environments in order to avoid states which endanger the robot or the environment.

The main purpose of the set of experiments described in this section is to evaluate the navigational imperfection of the system in well defined scenarios. Through these experiments we aim to derive information which is needed for safe navigation. First and foremost, we are interested in the deviation from predefined trajectories such as they are generated by the planning logic of our explorative navigation module. Opposed to Katusic [36], we do not only evaluate trajectories which are parallel to the x- and y-axis, but extend the evaluation to the flight in diagonal patterns for trajectories parallel to the surface as well as for trajectories, which vary significantly in the height. With this information it is possible to generate a realistic model of the collision probability. Secondly, we use this experiment to analyze the controller reaction speed. Ultimately, this information can then be used to derive a reasonable planning frequency for the explorative navigation module. As we noticed that the overall system suffers from severe lag spikes, we provide a separate analysis on this topic to relate the navigational imperfection to the lag spike occurrence.

For the plan execution in this work, we make use of a human-inspired fuzzy-logic controller of Katusic [36]. The controller was originally designed for the usage with a different quadrotor MAV (AscTec Pelican), consequently, some minor changes were applied to adapt the controller to the AR.Drone 2.0.

5.3.1 Experimental Procedure

For this experiment, we chose to use a richly textured scene, as depicted in Figure 5.7. The baseline for the initialization was chosen be the 20 cm and the depth of the feature points in the scene is approximately 4m. In this experiment we used in total 3 flight patterns. The first two are horizontal motion trajectories and the third one is a vertical trajectory. The flight plan with all three trajectories is shown in Figure 5.8.

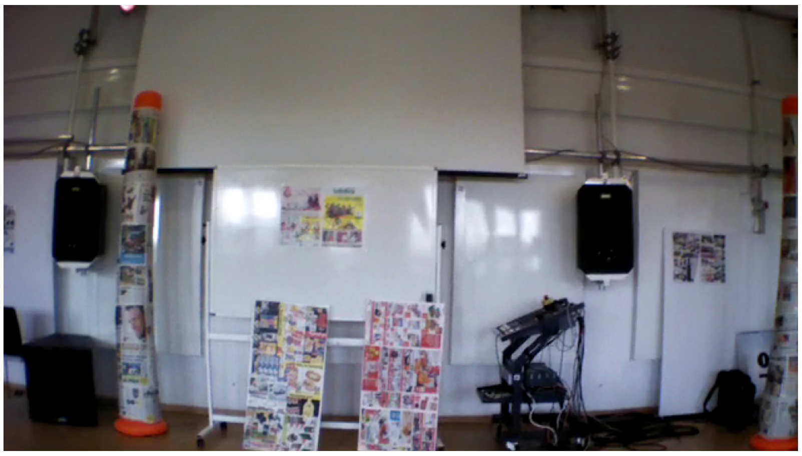


Figure 5.7: Chosen scene for the analysis of the navigational imperfection.

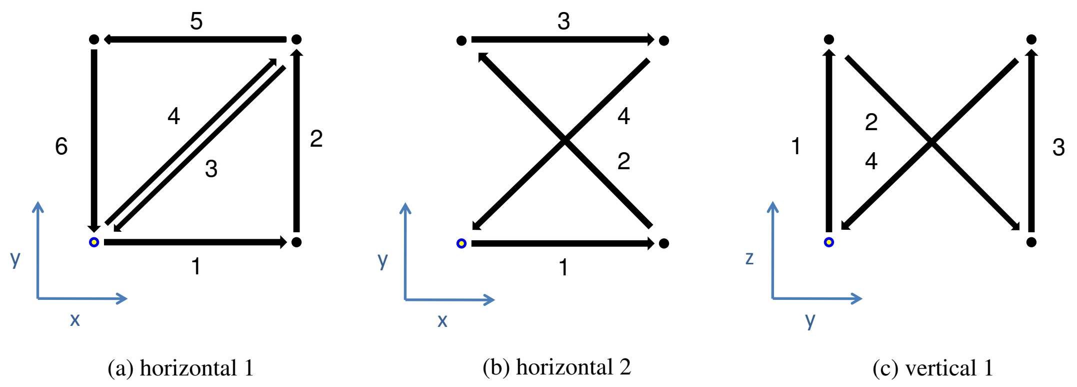


Figure 5.8: Flight plan with three different trajectories. The black dots represent *intermediate* destinations, whereas the blue-yellow dots symbolize the start and final destination positions. The numbers in the diagrams stand for the sequential order during the experiment. Note that the first two trajectories only contain motions parallel to the surface whereas the last trajectory is oriented vertically and does not contain any forward motion (x-direction).

5.3.2 Results

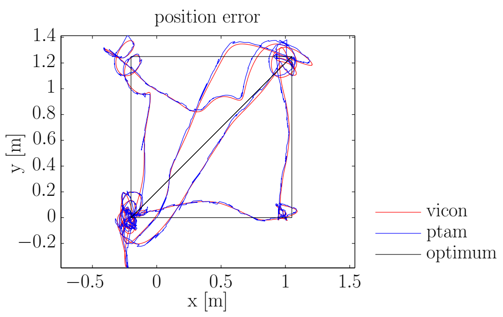
As it was not the task to implement or improve the navigation module, the main purpose of this evaluation is not a comparison to other controlling approaches, but to document the performance of the used controller. The performance of a robotic system strongly depends on the accuracy and speed of the navigation module, thus we deemed it necessary to evaluate the navigation module on its own to document its performance independent of our autonomous explorative navigation logic.

Trajectory Flight Performance. For the calculation of the navigation error, it was necessary to sample the ideal trajectories, which are denoted as "optimum" in the figures. Following the example of Katusic et al. [36], we chose a sampling step size of 0.5 mm, but opposed to their work, we do not consider the shortest distance to the whole trajectory, but the shortest distance to the current partial trajectory. Each partial trajectory corresponds to a straight line from the last destination to the current destination as depicted in Figure 5.8. When the system considers a destination as reached, we switch to the next trajectory.

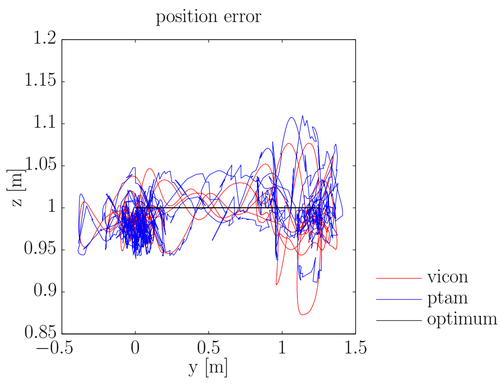
The experiment was conducted for three different target trajectories. The real trajectories, which were independently measured by the PTAM and the Vicon system, are depicted in Figure 5.9. Note the different scaling of the graphs. From these trajectories alone, one can see that the MAV trajectory shows a greater deviation parallel to the surface (xy plane) than in the vertical direction (z direction). This behavior is affirmed by the statistics in Table 5.3. In all three experiments the absolute mean and maximum deviation as well as the RMS error are significantly lower for the vertical direction than for the horizontal plane. The values of the absolute mean deviation support the notion that the navigational uncertainty in the horizontal plane is at least twice as large as the uncertainty in the vertical direction.

If we analyze the overall distance to the target trajectory, we see that the mean distance is below 14 cm with a standard deviation of approximately 10 cm. As the distance from the trajectory is not an independent random variable but depends on the current state of the MAV (position, velocity, acceleration ...), it is not possible to draw any conclusion in terms of the central limit theorem. Nevertheless, we can use the maximum values of this experiment to set the recommended distance to obstacles in a reasonable range. The farthest distance from the optimal trajectory was recorded with 0.53m. Consequently, if we set the minimum distance to known obstacles to the double of the maximum value the probability of collision should be sufficiently low.

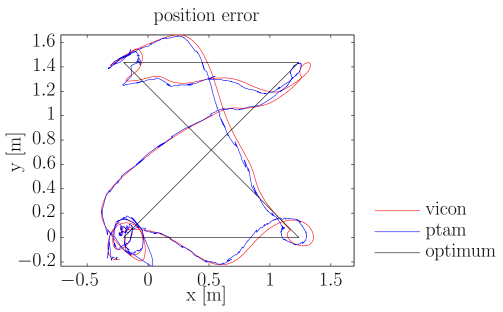
Controller Reaction Time. The planning module proposed in this work, generates nearby destination poses which can be safely reached without losing the visual localization. Generating new poses at a very high speed causes more harm than good to the overall system as the internal state of the PID controller needs finite time to adapt its internal state. In this experiment we analyze the time it takes the controller to close a distance of 20cm towards a newly received destination pose. This distance was chosen because it corresponds to the minimum planning distance in our other experiments.



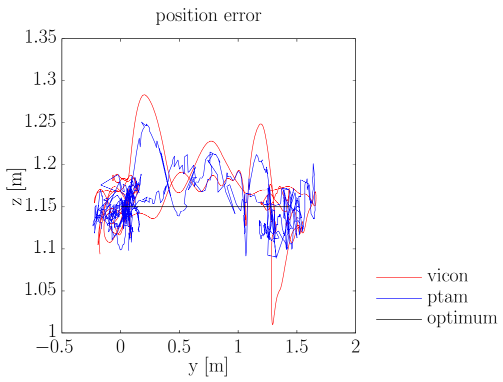
(a) horizontal 1: xy view



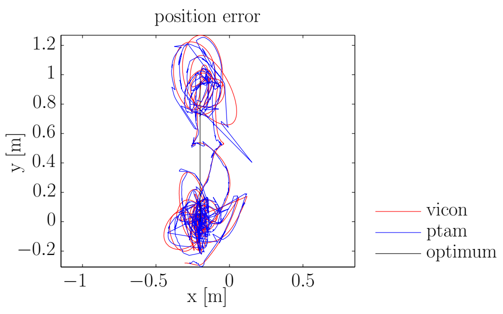
(b) horizontal 1: yz view



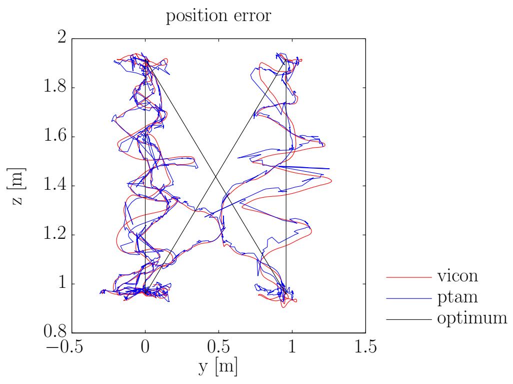
(c) horizontal 2: xy view



(d) horizontal 2: yz view



(e) vertical 1: xy view



(f) vertical 1: yz view

Figure 5.9: PTAM and Vicon pose for three different trajectories.

trajectory	dimension	RMSE	absolute mean	STD	absolute max
horizontal 1	x	0.075	0.054	0.052	0.272
	y	0.112	0.066	0.091	0.430
	z	0.027	0.022	0.017	0.127
	all	**0.138**	**0.107**	**0.086**	**0.430**
horizontal 2	x	0.108	0.067	0.084	0.373
	y	0.129	0.101	0.081	0.373
	z	0.035	0.025	0.025	0.140
	all	**0.172**	**0.138**	**0.103**	**0.527**
vertical 1	x	0.090	0.064	0.064	0.319
	y	0.118	0.094	0.071	0.336
	z	0.065	0.029	0.058	0.261
	all	**0.162**	**0.137**	**0.086**	**0.377**

Table 5.3: Statistics of the minimum distance to the optimal path for three different trajectories. The minimum Euclidean distance to the current trajectory was calculated for every ground truth pose which was measured by the Vicon system at approximately 120 Hz. For this purpose the trajectories were sampled in steps of 0.5 mm. The statistics for the 3D Euclidean distance are denoted as *all* and are printed with a bold face. Additionally, the table contains the same statistics for each dimension on its own. The x dimension in this experiment can be interpreted as "forward/backward", y as "left/right" and z as "up/down".

flight	mean	STD	min	max
unit	[s]	[s]	[s]	[s]
horizontal 1	9.90	2.97	5.02	13.86
horizontal 2	6.83	1.43	5.52	8.79
vertical 1	13.10	3.78	9.19	17.30
overall	9.94	3.63	5.02	17.30

Table 5.4: Controller reaction time. The table shows the statistics of the time it effectively took the MAV to close a distance of 20cm towards a newly received destination for the three different trajectories.

The results in Table 5.4 show that the mean reaction time is approximately 10 seconds with a high standard deviation of nearly 37 percent. On the one hand, the reaction time strongly depends on the current motion vector of the MAV as there is a great discrepancy between making a 90 degree turn or reversing the direction. On the other hand, it can be influenced by the occurrence of unpredictable lag spikes as the pose controller is running on a separate computer and depends on the Wi-Fi connection to the MAV. To this end, we analyzed the occurrence of *severe* lag spikes for each trajectory flight. We consider a lag *severe* if there is neither a new frame nor any IMU data coming through to the system for

flight	severe lags	severe lags per minute	maximum lag
unit	[1]	$[\frac{1}{min}]$	$[s]$
horizontal 1	12	6.26	1.43
horizontal 2	2	1.94	0.88
vertical 1	37	16.95	1.32
overall	51	9.94	1.43

Table 5.5: Lag statistics of the three experiments. We consider a lag *severe* if there is neither a new frame nor any IMU data coming through to the system for at least 0.5 seconds.

at least 0.5 seconds. The results are shown in Table 5.5. If we compare the reaction time to the lag occurrence, we can see a clear relation. For the flight "vertical 1" we recorded by far the slowest reaction time, but also the highest frequency of lag spikes with nearly 17 lags per minute. On the other hand, the flight "horizontal 2" shows the fastest reaction time, but during this flight the system only had to cope with 2 lags per minute.

