Fast and Robust Localization and Mapping on Micro Air Vehicles

Lorenz Meier

Master Thesis
March 2011

Dr. Friedrich Fraundorfer / Prof. Dr. Marc Pollefeys
Abstract

This thesis provides a robust localization and mapping pipeline for micro air vehicles (MAVs). It is designed for planar maps and known vertical direction. It enables fast onboard localization and robust measurements by using a tight coupling of vision and inertial measurements. It contributes a new rotation invariant feature extraction pipeline for micro air vehicles with the unique capability of reducing the feature extraction time by 50% with tight vision-IMU fusion. The pipeline allows to efficiently create local maps during localization. The localization and feature extraction pipelines are suited for cooperative mapping and efficient transmission of the created map. The suitability for applications in the field is demonstrated in benchmarks and with real flights of a quadrotor with onboard processing. The mapping approach is shown to work cooperatively in parallel on multiple vehicles.
Zusammenfassung

Contents

List of Figures vii

List of Tables ix

1. Introduction 1
   1.1. Application Areas ............................................. 1
   1.2. MAV specific challenges ...................................... 2

2. Requirements of Computer Vision on MAVs 5
   2.1. Critical properties for flight safety ......................... 6
   2.2. Position Accuracy, Update Rate and Time Delay .............. 6
   2.3. Processing Speed .............................................. 6
   2.4. Multi-Sensor Fusion .......................................... 7

3. Related Work 9
   3.1. Micro Air Vehicles ........................................... 9
   3.2. Computer Vision on Micro Air Vehicles ..................... 10
   3.3. Inertially Aided Localization and Odometry ................. 10

4. Overview of the Approach 13
   4.1. MAV Specific Computer Vision ............................... 13
   4.2. Vehicle Architecture ......................................... 14
   4.3. Vision-IMU Synchronization and Fusion ...................... 16
   4.4. Localization .................................................. 17
       4.4.1. Fronto-parallel View ................................... 17
       4.4.2. Feature Extraction and Matching ....................... 17
## Contents

4.4.3. Localization with 2pt Algorithm .......................... 18
4.5. Local Mapping and Swarm Mapping .......................... 18

5. **Feature Extraction** ........................................... 19
5.1. Overview .................................................. 19
5.2. SIFT ....................................................... 20
5.3. SURF ....................................................... 20
5.4. BRIEF and CENSURE ........................................ 21
   5.4.1. CENSURE ............................................ 21
   5.4.2. BRIEF Descriptor ....................................... 23
   5.4.3. Performance Evaluation ................................. 23

6. **Natural Feature based Localization** .......................... 27
6.1. Establishing 3D-2D correspondences .......................... 27
   6.1.1. Keypoint localization .................................. 27
   6.1.2. Orientation assignment ................................ 28
   6.1.3. BRIEF Extraction ...................................... 28
   6.1.4. Matching ................................................ 29
   6.1.5. Roll-pitch compensation ................................ 29
   6.1.6. Efficient linear-time Inlier Selection ................. 29
6.2. Absolute Pose from two 3D-2D Correspondences ............. 31
   6.2.1. Least-squares solution using sine-cosine constraint .. 32
   6.2.2. Least-squares solution using tangens substitution .. 33
   6.2.3. C/C++ Implementation .................................. 34

7. **Mapping and Optimization** .................................... 37
7.1. Position Refinement .......................................... 37
7.2. Hole Filling ................................................ 38
7.3. Flight Planning ............................................... 40
7.4. Multi-Vehicle / Distributed Mapping ........................ 40

8. **Experimental Results** .......................................... 41
8.1. Fully Autonomous Flight ..................................... 41
8.2. Single Vehicle Mapping ....................................... 43
8.3. Cooperative Mapping with multiple Vehicles ................. 43
8.4. Visualization and Flight Control ............................. 44
8.5. Additional Feature Extraction Samples ........................ 44

9. **Conclusion and Outlook** ....................................... 49

A. **Addition Technical Information** .............................. 51
   A.1. MAVLink communication protocol .......................... 51
   A.2. MAVCONN Aerial Middleware .............................. 51

**Bibliography** ................................................. 52
# List of Figures

4.1. Camera frame to Inertial Frame Relation ................. 14  
4.2. PXHAWK Cheetah Quadrotor, 10 inch version .............. 15  
4.3. pxIMU v2.5 Inertial Measurement Unit / Autopilot ........ 16  
4.4. Flight control pipeline .................................... 17  

5.1. Wavelet filters to calculate the gradient histogram .......... 22  
5.2. BRIEF gaussian distributed point comparison pattern .......... 23  
5.3. BRIEF feature extracted from image keypoint ................ 24  
5.4. Censure/BRIEF Extraction example with newly contributed orientation assignment ............................. 25  

6.1. Z estimations histogram .................................... 31  
6.2. X-Y-Z position estimate .................................... 35  
6.3. Yaw estimate ............................................. 36  

7.1. Output of 2pt localization / mapping pipeline ................. 39  

8.1. Fully autonomous hovering .................................. 42  
8.2. Localization output of autonomous hovering ................... 42  
8.3. Mapping results on outdoor gravel ......................... 43  
8.4. Mapping progress on aerial texture ........................... 45  
8.5. Cooperative Mapping, two vehicle maps shown side-by-side exchanging features ................................. 46  
8.6. QGroundControl 2pt GUI with realtime plot .................. 46  
8.7. Pipeline extracting features on gravel ....................... 47  
8.8. Pipeline extracting features on street ....................... 47  
8.9. Pipeline extracting features on grass ....................... 48
List of Figures
List of Tables

5.1. Runtime speeds of different feature extractors . . . . . . . . . 24
List of Tables
Introduction

Recent advances in mobile electrical energy storage, microminiature sensors and miniature computer design have enabled a new class of robots: Micro Air Vehicles. These tiny autonomous aircraft in the range of 0.3-0.5 meter wingspan or diameter currently represent the limit of miniaturization for autonomous flying vehicles capable of high-level onboard computer vision.