5.3.3 Conclusion

The aim of this series of experiments was to analyze the extent of the navigational imperfection of the system. In this experiment we recorded three different trajectories, which do not only contain axis parallel motions parallel to the surface, but also diagonal and vertical motions. We discovered that the deviation in the vertical direction is significantly smaller than parallel to surface. This means that an ellipsoidal metric, as discussed in Section 3.3.2, is better suited for the task of collision avoidance than a simple Euclidean metric. As the maximum deviation from the optimal trajectory in our experiments was 0.53m, we recommend to keep the distance from the planned trajectory to obstacles in the horizontal plane at approximately 1m, and 0.5m for the vertical direction, to keep the probability of a collision reasonably low. Note that the MAV is controlled over a Wi-Fi connection, and consequently it is impossible to make guarantees for the topic of collision avoidance. In fact, we measured communication disruptions of up to 1.5 second during this experiment and recorded a frequency of up to 17 severe lags per minute. This issue can only be fixed by using a more reliable data link. Finally, we analyzed the effective reaction time of the system to a new destination. We measured a mean time of 10 seconds to close a distance of 20cm towards a new destination pose. The experiments confirm that the reaction time of the system is influenced by the frequency and duration of the communication lags.

5.4 Localization Quality

The main aim of this work is to design a system which does not lose the visual localization during execution. To this aim, we devised a measure that we call "localization quality". It has the purpose of indicating the likelihood of a localization loss for an arbitrary camera pose. In this chapter, we evaluate the performance of the measure in forcing the tracking system into extreme situations, which result in a localization loss. In a static environment the easiest way to lose the visual localization is not translation but rotation. Consequently, we focus our evaluation on the effects of rotation.

5.4.1 Experimental Procedure

For this experiment, the camera of the MAV is the only necessary source of information, thus we decided to perform this experiment in a hand operated manner. The experiment was conducted with two different setups. The first setup is a fairly low textured laboratory scene (see Figure 5.10) and for the second setup we added highly textured feature targets to the scene to increase the number of feature points (see Figure 5.11) . To establish a ground truth for the MAV pose, we used the Vicon tracking system described in Section 5.1.2.

For both setups, we initialized the system with a baseline of 20cm mounted on a table with a height of 70 cm and a width of 120 cm. Then we produced in total 7 keyframes by moving the camera in vertical-rectangular motion above the table. After the generation of the 7 keyframes, the keyframe creation is deactivated to lock the state of the virtual scene representation. The parameters of the localization quality are automatically tuned after the initialization to adapt to the current scene.

After the initialization, we rotate the camera in a hand-held manner into 8 different directions as depicted in Figure 5.10 and 5.11. For each direction, we start with a central view and rotate the camera in the chosen direction until the localization is lost. Afterwards the camera is returned to the central view for the next direction. During the experiment, we record the MAV pose which is tracked by the Vicon system at a rate of 120 Hz and the camera pose of the PTAM system at a rate of 10 Hz. Using our MAV model, we can calculate the relative error between the MAV pose estimated by the PTAM system and the pose tracked by the Vicon system.

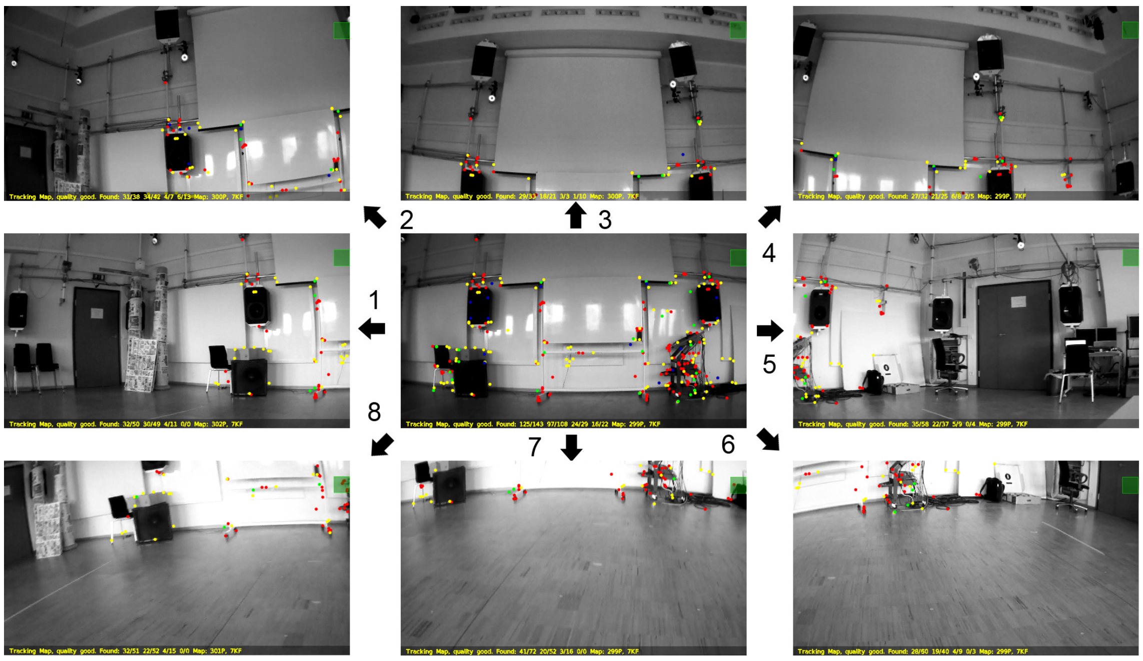


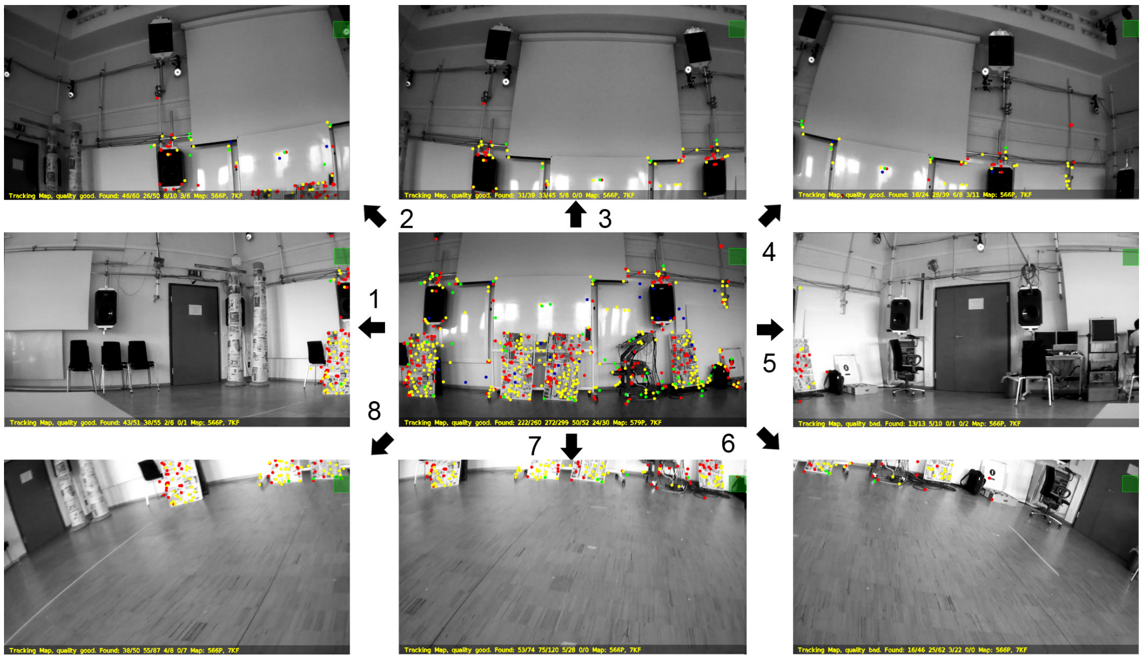
Figure 5.10: Localization quality evaluation procedure in a sparsely textured scene. The images are views of the PTAM system. The image in the center of the figures is the view after initialization and is the start point for the motion in each individual direction. In the outer images, one can see the same scene after the rotation in one of the 8 directions. The images were taken just before the system loses the visual localization. The colored dots represent the currently tracked map points. The black numbers stand for the experimental order for the rotations.

5.4.2 Results

In our experiment we consider two different scenes. The first one is sparsely textured and contains approximately 300 map points. The second one has the same background, but 4 feature targets were added, which roughly doubles the number of map points.

Figures 5.12 and 5.13 plot the pose error as well as the localization quality during the experiment. If we consider the curve of the localization quality, we can see that each loss of the visual localization is preceded by a significant drop in the localization quality. This is exactly the behavior that is needed for an active system to prevent a localization loss, as it enables the system to detect the approach to such a ”dangerous” state before it is too late.

If we take a look at the statistics in Table 5.6, we can observe that for both scenes the localization quality is always clearly beneath 20 percent at the time of the localization



```
Tracking Map, quality good. Found: 46/60 26/50 8/10 3/6 Map: 566P, 7KF
Tracking Map, quality good. Found: 31/39 33/45 5/8 0/0 Map: 566P, 7KF
Tracking Map, quality good. Found: 16/24 28/39 6/8 3/11 Map: 566P, 7KF
Tracking Map, quality good. Found: 43/51 38/55 2/6 0/1 Map: 566P, 7KF
Tracking Map, quality good. Found: 222/260 272/299 50/52 24/30 Map: 579P, 7KF
Tracking Map, quality bad. Found: 13/13 5/10 0/1 0/2 Map: 566P, 7KF
Tracking Map, quality good. Found: 38/50 55/87 4/8 0/7 Map: 566P, 7KF
Tracking Map, quality good. Found: 53/74 75/120 5/28 0/0 Map: 566P, 7KF
Tracking Map, quality bad. Found: 16/46 25/62 3/22 0/0 Map: 566P, 7KF
```

Figure 5.11: Localization quality evaluation procedure in a richly textured scene. The images are views of the PTAM system. The image in the center of the figures is the view after initialization and is the start point for the motion in each individual direction. In the outer images, one can see the same scene after the rotation in one of the 8 directions. The images were taken just before the system loses the visual localization. The colored dots represent the currently tracked map points. The black numbers stand for the experimental order for the rotations.

scene	mean	STD	min	max
sparsely textured	0.112	0.061	0.036	0.183
richly textured	0.049	0.054	0.002	0.177

Table 5.6: Localization quality as in Equation 3.14 at the point of localization loss for the richly and the sparsely textured scene.

loss.

If we compare the curves of the pose error to the localization quality, we can note that not every localization loss is preceded by rise of the localization error. This means that even if we could somehow determine the real localization error, it would be no indicator for an imminent localization loss.

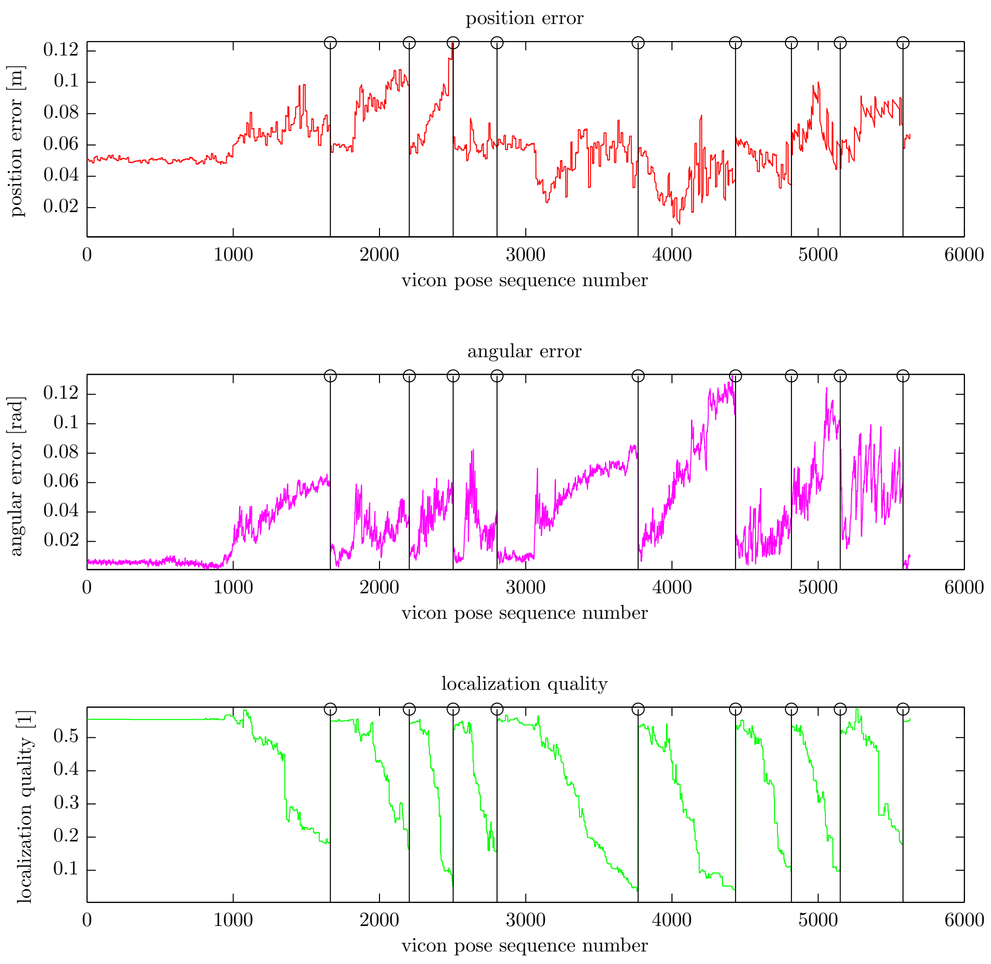


Figure 5.12: MAV pose error and localization quality for a sparsely textured scene. The black stems mark points where the localization is lost. The experiment was conducted 9 times in 8 different directions. The first and the last direction are the same.

5.4.3 Conclusion

The main purpose of this work is to maintain the visual localization at all times during the autonomous mission. Therefore it is crucial to avoid motions which will very likely lead to a loss of the localization. In this experiment we have demonstrated that an approach of the camera to such a "dangerous" state can be predicted using our novel localization quality measure. During our experiment the localization was never lost above a localization quality

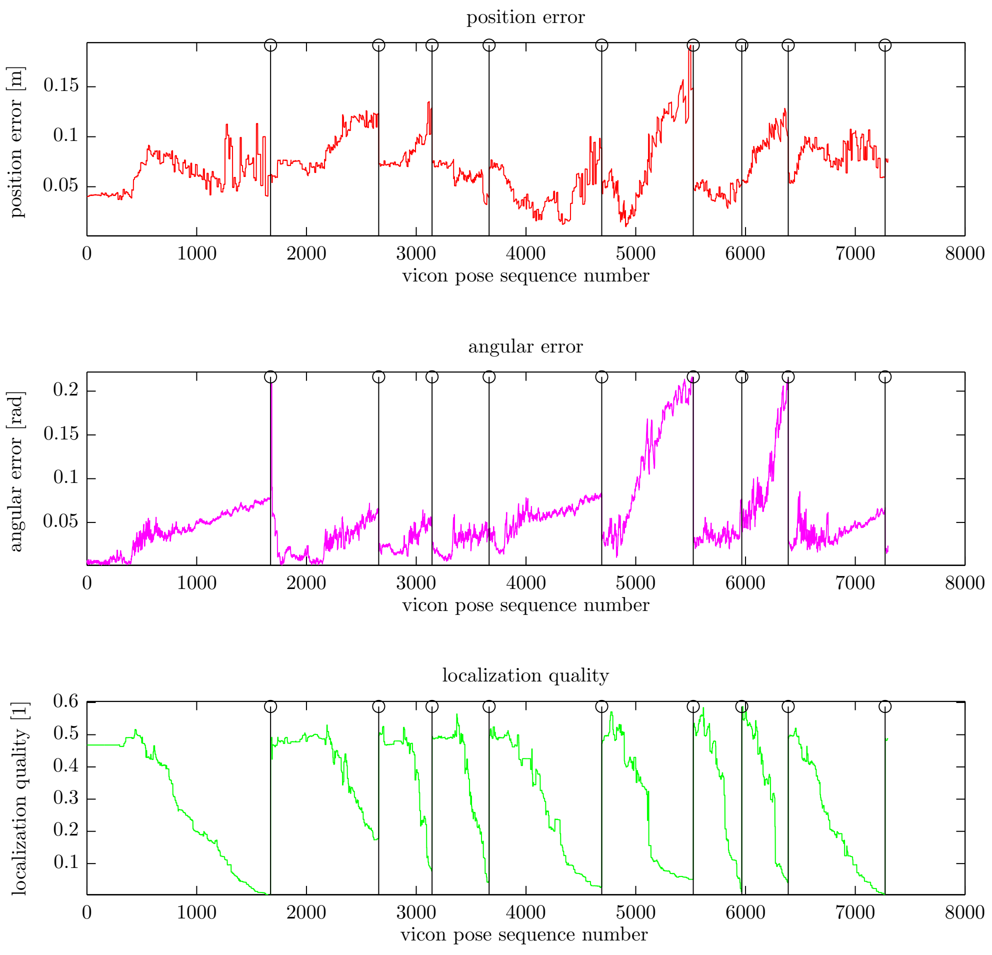


Figure 5.13: MAV pose error and localization quality for a richly textured scene. The black stems mark points where the localization is lost. The experiment was conducted 9 times in 8 different directions. The first and the last direction are the same.

value of 20 percent, neither for the sparsely textured scene nor for the richly textured scene.

5.5 Autonomous Explorative Navigation

In this work we devised a monocular system which can navigate towards a destination pose in an arbitrary environment. We do not assume that the environment is known before hand, but acquire the relevant information during the execution. The proposed system is able to autonomously decide whether a destination pose can be safely reached without risking a collision or the loss of the visual localization given the currently available information about the environment. In the case that insufficient information is available, the system autonomously attempts to generate the relevant information.

This chapter is concerned with the evaluation of the overall autonomous system. We evaluate the performance of the system in two different scenes with two different high level objectives (missions).

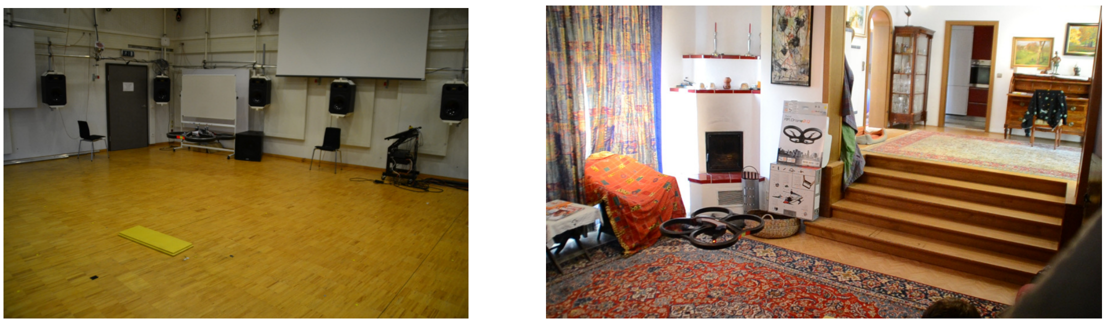

(a) vicon laboratory (b) multi-level living room

Figure 5.14: Experimental scenes.

The first scene is a controlled environment of a laboratory (Figure 5.14a) which allows us to capture the ground truth pose of the MAV using the Vicon tracking system. The main purpose of this experiment is to demonstrate the advantages of our explorative approach. We will demonstrate that our approach enables a monocular MAV to perform a full 360° turn in a low textured laboratory. The exact details of the experiment are described in Subsection 5.5.1.

For the second scene we have chosen a multi-level living room (Figure 5.14b). In this experiment we will demonstrate the full capabilities of our approach. During this mission the MAV is required to perform a 90° turn and then move across the staircase without losing the visual localization. In order to successfully complete the mission the system does not only have to generate 3D map points for localization, but also has to decide which parts of the scene are not occupied by any objects and can be used for a safe navigation. A detailed description of this experiment is provided in Subsection 5.5.2.

5.5.1 Full 360° Turn Experiment

Most of the current work with monocular MAVs [1, 18, 19, 36, 78, 80] is concerned with visual servoing. For evaluation they all use predefined trajectories which do not include any rotational motion. We assume that one of the main reasons for this choice is the fact that the easiest way to lose the visual localization is a plain rotational motion. As it is the aim of this work to avoid a loss of the visual localization, we choose a 360° turn as a high level objective for this experiment. This objective can only be achieved through generating a "reasonable" baseline which allows the construction of new 3D map points with a sufficiently large triangulation.

The aim of this experiment is to demonstrate the advantages of the explorative nature of our approach in a monitored environment as well as to draw conclusions in regard to the accuracy of the localization and map generation. We show that our approach can autonomously explore the necessary parts of the environment and can handle severe navigational imprecision as well as extensive network lags without losing the visual localization. The remainder of this section is structured as follows.

First of all, we provide the details on the experimental procedure and how the high-level objective of a 360° can be translated into a set of destination poses. Secondly, Subsection 5.5.1.2 analyzes the planning and emergency behavior of our approach. Thirdly, Subsection 5.5.1.3 evaluates the localization and mapping performance of the system with respect to the Vicon ground truth and the building floor plan, respectively. Finally, we draw conclusions from this experiment in Subsection 5.5.1.4.

5.5.1.1 Experimental Procedure

For this experiment we used a weakly textured scene as depicted in Figure 5.15. Additional to the internal visual pose estimate of the system, we also track the position of the MAV with the Vicon system. The PTAM system is initialized with a baseline of $0.5m$ in the center of the room with a distance of approximately $4.5m$ to the wall in front of the MAV. To provide some navigational space for the MAV, we defined an initial free space cube with a side length of $2m$. The center of the cube is placed $1m$ above the initialization position. In order to achieve a 360° turn we send 4 destination poses to the system. Each pose has the same xyz-position (0/0/1.2), but the yaw angle is increased by 90° from one destination pose to the next. This leads to destination poses with a yaw angle in this order: 90°, 180°, 270° and 0°. The system has to reach each destination pose in this order, but in between it can freely choose its path.

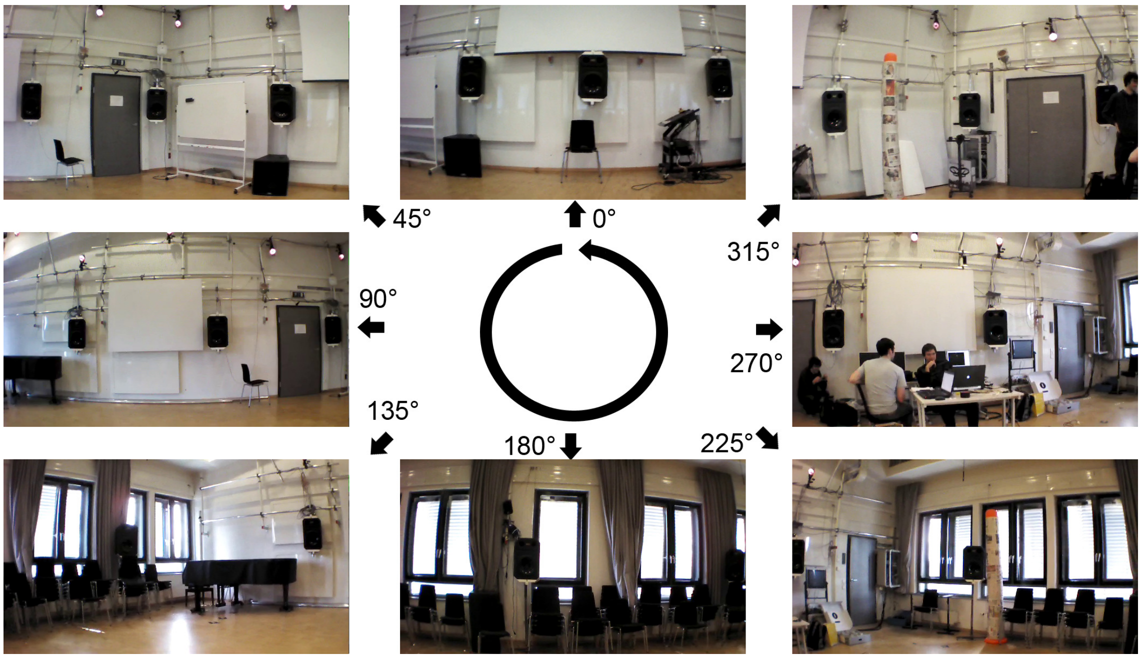
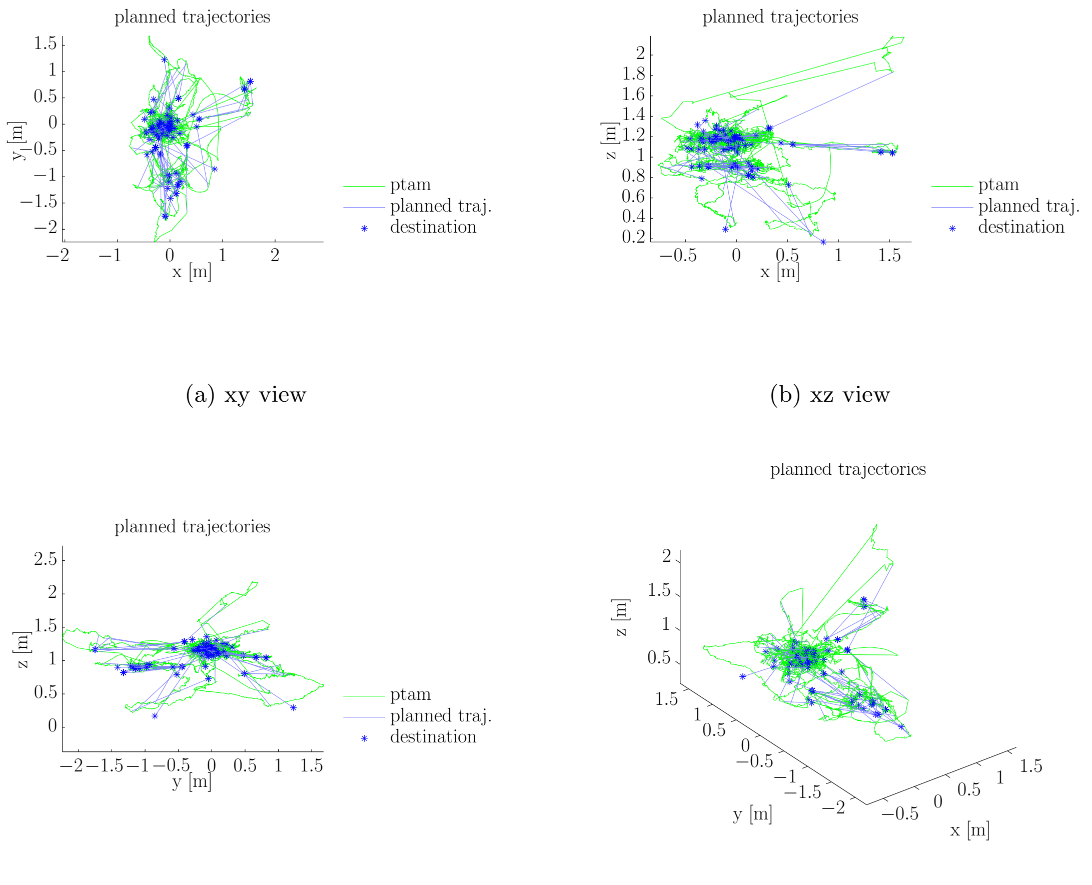