1.1. Application Areas

Micro Air Vehicles (MAV) offer a broad application spectrum. MAVs are not designed to carry heavy weights, however they can easily be equipped with cameras or other environmental sensors to perform tasks such as search and rescue, inspection of constructions and buildings or observation / mapping of large outdoor areas. Most of the current system designs are based on GPS control which offer autonomous control for outdoor flights but require a quite open environment with GPS reception. This kind of system is not capable of flying in cities with medium-sized buildings, both because GPS reception is reduced and because the accuracy of the GPS-based position control is not sufficient in proximity to buildings. Indoor flights or flights between buildings, trees or other obstacles are often not possible. MAVs can bridge this gap and explore dangerous, inaccessible and narrow spaces in GPS-denied indoor and outdoor environments by using computer vision.

One scenario for a MAV in a rescue task could be the following: A collapsed building, in which survivors are trapped. Instead of trying to get access with a
1. Introduction

A slow ground-based rescue robot, a helicopter sent in could bring back imagery of the interior within minutes. It can help civil engineers to visually assess the damage and rescue workers to find victims in need of help. It could thereby narrow down the search area and guide the rescue workers to the right location. It is also important to know the inner state of the building, to assess the damage and further risks to derive the best rescue strategy. Any ground-based rescue robot is very likely to get entangled in the rubble. The ground robot might even collapse the building completely by its weight and movements. A flying robot does not introduce these risks, and it also benefits from its vertical degree of freedom: It can evade obstacles on the ground by changing altitude and it can use vertical structures, such as elevator shafts, to access the lower levels. Since modern building structures often contain reinforced concrete, wireless transmission of the robot’s cameras is infeasible. In contrast to aerial vehicles, ground robots solve this issue by dragging a data link cable along, which however might trap the robot if it gets stuck. A more suitable remote camera device is a micro air vehicle, equipped with onboard cameras and autonomously exploring the collapsed structure. It brings within a few minutes after liftoff back the video data to its operator.

MAVs are in particular suitable for any visual inspection task involving the presence of the robot in a potentially fatal environment, such as visual inspection of nuclear plants, chemical facilities or semi-collapsed buildings. Because of the higher fidelity in comparison to a ground robot and the capability to land at points of interest and stay stationary for some time, the full range of visual inspection can be covered with a MAV. The system developed from 2009-2010 for this thesis also shows another important aspect: Micro air vehicles contain so few mechanical parts and electronics and are hence very affordable, that they can even be disposed after use in nuclear or chemical contaminated areas.

The advantages of the MAV approach even enhance the operational spectrum to tasks which have to be performed by human beings so far, like the inspection of wind engines. Instead of sending highly specialized staff to climb up the wind engine, the MAV can explore the situation autonomously and deliver video, pictures or sensor data to people on the ground.

1.2. MAV specific challenges

The MAV-size class poses new challenges in robotics not met previously. The need for fast processing, miniaturization and power efficiency present in this application area are challenging. This field therefore requires new solutions providing robust, high-speed position data, substantially faster and more robust than for ground robotics.

Up until now the majority of systems for localization and mapping was implemented following one of two design patterns: Either the technique tracked its
1.2. MAV specific challenges

position with a motion model and without a clear representation of 2D feature patches, such as MonoSLAM [DRMS07] and PTAM [KM07], or high-level visual features are extracted by using SURF [BETG08] or SIFT [Low04] and the 3D position of the camera is estimated from the global matched features. Due to the expensive SURF and SIFT descriptor extraction and matching, techniques based on extraction and matching were not able to provide the required realtime-speeds needed for micro air vehicles. Recently the PTAM algorithm was used on a MAV for onboard hovering [AAWS11], but without mapping or being used in a larger environment. PTAM requires significant longer time for localization after a tracking loss, which is undesirable in a flight environment. This master thesis addresses both issues with a new pipeline: By globally localizing the system for each frame, the problem of varying processing times of tracking approaches is solved. This is however only possible by employing new techniques for keypoint localization, feature extraction and matching which cut down the processing times of previous approaches using SURF/SIFT to onboard realtime speed. These steps typically consume the majority of the processing time, runtime optimization for a SLAM pipeline should therefore always focus on the keypoint extraction and matching first. For this very reason the global optimization of map feature positions with techniques such as bundle adjustment, particle filtering or Kalman filtering was not in the scope and not a goal of this work: The map optimization is a very active research field with many applicable contributions, but many suffer from too slow keypoint extraction and matching and/or do not provide simple yet fast localization. It therefore seems most appropriate to seek improvements in the flight capabilities of micro air vehicles in this area.

The presented results reduce the extraction and matching time for a medium-sized map from 250 ms (SURF) to 50 ms (this approach) on a typical onboard computer (Intel Core 2 DUO 1.86 GHz) and from 190 ms to 25 ms for the next-generation PIXHAWK platform (Intel i7 2.0 GHz). This is not only a reduction by a factor of five, but brings the position feedback from 4 Hz into the critical and needed 10-20 Hz range for autonomously controlled flight. Even more importantly, the presented results allow to treat each keypoint individually instead of relying on complex map structures involving keyframes, such as e.g. PTAM. This allows to scale this approach to multi-vehicle mapping and localization. As the map is not one large entity it can be easily segmented and transmitted/shared with other vehicles. As will be shown in chapter 7, this allows to employ heterogenous vehicle swarms. Since the utilized BRIEF [CLSF10] feature descriptor reduces the storage size by a factor of 8-16x compared to the state of the art, it is a fundamental contribution to further work on distributed / swarm localization and mapping.