Figure 5.15: Experimental scene. A 360° view of the experimental scene captured from the MAV during the experiment. The MAV was initialized with the view in the 0° direction. During the experiment the MAV turns counter-clockwise until it has turned a full circle. This behavior is achieved through the definition of 4 *intermediate* destination poses. Each pose has the same xyz-position, but the yaw angle is increased by 90° from one destination pose to the next.

5.5.1.2 System Logic

In this subsection we aim to explain and demonstrate the functionality of our system logic using the data of this 360° turn experiment. To this end, we illustrate the path planning scheme and the changes of the system safety and the system state during the experiment.

Planning. For our navigation we do not generate a global plan which defines the whole trajectory towards the destination, but rather short time plans. These short time plans have the advantage that they do not necessarily have to be completed during the execution, but can be replanned when new information is available. Furthermore, these short time plans are calculated considering the current MAV pose. These two properties lead to a set of unconnected trajectories, as the MAV hardly ever reaches one of these "*intermediate* destinations" before it replans and moves towards a new destination.



(a) xy view

(b) xz view

(c) yz view

(d) xyz view

Figure 5.16: Short time plans and the trajectory recorded by PTAM. The asterisks symbolize the position of the *intermediate* destinations. The blue lines connect the pose of the MAV at the time of the destination generation to the related destination. They can be seen as "ideal" trajectories. The green line is the effectively recorded trajectory by the PTAM system.

Figure 5.16 displays the planned trajectories of our planning scheme as well as the trajectory of the system's own localization module (PTAM). Note that most of the *intermediate* destinations are very close to the position on which we defined the four high-level destinations. More importantly, it can be observed that the system intentionally chose to leave this position to generate new high quality map points for further execution. This occurs when the system decides that a localization-safe navigation towards the current destination can no longer be guaranteed. In this case the system selects a "relevant" subset of potential points and tries to find a pose in the neighborhood of the MAV that maximizes the likelihood to generate new map points. During this experiment this event occurred five times and Figure 5.17 shows the "relevant" subset of potential points for each case as well as the "strategic pose" which maximizes the point generation likelihood. In all five cases the system was in fact able to generate new useful map points without losing the localization, which finally led to a successful completion of the 360° turn.

System Modes. In Figure 5.18 one can see the changes of the system safety state as well as the system mode. The 6 modes can be split into two parts; the explorative navigation and the emergency response. The modes of explorative navigation, *Hold Position* (HP), *Goal-Striving* (GS) and *Strategic Exploration* (SE), want to complete a high-level task, whereas the emergency modes, *Localization Improvement* (LI), *Collision Avoidance* (CA) and *Localization Recovery* (LR), want to keep the MAV safe and sound.

The HP mode simply signalizes that a destination has been reached. If further destinations are available the system directly switches back to GS. During the GS execution the system can generate *intermediate* destinations to close the distance to the current high-level destination. If no useful *intermediate* destination with a safe trajectory to the current MAV location can be generated the system mode switches to SE and starts to generate new useful map points.

The planned trajectories as depicted in Figure 5.16 consider all safety aspects, but due to the significant imprecision of the navigational system, it can occur that the safety constraints are violated nonetheless. Therefore it is possible to intercept the modes of the explorative navigation at every point in time. This interception is done in a top-down manner.

The highest priority is given to the time since the last pose estimate. The chart in Row 1 of Figure 5.18 shows the time that has past since the last frame was received from the MAV. The curve shows in total five groups of lag spikes of which all have peaks greater than 0.5 seconds. Four groups exceed 1 second, three groups 1.5 seconds and one major lag has a duration of nearly 20 seconds. These severe lags cause the system to switch in the LR mode in which it tries to stay at a safe position only using the IMU data. This mode was given the highest priority as we cannot reliably perform any task without a reasonable pose estimate.

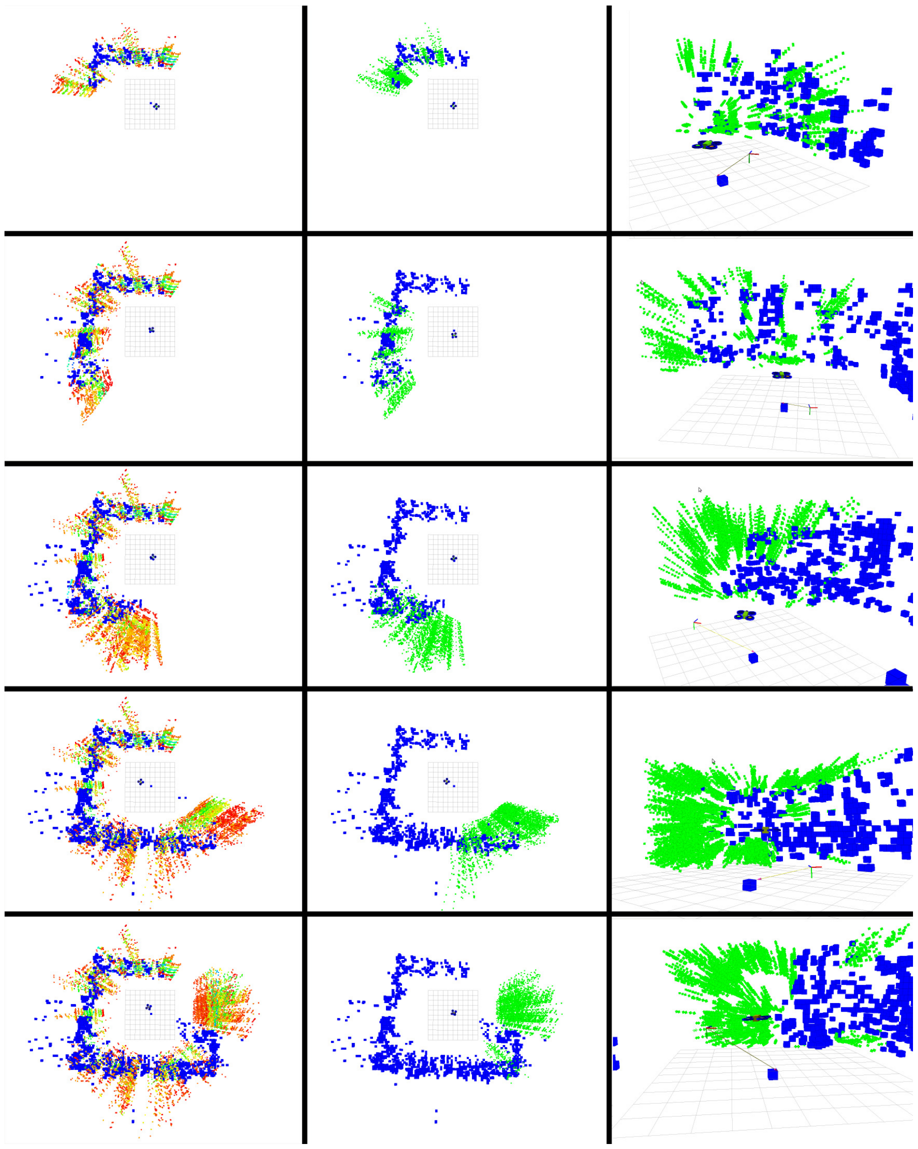


Figure 5.17: Strategic point generation. Each row depicts the ”world” state at a point in time at which the system decided that it needs to improve the point cloud to ensure a localization-safe navigation towards the destination pose. The dark blue cells can be regarded as known obstacles. The colored dots on the left hand side represent potential 3D map points. The color of these dots codes the likelihood of existence of a point at this location (”red” means high probability and ”green” the opposite). The columns in the center and on the right side depict the subset of potential points which the system deemed to be ”relevant” for the current task. The column on the right side additionally shows a special *intermediate* destination which could also be called ”strategic pose” as it maximizes the chance of generating new useful map points while localizing successfully at the same time.

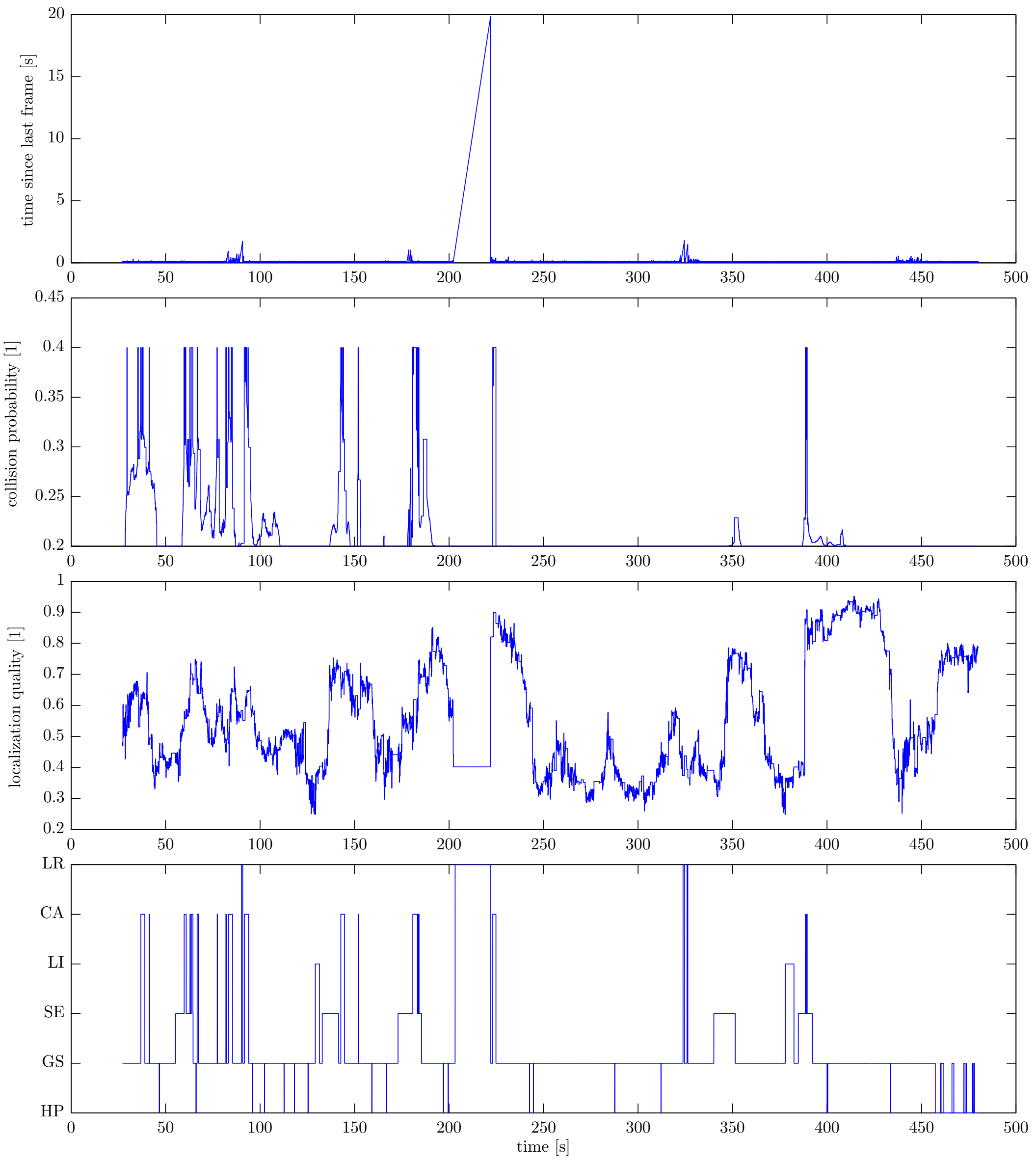


Figure 5.18: System state analysis. Row 1: The time since the last frame is used to detect lag spikes. Note that there are 4 lag spikes with a duration greater than 1 second and a major lag of nearly 20 seconds. Row 2: Collision probability over time. Due to the probabilistic model it can never drop below 20 percent. Row 3: Localization quality changes during the experiment. Row 4: System mode changes over time. The top 3 modes are emergency responses, whereas the others are part of the explorative navigation.

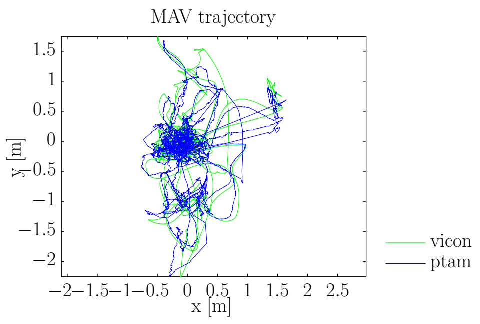
The mode with the second highest priority is the CA mode. This mode is enabled when the collision probability (Row 2 of Figure 5.18) exceeds a predefined threshold and prevents the system from getting too close to obstacles or unknown space. Note that there are no obstacles close to the MAV in this experiment. Therefore only the unknown space criterion causes this state to be entered. Due to the discrete nature of the volumetric map and the fast reaction of our system the collision probability never exceeds the threshold of 0.4 in this experiment.

The last of the emergency modes is LI which is entered if the localization quality (Row 3 of Figure 5.18) drops too low. Note that only significant navigational imprecision causes this mode to be entered. Through our planning scheme the system tries to avoid such states. In this experiment the system switches five times into the SE mode to create new useful map points and keep the localization quality high, whereas it only enters the LI mode two times. This means that, despite the extensive lags and the navigational imprecision, the system was able to prevent states with a low localization quality, in more cases than not, before they actually could occur.

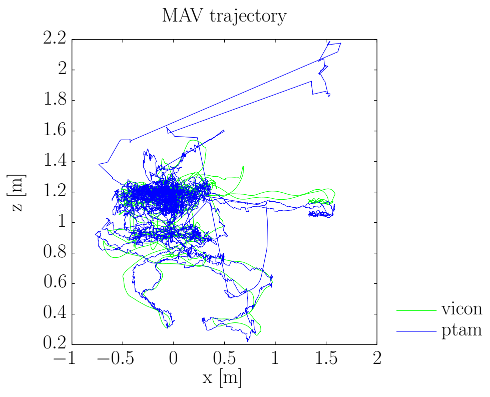
5.5.1.3 Localization and Mapping

In this subsection we analyze the accuracy of the visual localization by comparing the pose estimate of the PTAM system to the ground truth provided by the Vicon system. The main purpose of this part of the evaluation is to demonstrate that the pose estimation with PTAM is very close to the optimum and does not show any significant drift in this full loop experiment.

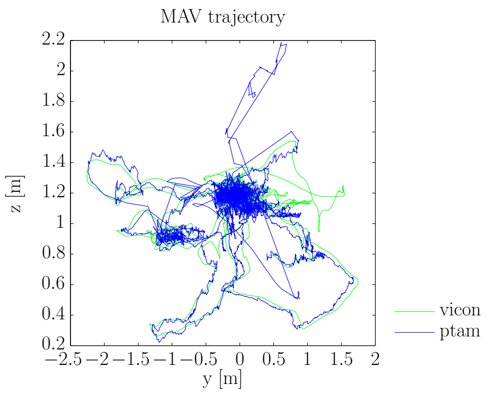
Figure 5.19 shows the trajectories of the MAV recorded by the PTAM system as well as the Vicon system. One can see that for most of the time the pose estimate of the PTAM system is very close to Vicon ground truth, but a few times the pose estimation of the PTAM system is totally wrong. The reason for this large error can be found in Figure 5.20. All major localization errors in this experiment were directly caused by severe lags. One of these lags lasted nearly 20 seconds. During this time the system did not receive any frames from the MAV and could not confirm its location visually. Note that after the visual input resumed, the system was in all cases able to fully recover its current pose. Despite from these major lag spikes the mean pose error was in an acceptable range with 0.25m and 0.08rad. The full error statistics of this experiment can be found in Table 5.7. During the experiment no significant drift could be detected. The error curve in Figure 5.20 shows that the system also successfully recognized its initial pose after a full 360° turn, which



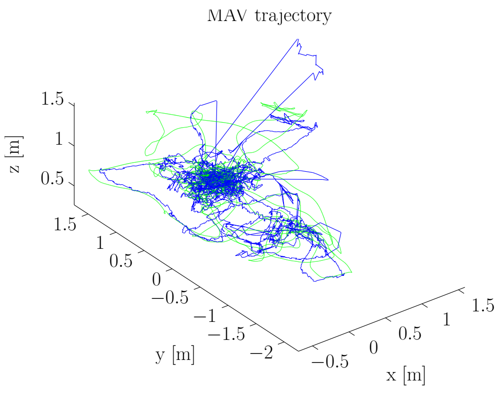
(a) xy view



(b) xz view



(c) yz view



(d) xyz view

Figure 5.19: Comparison of the trajectories recorded by the monocular localization (PTAM) and the ground truth (Vicon system).

results in a significant drop in the localization error at the end of the experiment.

error type	unit	RMSE	mean	STD	max
linear	[m]	0.334	0.248	0.225	1.953
angular	[rad]	0.109	0.078	0.076	0.876

Table 5.7: Localization error statistics. The error for the linear position (xyz) and the angular position (yaw) have been treated separately. The error was calculated by comparing the PTAM pose estimate to the Vicon ground truth.

In Figure 5.21 we compare the sparse reconstruction to the 2D ground truth plan of the room. From this comparison it is possible draw several conclusions. First of all, the system

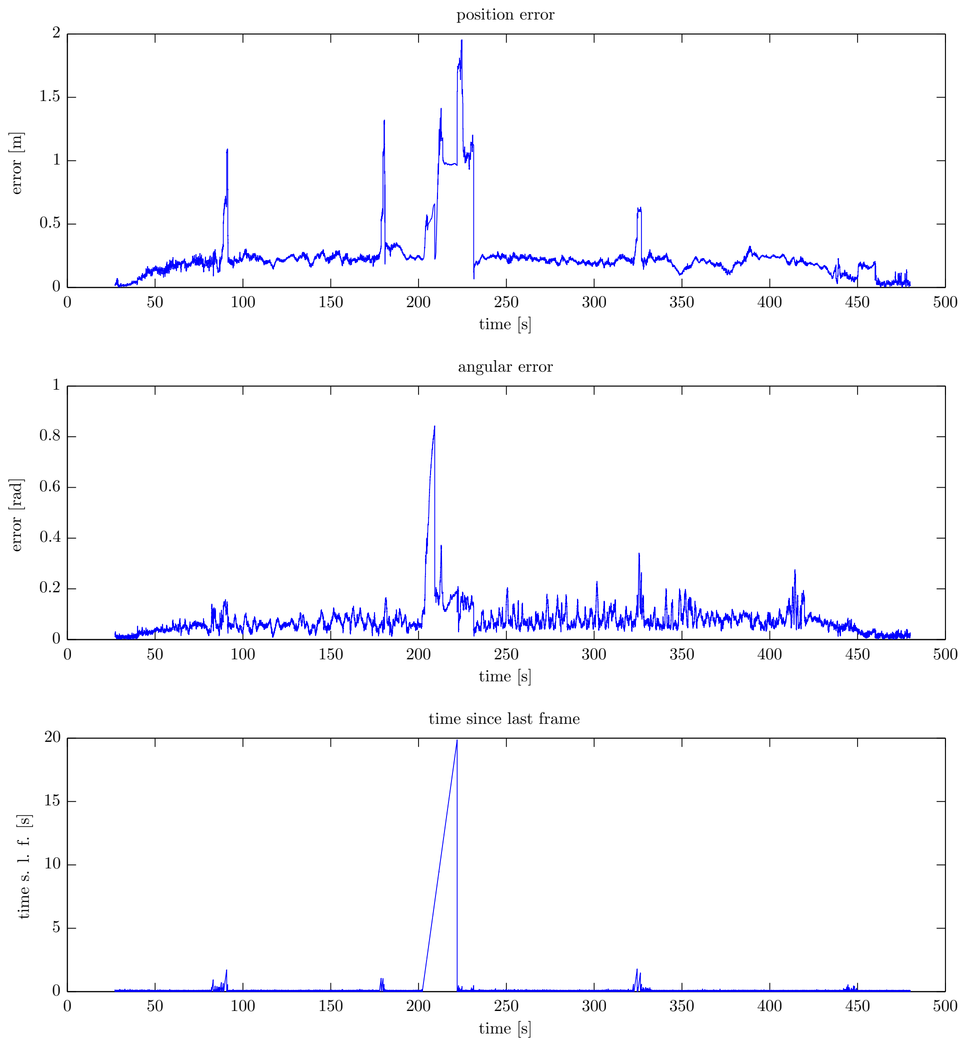


Figure 5.20: Localization error changes during the experiment. The error was calculated by comparing the PTAM pose estimate to the Vicon ground truth. Top: Linear error. Middle: Angular error. Bottom: Time since the last frame from the MAV reached the system. Note that all major localization error peaks are directly caused by severe communication lags.

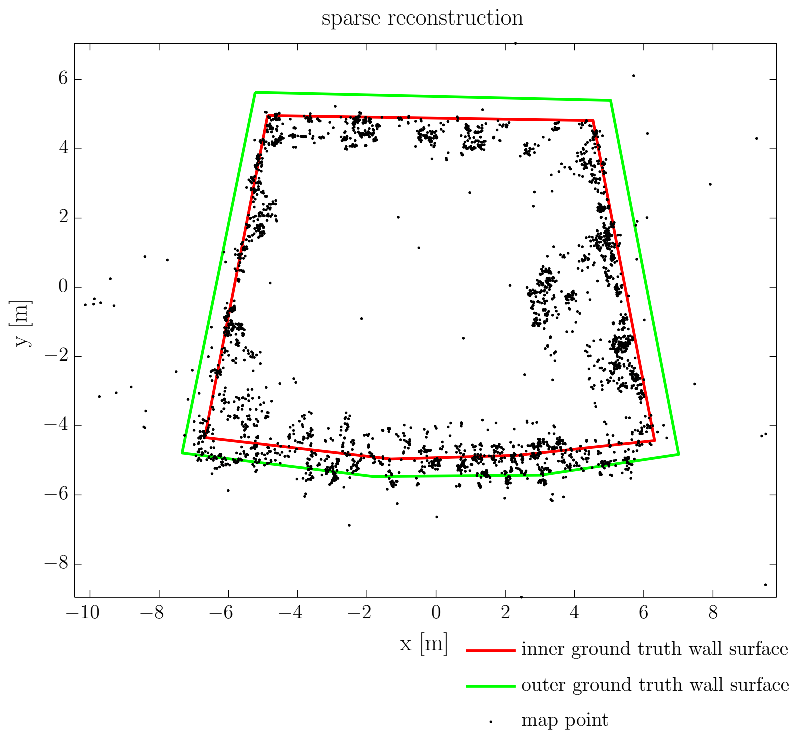


Figure 5.21: Sparse reconstruction compared to 2D ground truth. The black dots are the 3D map points of the point cloud after the full 360° turn. The red and green lines represent the inner and outer wall surface of the laboratory according to the building floor plan. For the alignment of the wall ground truth with the point cloud only translation and rotation were taken under consideration to maintain the scale. Note that the wall in the bottom of the figure contains a lot of windows which have a sun blind several centimeters away from the outer wall surface.

was able to successfully close the loop after the 360° turn. Secondly, the reconstruction successfully represents the skewed angles between the walls of the room. Thirdly, we can say that the scale of reconstruction is very close to the real scale. Note that some point groups along the bottom wall in the figure lie within as well as outside of the wall. These points are no outliers, but are valid map points on the windows and sun blind. The sun blind is positioned several centimeters away from the outer surface of the wall.

5.5.1.4 Conclusion

Simply turning is one of the hardest tasks for a monocular system as it needs a baseline between the camera positions to infer the depth of the scene. In this experiment we have shown that the proposed system is able to perform a full 360° degree turn in a barely textured scene without losing the visual localization. Due to the dynamic explorative nature of our approach the system was able to create a high quality map which allowed for a localization without any significant drift. Furthermore, this experiment once again demonstrates that our system is very robust against communication disruptions and can even recover from lags up to 20 seconds.