The following chapter 2 introduces the MAV vehicle class and the resulting requirements for safe and stable autonomous flight using computer vision for localization. Chapter 3 gives an overview of previous work and discusses the shortcomings with respect to the MAV requirements. The following chapter
1. Introduction

4 gives an overview of the approach. Chapters 5 - 7 explain the algorithmic steps in detail. Finally chapter 8 provides experimental results and 9 summarizes the results and shows future work.
Requirements of Computer Vision on MAVs

Micro air vehicles are by definition not capable of carrying a large sensor or a sensor rig. Therefore it is a requirement of the platform to narrow down the sensors needed for navigation and the application tasks. Due to the limited onboard power, the sensing system should also consume a minimum in power. To optimally support these tasks, the MAV has to be able to detect the structure and the appearance of its environment. While the structure can be assessed with active scanning systems, such as laser scanners or phase array radar, both approaches do not provide the appearance, the surface texture. As a computer vision system can provide structure and texture from a single camera, this sensor platform is the minimal sensing device for a micro air vehicle.

Computer vision on micro air vehicles serves several major purposes:

- Localization in GPS-denied indoor and outdoor environments
- Collision avoidance
- Mapping
- Object recognition
- Other higher-level artificial intelligence tasks.
2. Requirements of Computer Vision on MAVs

2.1. Critical properties for flight safety

Of all these applications, the localization is the first and most fundamental step, since the craft is not capable of operating safely without knowing its exact position and speed. In contrast to a driving robot, which can simply stop once the position is lost, a micro air vehicle needs position feedback even just to hover on a spot. This is a critical and fundamental observation which should not be underestimated. This property also implies that many well-working and proven visual localization pipelines for driving robots cannot be simply ported to an MAV platform.

The requirements regarding the robustness and runtime speeds of localization on a micro air vehicle are substantially higher than on ground-based robots, such as the work of Konolige et al. [KAB+08]. Another field where real-time operation has been achieved is virtual reality with hand-held applications, e.g. PTAM from Klein et al. [KM07]. Although PTAM was used on micro air vehicles, it required severe optimizations and the disabling of the mapping functionality during flight to achieve real-time speeds suitable for MAVs.

2.2. Position Accuracy, Update Rate and Time Delay

Any input to the onboard position control system has to satisfy three main properties: Sufficient accuracy, sufficient update rate and a short enough time delay. The accuracy is not a major concern, as long as a noise model of the measurements exist. Estimation techniques such as Kalman filtering allow to recover the true position given the noise model. Modeling and analyzing the system dynamics allows to estimate the required update rate and maximum acceptable time delay in order to control the system. The dynamic model for the PIXHAWK Cheetah quadrotor created by Goppert and Hwang [GH11] and the subsequent analysis suggests that the system can be controlled, in the presence of noise, with a localization update rate above 6 Hz and a time delay not greater than 100 ms. Of course there is a dependency between localization, Kalman estimator and controller, but based on this rough estimate one can conclude that a localization pipeline should provide 10 Hz update rate for navigation in small indoor spaces.

2.3. Processing Speed

Micro air vehicles pose therefore three major challenges not found in ground robotics or hand-held computer vision applications: According to the literature presented in chapter 3, the achieved update rate should be around 10 Hz
2.4. Multi-Sensor Fusion

or higher with less than 100 ms time delay, the estimated pose has to be valid (or it’s invalidity has to be known) and the computational power is severely limited in comparison to standard computers. Recent general purpose graphics processing unit (GPGPU) approaches, such as GPU-SIFT [SFP06] showed that they can meet the high-speed requirements of realtime localization. In the evaluations of these approaches it is however often neglected that the speedup over pure CPU processing (which is often in the range of 10x-20x) comes for the cost of significantly increased power consumption. Current desktop GPUs consume up to 244W, which is more than a large MAV with more than 1 kg takeoff weight needs to hover. A typical notebook GPU consumes 20W minimum, on average 30W, which equals the total power consumption of all onboard electronics of a large MAV, including the onboard computer \(^1\).

These numbers show that GPGPU processing has not only a worse performance per watt ratio than CPU processing, but also that even mobile GPUs already exceed the power budget of such a craft.

2.4. Multi-Sensor Fusion

Since a very high level of robustness and safety is required on a micro air vehicle, it is beneficial to use all available sensors for the position estimation. Particularly the camera, the inertial measurement unit and barometric pressure measurements. Since the reliability of barometric pressure is very limited in indoor environments, this work focuses on vision-IMU fusion. To successfully fuse these sensors, a precise time base is necessary. Typical machine vision cameras such as the Point Grey Firefly MV \(^2\) do not provide a precise time information. Therefore the author has contributed in an earlier work [MTFP11] a system design capable of tightly synchronizing the system time base by tightly coupling the inertial measurement electronics with the camera by means of a hardware shutter.

\(^1\)http://www.nvidia.com/object/product-geforce-gtx-580-us.html
\(^2\)http://www.ptgrey.com/products/fireflymv/fireflymv_usb_firewire_cmos_camera.asp
2. Requirements of Computer Vision on MAVs
Related Work

The successful vision based localization of a micro air vehicle requires a proper platform and architecture as base and a stable, fast and robust computer vision approach for the actual localization.