5.5.2 Multi-Level Experiment

The previous experiment mainly focused on the maintenance of the monocular localization and the generation of new useful map points. In contrast, this experiment additionally evaluates the capability of the system to avoid collisions. We have chosen this multi-level scene for two main reasons. Firstly, it shows that even in indoor environments airborne vehicles can have a significant advantage over ground robots due to their increased mobility. Secondly, the flight across the staircase is more challenging for the navigation module. The transition between the flat ground and the stairs causes the air stream beneath the MAV to change, which leads to turbulences that have to be compensated by the controller. While the staircase inherently increases the navigational imprecision, the frame around the stairs drastically limits the available space for navigation. In this experiment we show that our proposed system is able to safely navigate through such a challenging scene due to the detection of dangerous states with our collision probability measure and our recovery strategy. Furthermore, this experiment demonstrates once again that our localization quality measure and our point generation scheme enable a localization-safe navigation. To increase the expressiveness of this experiment, we repeated this experiment six times.

We split this section into four parts. Firstly, we start this section by providing the exact details of the experiment. Secondly, we demonstrate the functionality of our system in analyzing the MAV trajectory and the system logic. Thirdly, we use the repetitions of the experiment to estimate the scale uncertainty of the resulting reconstruction. We close this section by drawing conclusions from this experiment.

5.5.2.1 Experimental Procedure

For this experiment we initialize the system with the MAV positioned in the lower room with the camera facing the curtains as shown in Figure 5.22. For the initialization we use a baseline of $20cm$. Due to the limited space available, we only grant the system a very small initial free space cuboid with $1.2m$ in the x-direction (towards the curtains), $1.6m$ in the y-direction and $2m$ in the z-direction (from the floor upwards). The high-level objective in this scenario is to reach the destination pose up the stairs. The destination pose is defined as $(-0.5/-3.5/1.8/-1.57)$ in $(x/y/z/yaw)$-coordinates relative to the initialization pose. In plain words this means that the MAV should move across the staircase and face the antique desk as shown in Figure 5.22.

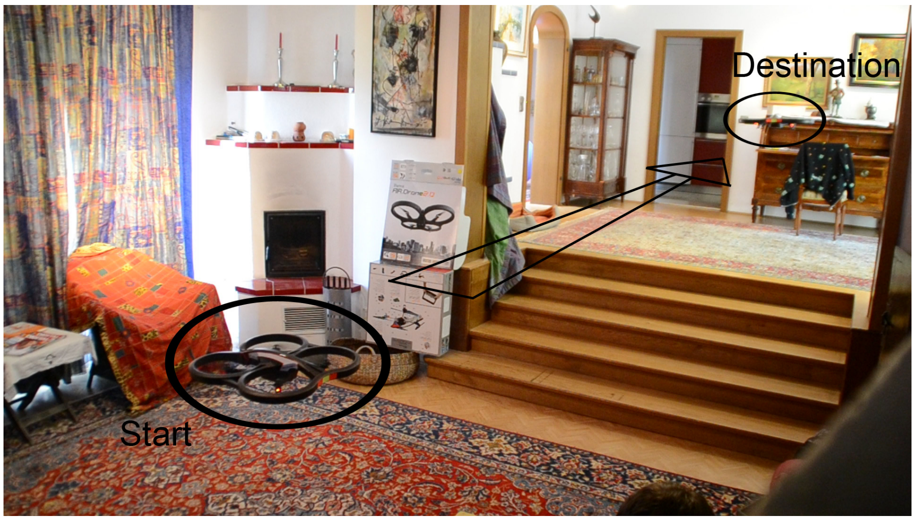


Figure 5.22: Mission for the multi level experiment. After initialization the MAV only has map points on the curtains. The mission is to safely move up the stairs and reach the destination facing the antique desk.

5.5.2.2 System Logic

The main aim of this subsection is to demonstrate the functionality of our approach in this challenging scenario through a detailed analysis of a single flight, and to review the most important statistics of all six flights.

Figure 5.23 visualizes the trajectory in sampling the MAV pose once per second. This figure connects all three representations of the same scene. The top most image illustrates the MAV trajectory in the "real" scene as a human would perceive it. In the middle one can see the sparse representation of the scene; the point cloud. The system uses

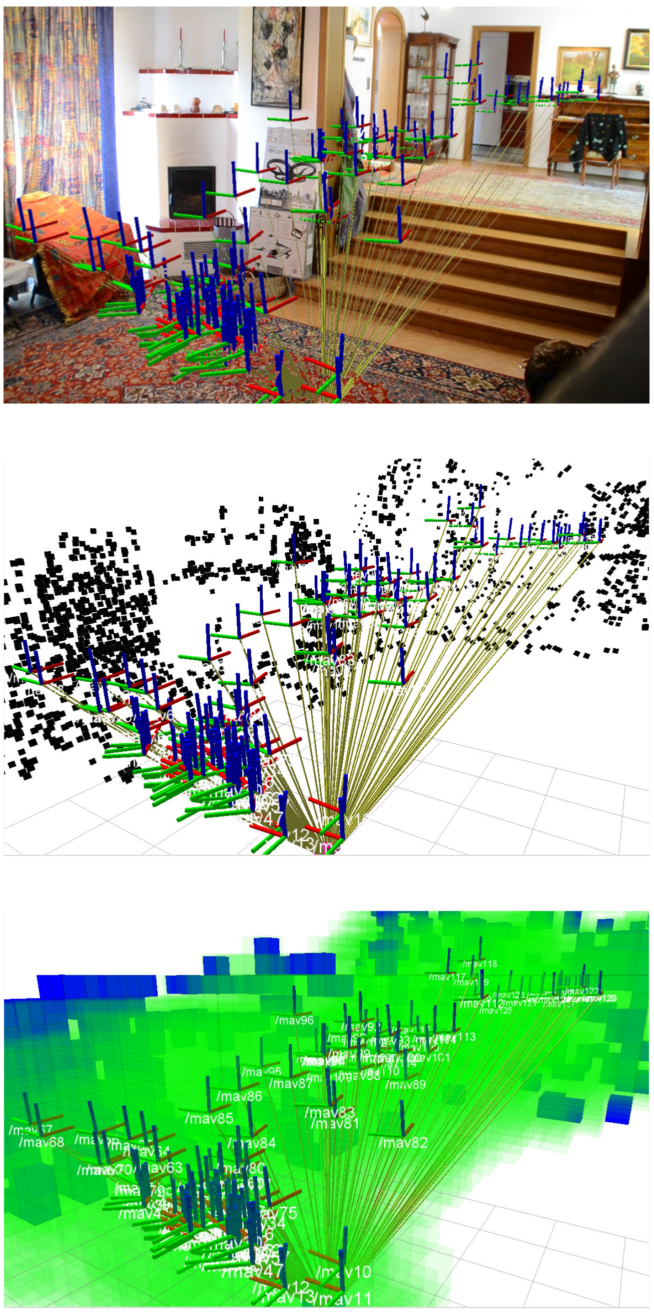
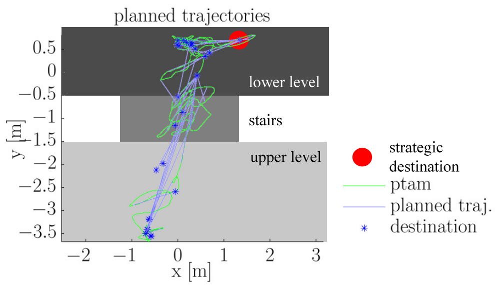


Figure 5.23: Trajectory sampled with a frequency of 1 Hz. Each image shows the same discretely sampled trajectory with a different background. The background in the top most image is the actual scene, whereas the center only shows the sparse reconstruction of the scene (point cloud). The bottom most image shows the volumetric representation of the scene. Blue means that a cell is occupied (obstacle) and transparent green means that a cells is very likely to be free of obstacles (free space).

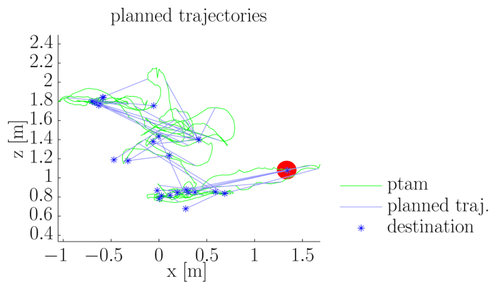
this representation to localize itself in the scene. The bottom most figure depicts the volumetric representation of the scene. This representation is used to detect obstacles and ensure a safe navigation through parts of the scene which are known to be free of obstacles. These parts of the scene are called "free space" and are visualized with transparent green cells. Note that the MAV never leaves the "free space" during the whole autonomous flight. This figure also demonstrates that prior to translating towards the destination, the system turns in the direction of the destination. This is done automatically to carve free space in the direction of the destination. Without this intentional free space carving the MAV would not be able to safely navigate towards the destination.

In Figure 5.24 we depict not only the trajectory recorded with PTAM, but also the generated short-time plans which influence the trajectory. From the figure we can draw several conclusions. Firstly, the nadir view in Figure 5.24a shows that the strategic destination (red dot) does not help to close the distance to the destination, but instead helps to generate new relevant points in the room upstairs. Secondly, one can see in Figure 5.24a that the transition between the flat surface of the room and the steep structure of the stairs causes great navigational problems for the MAV (y value between -0.5 and -1). During the navigation across the narrow path above the stairs (y value between -0.5 and -1.5) the turbulences cause the system to enter the CA state as it gets too close to walls on the side. This leads to the eight trajectories leading towards the pose at $(0.42/-0.06/1.4)$ as this pose is the safest keyframe pose in the neighborhood with best localization quality. Note that this pose does not necessarily have to be reached, but rather defines a safe recovery trajectory until the overall state of the system is safe enough to resume to the normal execution. This recovery strategy ensures that the system does not lose the visual localization while moving to a safer position.

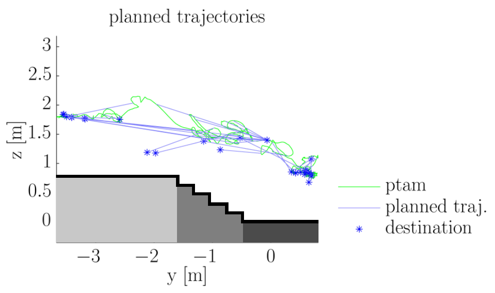
Figure 5.25 illustrates the changes in the system state during the experiment. The top most chart shows that the system suffered from severe lags in the interval between 80 and 90s, which did not pose a serious problem for our system. If we take a look at the chart with the collision probability, we notice a major spike at the time of 48 seconds. This spike was caused by an outlier which could be erased from the probabilistic map in less than a second after its appearance. During the navigation above the staircase (approximately 95 to 125s) the turbulences cause the system to enter the CA state nine times. This behavior keeps the MAV always far enough from any obstacle to avoid collisions. The localization quality only drops one time at 62s at the end of the 90° rotation, which caused the system to switch into the SE mode to generate new points in the room upstairs.



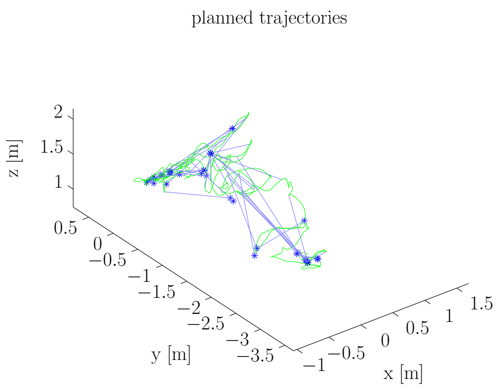
(a) xy view



(b) xz view



(c) yz view



(d) xyz view

Figure 5.24: Short time plans and the trajectory recorded by PTAM. The asterisks symbolize the position of the current *intermediate* destination. The blue lines connect the current pose of the MAV at the time of the destination generation to the related destination. They can be seen as ”ideal” trajectories. The green line is the effectively recorded trajectory by the PTAM system. In (a) and (b) we marked one destination with a red dot. This type destination is called ”strategic” as it does not help to close the distance to the *user defined* destination, but has the purpose of generating new useful map points. In (a) and (c) we additionally depict the floor levels and stairs to make the figures more comprehensible.