3.1. Micro Air Vehicles

Micro air vehicles include a wide range of airframes, ranging from fixed-wing aircraft to quadrotors. As the platform used in this thesis is a quadrotor, similar airframes are most relevant. Of course fixed-wing aircraft could be used as well for localization and mapping. The STARMAC quadrotor developed by Hofiann et al. at Stanford [HRW] has a PC104 form factor onboard computer, but does not utilize it for vision processing due to limited performance. Bouabdallah has an AMD Geode processor module onboard the OS-4 quadrotor [BS], but obstacle avoidance was based on ultrasonic sensors since the processing power of the onboard computer was not yet sufficient for onboard computer vision. In the work of Roberts et al. [RSZF07] a quadrotor capable of indoor autonomous hovering was developed using ultrasonic rangefinders and no onboard computer. All these works show a custom quadrotor system design optimized for research use, but do either not have or not really utilize the onboard processing power for computer vision. On the contrary Kemp [Kem06] uses an off-the-shelf quadrotor platform. He shows autonomous flight using wireless video transmission and off-board processing. His approach is optimized for edges and therefore particularly strong in indoor and office spaces, but does not generalize to a mixed indoor/outdoor use due to
3. Related Work

high distances to outdoor buildings and the resulting position uncertainty.

3.2. Computer Vision on Micro Air Vehicles

The lastest MAV designs use onboard processing of sensor data for navigation. While up until recently onboard control was limited to GPS, recent advances in miniaturization of high-performance computing allow onboard computer vision. Previously, only larger Unmanned Aerial Vehicles (UAV) processed images onboard or used off-board processing. Roy et al. utilized an off the shelf system to capture analog video and processed this off-board for target tracking [RHBA10]. Conte et al. [CD08] performed visual odometry on a RMAX rotary wing UAV, but did not use the output for flight control, although the large platform would be capable of lifting substantial onboard processing power. However this helicopter type has 3.6 m rotor diameter and a takeoff weight of 20 kg. This is about a factor of 20 larger and heavier than the typical micro air vehicle with 500-1000 g weight and 300-700 mm diameter. The challenge in the current research is to deliver efficient algorithms for onboard localization and mapping that can be used on small-scale MAVs.

MAVs were previously constrained to low-level vision processing, such as onboard optical flow. The work of Fowers et al. [FLT+07] utilized optical flow mouse sensors to stabilize the craft and to do simple obstacle avoidance. The benefit of this approach is the direct and high speed (200-1000Hz) speed feedback, which allows to stabilize the position very efficiently. The huge drawback is however the lack of the ability to keep a global position or to detect loop closure. Achtelik et al. [ABH+09] used an Asctec Pelican Quadcopter with an onboard Hokuyo laser scanner and Intel ATOM single board computer to capture laser scan data and images for off-board processing. Bloesch et al. [BWSS] showed tethered visual localization using the work of Klein and Murray [KM07] on a commodity PC off-board by dragging an USB camera cable behind the MAV. Achtelik et al. recently moved the localization of Klein and Murray to an Intel ATOM onboard computer module and demonstrated vision-IMU based hovering in GPS denied environments [AAWS11].

3.3. Inertially Aided Localization and Odometry

Fusion of inertial data and vision has been used extensively in literature, however not for fusing the roll, pitch attitude and two points on the ground plane. In the work of Fraundorfer et al. [FTP10] a new approach for deriving the global pose from three arbitrary points was derived. Ready et al. [RT09] performed inertially aided visual odometry in a Kalman framework by registering whole images. This approach has the drawback over a natural feature
based approach that it cannot well handle partial occlusions and partial 3D structures. Both deficiencies of the ground plane assumption are filtered out as outliers in any natural feature based approach. An off-board feature based approach is presented by Angeli et al. [AFM06]. This approach however assumes a fully horizontal position of the airframe and does not account for roll and pitch angles, which requires a very steady flight position. As the authors however calculate SIFT descriptors based on the keypoint metric of Shi [Shi02] to optimize for speed, they lose some generality as the feature extraction is not any more rotation or scale invariant. Kukelova [KBP11] provided a first closed-form solution to localize from two points and known vertical direction, the presented solution in this thesis is however computationally more efficient.
3. Related Work
Overview of the Approach

After the thorough analysis of the system requirements and the state of the art, this chapter presents a quick overview the approach derived to fulfill the requirements while evading the shortcomings of previous works. The major requirements not met with the state of art were so far:

- Onboard processing for full autonomy
- Localization at rates over 10 Hz
- Time-delay smaller than 100 ms
- Option to run localization and mapping steps independently
- Ability to fragment and distribute the map for future swarm application

4.1. MAV Specific Computer Vision

This work is focused on application areas where the use of computer vision for localization is either beneficial because of completely missing GPS reception, such as indoors, or because reception is weak. Not taking a general malfunction or a GPS shutdown into account, the loss of signal is most probable near high vertical structures, such as in cities or near power lines.

Therefore in all situations where computer vision provides a higher accuracy than GPS, the following assumptions hold:

- Underground is often planar (indoor, streets, grass)
4. Overview of the Approach

- GPS-location is unstable due to low signal and reflections
- Most application areas will offer enough light
- Flight altitude is relatively low in the 2 - 10 m range
- The MAV is able to estimate its attitude for roll and pitch
- The MAV is able to estimate the yaw speed

This allows to safely assume that the majority of ground texture is on a horizontal plane and that the vertical direction perpendicular to this plane is known. Fig. 4.1 shows the relation between body frame, gravity vector and ground plane. It also allows to assume to know the rotational speed around the yaw axis. These very basic assumptions allow to significantly increase robustness and processing speed, as will be shown in the reminder of this paper.

![Figure 4.1: Camera frame to Inertial Frame Relation](image)

4.2. Vehicle Architecture

The vehicle used throughout this work was the PIXHAWK\(^1\) Cheetah quadrotor. The system includes in just 1.3 kg liftoff weight:

- 4x Point Grey FireFly MV 640x480 monochrome cameras
- 1x Intel Core 2 DUO 1.86 GHz single board computer (20-27W)
- 1x pxIMU Inertial Measurement Unit / Autopilot
- 4x Motor with controller
- 1x 3700 mAh lithium polymere battery

\(^1\)http://pixhawk.ethz.ch
4.2. Vehicle Architecture

It is capable of staying 15-18 minutes in the air at full CPU load. Due to the relative small size of 0.55 m diameter (maximum diameter from rotor-tip to rotor-tip) the system is suited for indoor and outdoor use. Fig. 4.2 depicts a CAD drawing of the system. The four cameras are mounted in two pairs, one pair oriented downwards, one front. The inertial measurement unit is placed in the center of the structure. The onboard Intel Core 2 DUO computer is the rectangular structure on top.

![Figure 4.2: PIXHAWK Cheetah Quadrotor, 10 inch version](image)

As shown in Figure 4.4, several processes participate in the localization and position control steps. Camera images and attitude measurements are captured at the very same time, which is enforced by the IMU triggering the electronic shutter of the camera chip. As the camera hardware is interfaced by USB 2.0 and the IMU by serial port, the image hub Linux process collects and unifies the measurements and images and delivers the synchronized data to the connected computer vision processes. This architecture and implementation has been contributed by the author in an earlier work [MTFP11]. After the localization process has estimated the current position, the $X, Y, Z$ and yaw ($\psi$) values are transmitted back to the IMU/autopilot unit. This unit has a Kalman filter using a constant speed model to estimate the position and the position change rate of these four dimensions. To simplify the calculation on the 60 MHz ARM7 processor of the pxIMU unit (Fig. 4.3) [Dob10], the position estimation is implemented as four independent 1-dimensional discrete Kalman filters.

The four independent Kalman filters are modelling the system with a constant speed model. The states are then:

$$
x_k = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad y_k = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}, \quad z_k = \begin{bmatrix} z \\ \dot{z} \end{bmatrix}, \quad \psi_k = \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix}
$$
4. Overview of the Approach

For the first dimension $x$, the filter estimates the current state $x_k$, which is modeled by

$$x_k = A \cdot x_{k-1} + w_{k-1}.$$  

The dynamics matrix $A$ models the law of motion, $x_{k-1}$ represents the previous state and $w_{k-1}$ the process noise. The input is the position measured at certain time steps, where the measurements are expressed as the gain $H$ times the current state plus the measurement noise $v$.

$$z_k = H \cdot x_k + v_k$$

This constant-speed model allows to estimate the vehicle’s speed from position measurements only. The formulation using the law of motion for $A$ also allows to employ varying time intervals for the Kalman updates. This is important since the processing times of a computer vision pipeline typically vary with image content.

4.3. Vision-IMU Synchronization and Fusion

MAVs need to employ inertial measurement units (IMUs) to estimate the attitude of the vehicle to control it’s flight. The IMU is therefore already available on the MAV. The available commercial systems however provide no means to synchronize the inertial estimates to image data. This is does not allow tight vision-IMU fusion. Therefore a new system was designed by author [MTFP11] to synchronize vision and IMU information by using a hardware shutter device.

Interestingly recently cell phones also got the same or at least very similar sensors as commonly used in MAV IMUs. These sensors are used for digital
image stabilization or motion blur canceling. These techniques have the same requirement of synchronizing the sensor data to image data, which implies that the results presented here can be transferred to mobile devices as the hardware support is readily available.

4.4. Localization

The presented localization and mapping pipeline is, to the authors knowledge, the fastest pipeline utilizing a global localization scheme for each frame for micro air vehicle. With onboard runtime speeds of on average of 15 Hz and with a minimum of 10 Hz realtime flight control was demonstrated using this localization pipeline. Through the global localization the pipeline is, in contrast to the state of the art which mostly employs tracking for realtime performance, capable of recovering the pose at the same speed even if the intermediate frame did not contain any features, e.g. because the field of view of the camera was blocked. This makes this technique particularly robust.

4.4.1. Fronto-parallel View

The pipeline rotates the feature positions into the fronto-parallel view, which is possibly through the previously introduced known vertical direction and known horizontal plane assumptions. This leads to the alignment of camera and inertial frame yaw ($\psi$) axes.

4.4.2. Feature Extraction and Matching

In order to localize the micro air vehicle, the first required step is to establish point correspondences between the current camera frame and the global map. These correspondences then allow to compute the global pose of the system. This already implies three main requirements for the feature extraction: It should give correct correspondences, it should localize the points precisely and it should be computationally efficient. This work uses the CenSurE keypoint localization and the BRIEF feature extractor. This combination would however not provide any usable results, since it is not orientation invariant.
4. Overview of the Approach

Therefore CenSurE was extended with an orientation assignment step using a gradient histogram approach. The new rotation invariance contributed in this work can directly couple inertial information with the feature extraction step. This reduces the overall feature extraction time by more than 50% while still maintaining rotation invariance.

4.4.3. Localization with 2pt Algorithm

Utilizing the known vertical direction and the ground plane assumption, the absolute pose of the vehicle can be recovered from just two point correspondences. The vehicle body has six degrees of freedom:

\[
v = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}
\]

Out of which \( \phi \) and \( \theta \) are known. This leaves four unknowns which can be solved for with the two equations each point correspondence contributes.