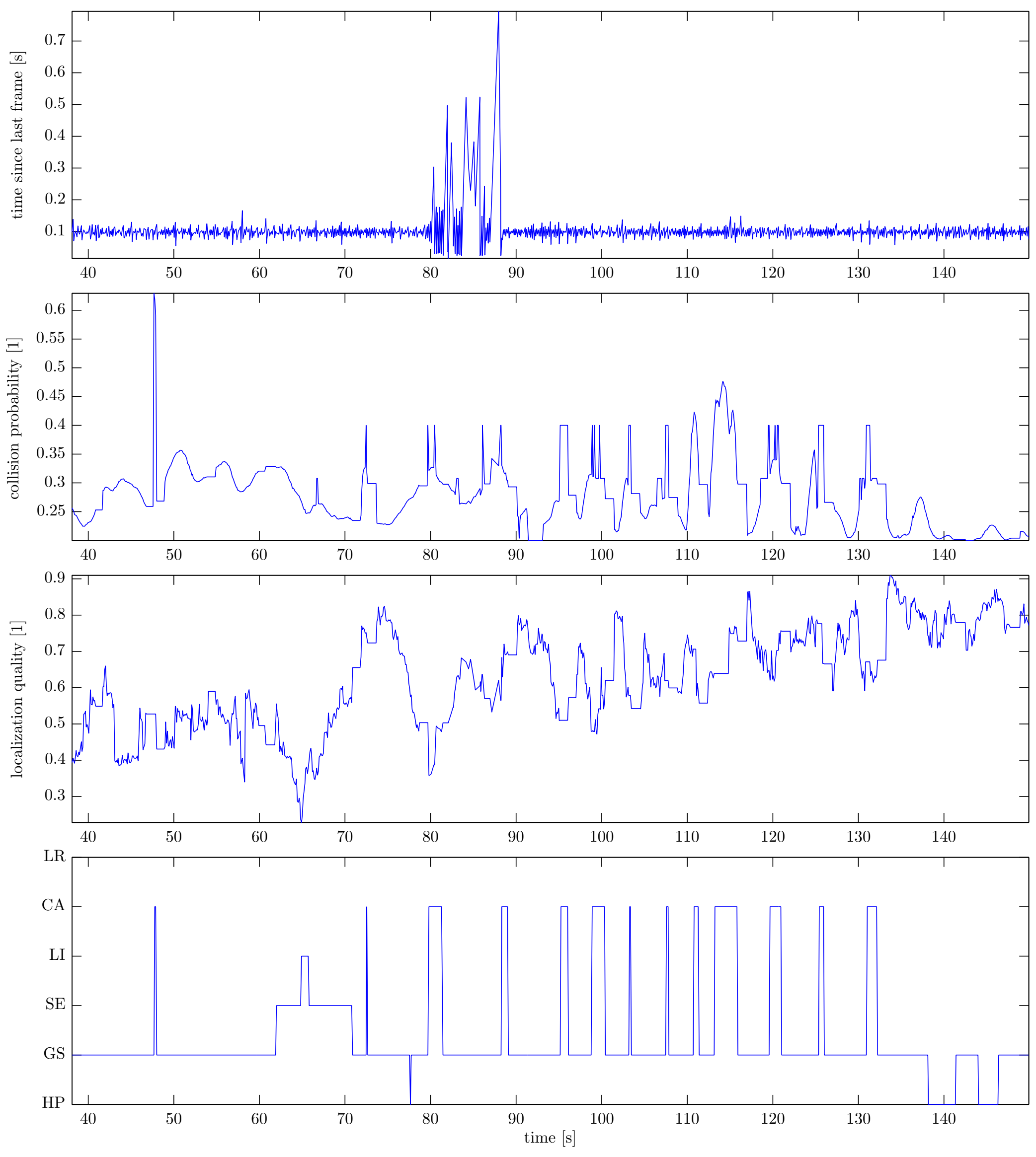


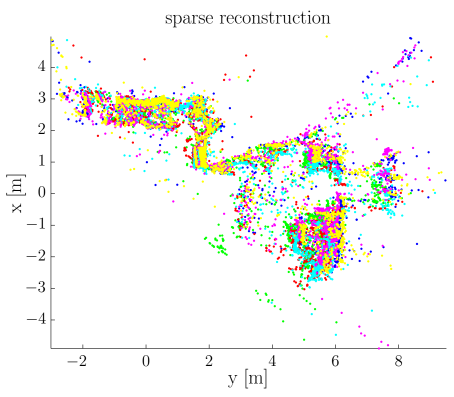
Figure 5.25: System state analysis. Row 1: The time since the last frame is used to detect lag spikes. Note that during this experiment the system suffered from serious communication problems in the interval of 80 to 90s. Row 2: Collision probability over time. Row 3: Localization quality changes during the experiment. Row 4: System mode changes over time. The top 3 modes are emergency responses, whereas the others are part of the explorative navigation.

In the selected flight it took the MAV approximately 100 seconds after the receipt of the destination pose to reach this destination. Overall six flights it took the MAV on average 187.3 seconds to reach the destination, with a standard deviation of 68.7 seconds. In all flights the MAV successfully explored all relevant parts of the scene and managed to reach the destination while keeping the visual localization steady at all times. On average the system entered 1.33 times the SE mode to generate new points (STD: 1.03). Our collision probability measure enabled our system to detect dangerous states and helped to keep the MAV far away from any obstacles while always respecting the need for the visual localization. In all experiments the MAV was able to reach the destination without any collision or loss of the visual localization.

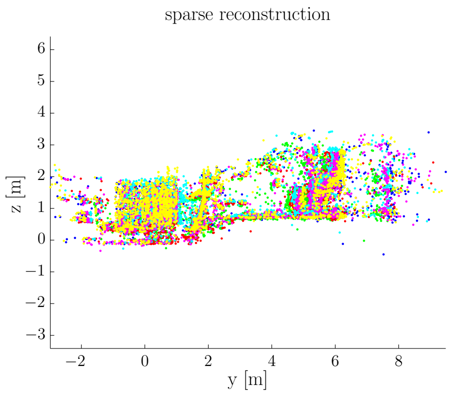
5.5.2.3 Scale Uncertainty

In this part of the analysis we use the repetition of the experiment to draw conclusions about the scale uncertainty of the reconstruction. For this purpose we consider the resulting point clouds of all six experiments as depicted in Figure 5.26 . If we take a look at Figure 5.26, we can see that all reconstructions agree on the same topology of the scene. The reconstruction of the room on the lower level ($y < 2$), which is close to the initial pose of the MAV, is nearly the same for all flights. All reconstructions agree on the same level for the upper floor, but the wall on the upper floor ($y \in [5, 6]$) shows a great variation in its depth value.

This motivated us to take a closer look at the variation of this wall. Note that the wall in question is perpendicular to the wall with the curtains in the lower room which was used for the initialization. To analyze the variation of the wall depth we only consider a part of the scene which only contains features on this specific wall and no other objects. In this scene this corresponds to the volume of $x \in [-2, 0]$, $y \in [5, 6.5]$ and $z \in [2.3, 3]$, which we displayed in Figure 5.27. Using this representation it is very easy to see that the individual reconstructions do not agree on the exact position of the wall. The average depth value (y-axis) of the wall is $5.98m$ with a standard deviation of $0.19m$. This means that the relative depth uncertainty of the wall is 3.2 percent, which can be seen as an indicator for the scale uncertainty of the resulting reconstruction. Note that this uncertainty does not cause any problems for the navigation per se, but can lead to problems if the *client defined* destination does not respect this scale uncertainty. In the worst case this can lead to *client defined* destinations inside an obstacle, which then of course can never be reached by the MAV.



(a) xy view



(b) yz view

Figure 5.26: Resulting point clouds of all six flights. The point cloud of each flight is depicted in a different color. Note that we flipped the y-axis in all figures to create a view which corresponds to the real world. Thus, (a) can be seen as a top-down view and (b) looks in the direction in which the system was initialized.

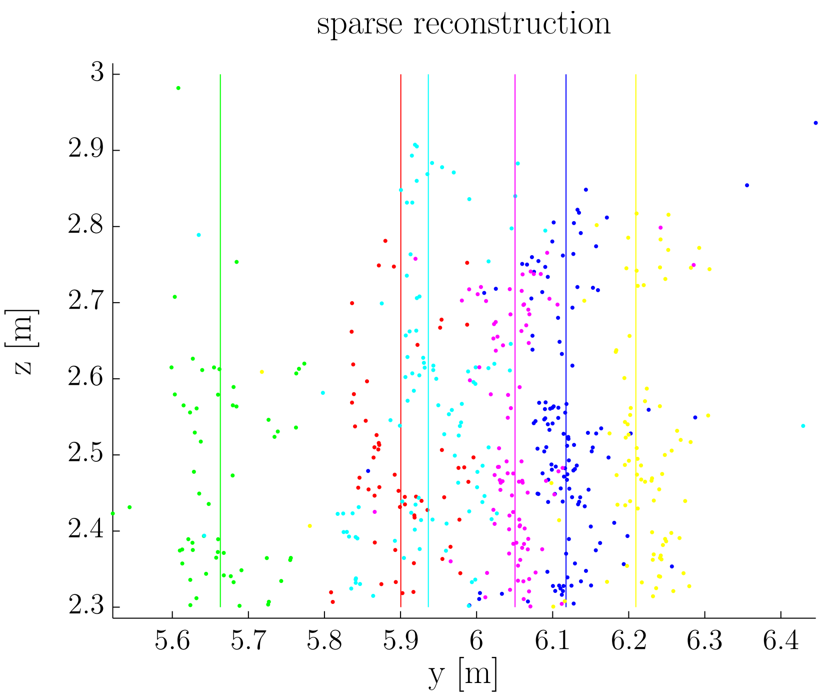


Figure 5.27: Depth uncertainty of the wall in the upper room. The colored dots represent the map point on the wall. All dots of a specific flight have the same color. The colored lines represent the mean depth of the wall.

5.5.2.4 Conclusion

With this scenario we have shown that our system is capable of autonomous navigation through an unknown complex environment. Even in the situation of severe navigational disturbances the system is not only able to avoid collisions, but also to maintain the visual localization at all times due to the inherent robustness of our proposed measures. Through repetition of the experiment, we estimated the scale uncertainty of reconstruction to be under four percent. This scale uncertainty is too small to have an impact on the navigational behavior of our system, but should be respected during the creation of *client defined* destination poses.

5.6 Summary

In this section we did not only demonstrate the overall functionality of our system for the task of active monocular localization, but also evaluated the individual modules of our system as independently as possible.

In our first series of experiments we showed that the point recognition probability of the tracking system (PTAM) is strongly influenced by viewpoint changes. For a change in the viewing angle we found that the recognition probability significantly declines above an angle of 30° and drops below 25 percent above 70°. We also showed that the relative scale change of a feature point can only be handled within a certain range. This range was proven to vary strongly depending on the scale at which the feature was originally extracted. In our experiments the PTAM system was able to handle a decrease of the scale (moving farther away) much better than increasing the scale (moving closer). The outcome of this experiment enabled us to device a realistic but still computationally cheap model of the point recognition probability.

The main purpose of our second series of experiments was to analyze and document the performance of the navigational module in combination with PTAM and the AR.Drone 2.0. The experiments have shown that navigation accuracy in the vertical direction is significantly higher than in the horizontal direction. This leads to the conclusion that an ellipsoidal metric, as proposed in this work, is better suited for the task of collision avoidance than a simple Euclidean metric. Our experiments show that the mean navigational error during the execution is below 14cm, but can under certain circumstances even exceed 0.5m. Furthermore, we measured a mean time of 10 seconds to close a distance of 20cm towards a new destination pose. The experiments confirm that the reaction time as well as the navigational precision of the system are influenced by the frequency and duration of the communication lags. Due to these unpredictable communication disruptions, it is not possible to make any guarantees regarding the collision avoidance.

In our third experiment we forced our system into extreme situations, which led to a loss of the visual localization. In any case it was possible to predict the loss of the visual localization with our novel localization quality measure. At the time of the loss of the visual localization the localization quality was always clearly below 20 percent.

In our final series of experiments we demonstrate the capabilities of our approach in two different scenarios. In the first scenario we showed that our active system was able to autonomously perform one of the hardest tasks for a monocular system. Our system successfully completed a full $360°$ degree turn in a barely textured scene without losing the visual localization although it suffered from severe communication disruptions of up to 20 seconds. Due to the dynamic explorative nature of our approach the system was able to create a high quality map which allowed for an automatic loop closure. In the second scenario we demonstrated that our system is able to safely explore and navigate through complex indoor scenes containing multiple floor levels. This experiment showed that even in the situation of severe navigational disturbances the system is able not only to avoid collisions but also to maintain the visual localization at all times due to the inherent robustness of our proposed measures.

In the following section we will conclude this work in summarizing the most important facts and providing an outlook to our future work.

Chapter 6

Conclusion and Future Work

In this thesis we bridge the gap between monocular SLAM and autonomous robotic systems. This bridge is build on the solid pillars of our novel measures which enable the system to evaluate the localization stability, the likelihood of generating new points as well as the physical safety for arbitrary camera and MAV poses. All that is needed for the calculation of these measures is the information which is available in every bundle adjustment based monocular SLAM approach; i.e. a sparse reconstruction of the scene (map points), the set of cameras poses which observe the map points, the 2D feature points and the information which map point was observed from which camera.

Using our novel measures we developed a monocular MAV system which is capable of autonomous way-point navigation in unknown environments. The resulting system respects the constraints for localization as well as physical safety along every planned trajectory. To ensure a localization safe navigation the system does not only avoid states which could lead to a loss of the visual localization, but actively generates new points so that the way-points can be safely reached.

In our experiments we demonstrate the effectiveness of our novel measures and capabilities of our autonomous explorative navigation approach. We show that our system can autonomously navigate between way-points in challenging unknown environments while maintaining the visual localization at all times. Through the explorative nature of our approach, we are able to complete tasks like a full $360°$ turn in a sparsely textured environment which is one of the hardest tasks available for a monocular system. Furthermore, we demonstrate that our navigation approach as well as our novel measures are very robust against a high degree of navigational imperfection which suggests that our approach is well suited to cope with changing dynamics and external influences such as wind and turbulences.