4.5. Local Mapping and Swarm Mapping

The plane assumption made in the localization step can be again exploited for very efficient mapping, increasing the overall certainty about the feature position. Without the assumption, two observations of the same feature would be necessary to initialize its Z-position, with the position uncertainty of the two cameras adding up. Since the plane normal is known and the position has been optimized based on the current position using a least-squares estimate, the initial positioning of the feature on the map is already locally optimal. It could be further optimized using local or global optimization techniques such as bundle adjustment, particle filtering or Kalman filtering, but at high runtime costs.

Since features are initialized from a single observation only and are completely independent, the map can be shared on a per-feature basis with other MAVs. Chapter 7 will show an initial demonstration of multiple vehicles successfully mapping cooperatively.
Feature Extraction

In order to localize the micro air vehicle, the first required step is to establish point correspondences between the current camera frame and the global map. These correspondences then allow to compute the global pose of the system. There are three main requirements for the feature extraction: It should give correct correspondences, it should localize the points precisely and it should be computationally efficient. As will be shown in this chapter, correct matches and correct localization contradict the efficiency criterion. It is therefore rather a question of the right tradeoffs.

5.1. Overview

Most state of the art feature extractors suitable for this application have a three-stage pipeline: The first stage is to identify salient locations in the image which have a high probability of being detected again in another image of the same scene under different lighting conditions or a different viewpoint. A closely coupled step is to assign the correct scale to this keypoint. Since real-world scenes seldomly consist of only Dirac pulses, the detector has to find the correct location and scale. Failure to do so might lead to the selection of the wrong location and thus makes it difficult to re-detect the same location from a different viewpoint. This is therefore a typical failure scenario for detectors not estimating the scale, such as FAST [RD06]. The third, often decoupled step is to assign an orientation to the keypoint. Since a rotation around the camera Z-axis leads to huge changes in the image, a detector invariant to 3D movements of the camera has to estimate what the "unique" orientation of
5. Feature Extraction

each keypoint is.

5.2. SIFT

The scale invariant feature transform of Lowe [Low04] has become the de-facto standard in feature extraction. It combines the steps of keypoint extraction and orientation assignment with a robust descriptor. Lindeberg [Lin93] proved in his work that, taking a number of justified assumptions into account, the only scale space function is the Gaussian function. SIFT approximates the Laplacian of Gaussians \( \delta^2 \nabla^2 G \) at different scales with the difference of gaussians (DoG), which is the so-called image pyramid. This operator provides keypoint locations and scale, but requires to smooth the image with several times with a Gaussian kernel, a computationally very expensive operation. The detected extrema are optimized with a further localization step. Key-point candidates are evaluated using the Harris formulation of the Hessian of the image to reject edge responses. Before the SIFT descriptor can be extracted, the dominant orientation of the feature is calculated using a gradient histogram. The SIFT descriptor is then extracted in the dominant direction of the feature. It is composed of of weighted sums of the gradient histogram. As the successful application of the principal component analysis [Pea01] to SIFT shows, the descriptor is however redundant with its 128x32bit IEEE754 single precision floating point numbers. Ke et al. [KS04] even show that the descriptor is more distinctive than the uncompressed version. This insight is the basis for using a different approach to extract the feature vector in the scope of this work.

5.3. SURF

An approximation to SIFT is the SURF detector/extractor [BETG08]. While the extraction step is based on an approximation to the Difference-of-Gaussians (DoG), the feature vector itself is an extension of the SIFT feature vector, which needs only 50% of the space (128 vs 64 bytes), but outperforms SIFT according to the authors. The difference of gaussians can be approximated with the difference of boxes by approximating the second order partial gaussian derivatives with box filters. This operation alone would save the filter kernel some floating point operations. The main speed improvement over SIFT is however obtained by exploiting another important property of the box filter: It can be efficiently calculated from a box-based integral image. The integral image is built by summing up the contents of the image on the upper-left of each pixel. By subtracting the sums of the different corner points, one can easily get the intensity sum for very large image areas in constant time.

Because of this property SURF does not subsample and scale the image while
traversing the scale space, but scales the filter kernel.

5.4. BRIEF and CENSURE

While SIFT and SURF both provided a keypoint detector and feature extractor, the BRIEF and CENSURE combination presented here uses a pure keypoint detector with a pure feature extractor to obtain the localized feature vector.

5.4.1. CENSURE

The CENter SURround Extrema detector [AKB08] is calculating a scale-space similar to SIFT and SURF, with the major difference that the scale space operator approximation technique is Laplace / Center-surround instead of Laplace / differences of gaussians (SIFT) or Laplace / differences of boxes (SURF). CenSurE achieves full spatial resolution at every scale as it does not build an image pyramid, but computes the spatial responses at every scale. Approaches such as SIFT and SURF subsample the image or filter and therefore the scale resolution decreases with increasing feature size, as the sample intervals increase with respect to the original image. The selected center-surround approximation of the Laplacian has also be shown to to provide a better scale estimate by Mikolajczyk et al. [MTS⁺05]. Mikolajczyk et al. also identified [MS02] two functions to select keypoints at different scales to perform best, the Harris-Laplacian and the Hessian-Laplacian operator. Both provide an estimate of the scale and the cornerness of a particular point in the image and allow to select the extrema using non-maximum suppression and the final values using a simple threshold on the cornerness. CenSurE approximates the Harris-Laplacian function by first approximating the Laplacian with a bi-level wavelet filter and then calculating the Harris response at candidate locations of the Laplacian. This order is inverting the more obvious order of first finding a locally maximal Harris response and then evaluating the Laplacian at this location to obtain the correct scale. The authors however claim that the Laplacian calculation can be more easily approximated, allowing to calculate the Harris response only at selected locations. The calculation is speeded up by using the integral image technique. They use the scale-adapted Harris response in contrast to the non-adapted response SIFT uses:

$$H = \begin{bmatrix} \sum L_x^2 & \sum L_x L_y \\ \sum L_x L_y & \sum L_y^2 \end{bmatrix}$$

Since the bi-level filter has the form of an octagon to provide better rotation
5. Feature Extraction

...invariance than a difference of boxes, the integral image is slanted, allowing to calculate 45 degree integrals as well. As the algorithm outperforms the keypoint detection step of SURF roughly by a factor of three, this claim seems to be valid. The authors also provide experimental results on visual odometry indicating that the octagon based approximation of the Laplacian performs best regarding the deviation from ground truth.

Inertial Orientation Assignment with Histogram Fallback

One of the key contributions of this thesis is the inertially aided orientation assignment. Since a highly accurate angular speed estimate is available through the onboard gyroscopes, calculating the gradient histograms for all keypoints is not necessary for every frame. Instead the orientation can be tracked using the gyroscopes between frames and updated based on the position estimate from the localization pipeline. This leads to a drift-free orientation assignment without the need for gradient histogram extraction. Of course this pipeline need initialization and re-initialization, which is why the CenSurE/BRIEF combination presented first by Calonder et al. [CLSF10] was extended with rotation invariant BRIEF extraction in the scope of this work. Although the BRIEF detector is roughly rotation invariant to about $\pm$5 degrees, this does not allow for any significant viewpoint change. An approach similar to [BETG08] was used: First a circular neighborhood of size $2s$ is sampled with haat wavelet filters of side length $1s$ to calculate the gradient directions at each location of the neighborhood. The feature direction is then calculated as the maximum of a sliding window of size $\frac{\pi}{3}$. This orientation is then used to apply the following feature extraction step to the rotated patch, which makes CenSurE/BRIEF fully rotation invariant. Interestingly the presented case for the yaw-orientation is the most challenging case for inertial orientation assignment. For handheld applications, such as cell phone or digital cameras, the important features will mostly be on the vertical planes. The orientation of these planes can be easily directly estimated by the gravity vector. This implies that no initial histogram based assignment is necessary and can help to dramatically improve runtime speeds on mobile phones.

Figure 5.1.: Wavelet filters to calculate the gradient histogram
5.4. BRIEF and CENSURE

5.4.2. BRIEF Descriptor

BRIEF is based on the insight that the existing features descriptors (e.g. SURF and SIFT), could be compressed by dimensionality reduction techniques and quantization techniques from 128 (SIFT) or 64 (SURF) floating point values to just a few bits per dimension without loosing recognition performance, as shown by Tuytelaars et al. [TS07] and Winder et al. [WHB09]. This shows that the SIFT and SURF descriptors contain redundant information. The BRIEF descriptor uses a series of individual binary intensity comparisons along a line on the feature patch to establish the feature vector. Fig. 5.2 show the comparison pairs for BRIEF-32 with 256 binary comparisons.

![Figure 5.2: BRIEF gaussian distributed point comparison pattern](image)

These binary comparisons are executed on image patch locations. Fig. 5.3 shows an image patch and the resulting binary vector, visualized in this case line-wrapped.

These lines are distributed randomly with a gaussian centered over the key-point [CLSF10]. As Calonder et al. explain in their paper, random sampling per definition provides optimal compression of the sampled information. BRIEF shows comparable recognition results to SIFT and SURF, but only needs 16/32 bytes instead of 64/128 bytes. Results shown in the remainder is this work used the BRIEF-16 descriptor, which was sufficient to successfully match about 200 features per camera frame from 7000-14000 features in the database.

5.4.3. Performance Evaluation

Table 5.1 below summarizes typical runtimes for the image frame depicted in Fig. 5.4 of the three extractors. BRIEF + CenSurE has been tested once in the original form and once in the new variant with optional orientation assignment developed in this work. The BRIEF implementation was implemented
5. Feature Extraction

Figure 5.3: BRIEF feature extracted from image keypoint

from scratch with careful optimization for speed in the orientation assignment and feature extraction steps. Speed differences between the extractors typically vary ± 10% depending on the image content for comparable number of keypoints detected/extracted. BRIEF is abbreviated as B, CenSurE is abbreviated as C and the newly contributed orientation assignment is abbreviated with O. All readings were taken on an Intel Core 2 DUO low-voltage 1.86 GHz with 2 GB RAM using a single core. Lines marked with i7 were taken from a System with Intel i7 2.6 GHz with 8 GB RAM using a single core. The analysis was only conducted to estimate the relative differences between SIFT, SURF and CenSurE/BRIEF. As the results show, SURF is about twice as fast as SIFT. BRIEF outperforms SIFT almost by an order of magnitude and is still almost five times as fast SURF. The respective SIFT/SURF implementations were taken from OpenCV\(^1\). For an extensive comparison of runtimes and recall/precision, please refer to [CLSF10].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>535</td>
<td>422.94 ms</td>
<td>216.39 ms</td>
<td>639.33 ms</td>
<td>1.195 ms</td>
</tr>
<tr>
<td>SURF</td>
<td>480</td>
<td>118.71 ms</td>
<td>163.77 ms</td>
<td>282.48 ms</td>
<td>0.589 ms</td>
</tr>
<tr>
<td>B+C</td>
<td>431</td>
<td>35.21 ms</td>
<td>21.56 ms</td>
<td>56.77 ms</td>
<td>0.132 ms</td>
</tr>
<tr>
<td>B+O+C</td>
<td>431</td>
<td>71.53 ms</td>
<td>22.21 ms</td>
<td>93.42 ms</td>
<td>0.217 ms</td>
</tr>
<tr>
<td>B+C(i7)</td>
<td>431</td>
<td>25.18 ms</td>
<td>4.81 ms</td>
<td>29.99 ms</td>
<td>0.07 ms</td>
</tr>
<tr>
<td>B+O+C(i7)</td>
<td>431</td>
<td>52.22 ms</td>
<td>5.24 ms</td>
<td>57.46 ms</td>
<td>0.133 ms</td>
</tr>
</tbody>
</table>