Future Work. In our future work we want to combine our explorative navigation approach with a Next-Best-View algorithm and thus develop a fully autonomous exploration system which does not require any exteroceptive sensors except from a monocular camera. To improve the scalability of our approach, it is necessary to remove the cubic memory consumption of the volumetric representation. This can be very likely achieved through the construction of a surface mesh. Up to now the main purpose of surface meshes is to model solid objects, but in order to achieve navigational safety its is necessary to explicitly model unknown parts of the scene. Such mesh, which yet has to be proposed, could reduce the memory consumption drastically and concurrently provide an unlimited resolution for obstacle avoidance.

Appendix A

Acronyms

List of Acronyms

AGAST	Adaptive and Generic Accelerated Segment Test
AR	Augmented Reality
BA	Bundle Adjustment
BRIEF	Binary Robust Independent Elementary Features
CA	Collision Avoidance
CPU	Central Processing Unit
DoG	Difference of Gaussian
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FPS	Frames Per Second
FOV	Field Of View
GBA	Global Bundle Adjustment
GPS	Global Positioning System
GPU	Graphics Processing Unit
GS	Goal-Striving
HDD	Hard Disk Drive
HP	Hold Position
IR	InfraRed
IMU	Inertial Measurement Unit
LBA	Local Bundle Adjustment
LI	Localization Improvement

LR	Localization Recovery
MAV	Micro Aerial Vehicle
MLE	Maximum Likelihood Estimator
MSER	Maximally Stable Extremal Region
NBV	Next-Best-View
PRM	Probabilistic RoadMap
PTAM	Parallel Tracking and Mapping
RAM	Random-Access Memory
RANSAC	Random Sample Consensus
RGB	Red, Green, Blue (color space)
RGBD	Red, Green, Blue, Depth
RMS	Root Mean Square
RMSE	Root Mean Square Error
RRT	Rapidly-exploring Random Tree
ROS	Robot Operating System
SfM	Structure from Motion
SDK	Software Development Kit
SE	Strategic Exploration
SIFT	Scale Invariant Feature Transform
SLAM	Simultanious Localization and Mapping
SSD	Sum of Squared Differences
STD	STandard Deviation
SURF	Speeded Up Robust Features
SUSAN	Smallest Uni-Value Segment Assimilating Nucleus Test
ToF	Time-of-Flight
UAV	Unmanned Aerial Vehicle
USAN	Uni-Value Segment Assimilating Nucleus
VDD	Variable Depth Distribution
VSLAM	Visual SLAM
WiFi	Wireless Fidelity
ZMSSD	Zero-Mean Sum of Squared Differences

Bibliography

[1] 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 *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

[2] 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. *The International Journal of Robotics Research*, 31(11):1320–1343.

[3] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110:346–359.

[4] Beder, C. and Steffen, R. (2006). Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence. In *Proceedings 28th DAGM Pattern Recognition Symposium*, pages 657–666, Berlin, DE.

[5] Bontemps, A., Vanneste, T., Paquet, J.-B., Dietsch, T., Grondel, S., and Cattan, E. (2013). Design and performance of an insect-inspired nano air vehicle. *Smart Materials and Structures*, 22(1).

[6] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: binary robust independent elementary features. In *Proceedings 11th European Conference on Computer Vision*, Hersonissos, Greece.

[7] Cornelis, N. and Van Gool, L. (2008). Fast scale invariant feature detection and matching on programmable graphics hardware. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, Anchorage, AK.

[8] Crowley, J. (1989). World modeling and position estimation for a mobile robot using ultra-sonic ranging. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 674 – 680, Scottsdale, Arizona.

[9] Daftry, S., Maurer, M., Wendel, A., and Bischof, H. (2013). Flexible and User-Centric Camera Calibration using Planar Fiducial Markers. In *Proceedings 24th British Machine Vision Conference*, Bristol, GB.

[10] Davison, A. J. (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings 9th International Conference on Computer Vision*, Nice, FR. IEEE Computer Society.

[11] Davison, A. J. and Murray, D. W. (1998). Mobile Robot Localization using Active Vision. In *Proceedings 5th European Conference on Computer Vision*, pages 809–825, Freiburg, DE.

[12] 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*, 29:1052–1067.

[13] Donoser, M. and Bischof, H. (2006). Efficient maximally stable extremal region (MSER) tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 553–560, New York,NY.

[14] Dunn, E. and Frahm, J.-M. (2009). Next best view planning for active model improvement. In *Proceedings 20th British Machine Vision Conference*, pages 1–11, London, GB.

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

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

[17] Eade, E. and Drummond, T. (2007). Monocular SLAM as a Graph of Coalesced Observations. In *Proceedings 11th International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil.

[18] Engel, J., Sturm, J., and Cremers, D. (2012a). Accurate figure flying with a quadrocopter using onboard visual and inertial sensing. In *Proceedings of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS).*

[19] Engel, J., Sturm, J., and Cremers, D. (2012b). Camera-based navigation of a low-cost quadrocopter. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal.

[20] Eudes, A. and Lhuillier, M. (2009). Error Propagations for Local Bundle Adjustment. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida.

[21] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

[22] Förstner, W. (2005). Uncertainty and projective geometry. In Bayro-Corrochano, E., editor, *Handbook of Geometric Computing.*, pages 493–535. Springer.

[23] 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 *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal.

[24] Geraerts, R. and Overmars, M. (2004). A comparative study of probabilistic roadmap planners. In Boissonnat, J.-D., Burdick, J., Goldberg, K., and Hutchinson, S., editors, *Algorithmic Foundations of Robotics V*, volume 7 of *Springer Tracts in Advanced Robotics*, pages 43–58. Springer Berlin Heidelberg.

[25] Grzonka, S., Grisetti, G., and Burgard, W. (2009). Towards a navigation system for autonomous indoor flying. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan.

[26] Haner, S. and Heyden, A. (2011). Optimal view path planning for visual SLAM. In *Proceedings of the 17th Scandinavian Conference on Image Analysis*, SCIA'11, pages 370–380, Ystad, Sweden. Springer-Verlag.

[27] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings 4th Alvey Vision Conference*, pages 147–151, Manchester, GB.

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

[29] He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a gps-denied environment. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, California.

[30] Holmes, S. A., Sibley, G., Klein, G., and Murray, D. W. (2009). A relative frame representation for fixed-time bundle adjustment in SFM. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 2264–2269, Kobe, Japan.

[31] Hoppe, C., Wendel, A., Zollmann, S., Pirker, K., Irschara, A., Bischof, H., and Kluckner, S. (2012). Photogrammetric camera network design for micro aerial vehicles. In *Proceedings of the Computer Vison Winterworkshop*, Mala Nedelja, Slovenia.

[32] 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*, 34:189–206. Software available at `http://octomap.github.com`.

[33] Hrabar, S. (2008). 3D Path Planning and Stereo-based Obstacle Avoidance for Rotorcraft UAVs. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France.

[34] Kanatani, K. (2004). Uncertainty modeling and model selection for geometric inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26:2004.

[35] Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. In *Robotics: Science and Systems*.

[36] Katusic, M. (2012). Human-inspired visual servoing for automatic take-off, hovering and landing of MAVs. Master's thesis, Institute for Computer Graphics and Vision, Graz University of Technology.

[37] Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, Nara, Japan.

[38] Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *Proceedings 10th European Conference on Computer Vision*, pages 802–815, Marseille, France.

[39] Klein, G. and Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *Proceedings 8th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Orlando, FL.

[40] Kushleyev, A., Mellinger, D., and Kumar, V. (2012). Towards a swarm of agile micro quadrotors. In *Robotics: Science and Systems*.

[41] Leonard, J. J. and Durrant-Whyte, H. F. (1991a). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7:376–382.

[42] Leonard, J. J. and Durrant-Whyte, H. F. (1991b). Simultaneous map building and localization for an autonomous mobile robot. pages 1442–1447, Osaka, Japan.

[43] Lewis, M. A. (1997). Visual navigation in a robot using zig-zag behavior. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, pages 822–828.

[44] Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 2(21):224–270.

[45] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

[46] Lupashin, S. and D'Andrea, R. (2012). Adaptive fast open-loop maneuvers for quadrocopters. *Auton. Robots*, 33(1-2):89–102.

[47] Mair, E., Hager, G. D., Burschka, D., Suppa, M., and Hirzinger, G. (2010). Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings 11th European Conference on Computer Vision*, Hersonissos, Greece.

[48] Matas, J., Chum, O., Martin, U., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings 13th British Machine Vision Conference*, pages 384–393, Cardiff, GB.

[49] Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525, Shanghai, China.

[50] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, 65:43–72.

[51] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598.

[52] Morris, W., Dryanovski, I., and Xiao, J. (2010). 3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids. In *RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras*.

[53] Nelson, R. C. and Aloimonos, J. (1989). Obstacle avoidance using flow field divergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1102–1106.

[54] Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *Proceedings 13th International Conference on Computer Vision*, Barcelona, Spain.

[55] Oxford University Press (2013). lidar: definition of lidar in Oxford dictionary - American English (US). *http://oxforddictionaries.com/us/definition/american_english/lidar.*

[56] Ozog, P. and Eustice, R. M. (2013). On the importance of modeling camera calibration uncertainty in visual slam. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 3762–3769, Karlsruhe, Germany.

[57] Parrot (2013). AR.Drone 2.0. Parrot new wifi quadricopter- Specifications. *http://ardrone2.parrot.com/ardrone-2/specifications/.*

[58] 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 *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany.

[59] Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proceedings 10th International Conference on Computer Vision*, volume 2, pages 1508–1511, Beijing, China.

[60] Rumpler, M., Irschara, A., and Bischof., H. (2011). Multi-View Stereo: Redundancy Benefits for 3D Reconstruction. In *Proceedings of the 35th Workshop of the Austrian Association for Pattern Recognition (AAPR/OAGM)*, Graz, Austria.

[61] Schauwecker, K., Ke, N. R., Scherer, S., and Zell, A. (2012). Markerless visual control of a quad-rotor micro aerial vehicle by means of on-board stereo processing. In *Autonomous Mobile Systems (AMS)*, pages 11–20.

[62] Shen, S., Michael, N., and Kumar, V. (2011). Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

[63] Shen, S., Michael, N., and Kumar, V. (2012). Autonomous Indoor 3D Exploration with a Micro-Aerial Vehicle. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota.

[64] Shi, J. and Tomasi, C. (1994). Good features to track. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA.

[65] Sinha, S. N., Frahm, J.-M., Pollefeys, M., and Genc, Y. (2006). GPU-based Video Feature Tracking and Matching. Technical report.

[66] Smith, R. and Cheeseman, P. (1986). On the Representation and Estimation of Spatial Uncertainty. *International Journal of Robotics Research*, 5(4):56–68.

[67] Smith, R., Self, M., and Cheeseman, P. (1986). Estimating uncertain spatial relationships in robotics. In *Proceedings of the Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pages 267–288, Corvallis, Oregon. AUAI Press.

[68] Smith, S. and Brady, J. (1997). SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78.

[69] Sobey, P. J. (1994). Active navigation with a monocular robot. *Biological Cybernetics*, 71:433–440.

[70] Stewénius, H., Engels, C., and Nistér, D. (2006). Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294.

[71] Strasdat, H., Davison, A., Montiel, J., and Konolige, K. (2011). Double window optimisation for constant time visual SLAM. In *Proceedings 13th International Conference on Computer Vision*, Barcelona, Spain.

[72] Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010). Real-time monocular SLAM: Why filter? In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 2657–2664, Anchorage, Alaska.

[73] Sugaya, Y. and Kanatani, K. (2003). Outlier removal for motion tracking by subspace separation. *IEICE Trans. Information and Systems*, 86:1095–1102.

[74] Thrun, S. (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141.

[75] Trummer, M., Munkelt, C., and Denzler, J. (2010). Online Next-Best-View Planning for Accuracy Optimization Using an Extended E-Criterion. In *Proceedings of 20th International Conference on Pattern Recognition*, pages 1642–1645, Instanbul, Turkey.

[76] Wagner, D. and Schmalstieg, D. (2007). ARToolKitPlus for Pose Tracking on Mobile Devices. In *Proceedings 12th Computer Vision Winter Workshop*, pages 139–146, St. Lambrecht, Austria.

[77] Weiss, S., Achtelik, M. W., Lynen, S., Chli, M., and Siegwart, R. (2012). Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 957–964, St. Paul, Minnesota.

[78] Weiss, S., Scaramuzza, D., and Siegwart, R. (2011). Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments. *Journal of Field Robotics*, 28(6):854–874.

[79] Wendel, A., Maurer, M., Graber, G., Pock, T., and Bischof, H. (2012a). Dense Reconstruction On-the-Fly. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI.

[80] Wendel, A., Maurer, M., Katusic, M., and Bischof, H. (2012b). Fuzzy visual servoing for micro aerial vehicles. In *Proceedings of the Austrian Robotics Workshop*, Graz, Austria.

[81] Wenhardt, S., Deutsch, B., Angelopoulou, E., and Niemann, H. (2007). Active Visual Object Reconstruction using D-, E-, and T-Optimal Next Best Views. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, Minneapolis, Minnesota.