Table 5.1: Runtime speeds of different feature extractors

\(^1\)http://opencv.willowgarage.com
Figure 5.4: Censure/BRIEF Extraction example with newly contributed orientation assignment
5. Feature Extraction
Natural Feature based Localization

To localize the aircraft, 3D-2D correspondences and the roll and pitch angles are used to solve for position. Since a free moving body has six degrees of freedom, of which three are translational and three are rotations, the known roll and pitch angles eliminate two. The remaining four degrees of freedom can be covered with two 3D-2D correspondences, hence the approach is the 2-point algorithm. In the remainder in this work it is assumed that all 3D points lie on a plane orthogonal to the gravity vector, the ground plane. This assumption is not required for the localization step, but later on in the mapping step, as it allows to assign a Z-value from a single observation of the feature.

6.1. Establishing 3D-2D correspondences

6.1.1. Keypoint localization

Keypoints are localized using the CenSurE keypoint extractor. Since it does not only localize extrema of the Harris operator, but also provides scale by means of a center-surround Laplacian approximation, the output of this step are 2D locations in the image frame and a size value, which is a radius in pixels around the 2D location.
6.1.2. Orientation assignment

CenSurE does not provide orientation of the keypoint, which would make the feature matching impossible if the MAV does not have the same orientation as in the map. Therefore CenSurE was extended with an intelligent orientation extraction scheme. CenSurE leaves out the orientation assignment for a good reason: To estimate the orientation, the commonly used technique is a gradient histogram around the keypoint. As filtering the surrounding with wavelets is necessary, this is a very expensive step. In the experiments conducted during this work orientation assignment roughly doubled the CenSurE extraction time.

Following the widely known design pattern of trading memory for speed, the feature database (map) contains the feature vector for each keypoint in two versions: One with orientation assignment and rotated to the dominant orientation, one upright. Since each BRIEF feature consumes only 16/32 bytes, the memory consumption is negligible, one million brief features consume only 16/32 MB of main memory and the more usual number of of 10-15,000 for a medium sized map only 0.5 MB.

The rationale for this double storage is simple: Since all features lie on a common plane, all have the same orientation, in the case of a downward looking camera it is the yaw angle $\psi$. As the MAV is equipped with a gyroscope, the relative motion between the previous and the current frame can be very precisely estimated. Thus the orientation extraction for all features has to be done only once and from then on the orientation can be pre-estimated by the vision-based absolute yaw angle of the previous frame and the relative, gyro-estimated offset to the current frame. Please note that this yaw estimation pipeline is drift-free.

Similar to the design pattern employed by Wagner et al. \cite{WRM08}, where memory was traded for less processing time at runtime on scale estimation, this allows to only assign the orientation to the feature if needed. The yaw angle tracking worked during the testing however so well, that this is seldomly necessary. Therefore the orientation tracking cuts down the feature extraction time roughly by 50 \%.

To account for yaw, the randomized BRIEF locations are rotated by the system yaw: $b' = R_\psi b$.

6.1.3. BRIEF Extraction

The BRIEF feature vector is extracted as described in chapter 5. To minimize the extraction time, the locations for the pixel tests are rotated and then the extraction is performed on the upright image, instead of rotating each individual pixel patch, as commonly done in SURF and SIFT implementations.
6.1.4. Matching

The BRIEF feature vector is a binary string of 128 or 256 bits length (16 or 32 bytes). Compared to the 64 dimensional SURF descriptor in floating point precision or the 128 dimensional SIFT vector this seems less descriptive at first glance. The intuition is however wrong: Through the ‘curse of dimensionality’ [BC57] the volume decreases exponentially in a high-dimensional space. This means that the nearest neighbor of an arbitrary point if spaced very far away. Therefore only a binary distance metric is sufficient to successfully distinguish features. This intuitive argument was already backed in chapter 5 by the fact that PCA-SIFT, a dimension-reduced variant of the SIFT descriptor, performs equally well or better than the original descriptor. It is already obvious that BRIEF uses 8-16x less space than SURF and 16-32x less space than SIFT. More importantly BRIEF features can be matched using the Hamming distance [Ham50] (number of toggled bits after the XOR operation of two bit vectors), which is commonly used in data compression algorithms. This led to the recent addition of the POPCNT operation is Intel and AMD processors, which allows to calculate the hamming distance of a 64 bit vector with one XOR and one POPCNT operation. Due to the pipelining of current CPUs this leads to feature matching at around a quarter of the CPU speed per core.

6.1.5. Roll-pitch compensation

To compensate for roll and pitch, each feature is projected into the unit-space and then rotated to the fronto pallel view:

\[ u' = R_{\phi \theta}K^{-1}u \]

Which is, expressed in homogeneous coordinates:

\[
\begin{bmatrix}
    u' \\
    v' \\
    z'
\end{bmatrix} =
\begin{bmatrix}
    \cos(\theta) & \sin(\phi)\sin(\theta) & \cos(\phi)\sin(\theta) \\
    0 & \cos(\phi) & -\sin(\phi) \\
    \sin(\theta) & \cos(\phi)\sin(\phi) & \cos(\phi)\cos(\theta)
\end{bmatrix}
\begin{bmatrix}
    f_x & 0 & p_x \\
    0 & f_y & p_y \\
    0 & 0 & 1
\end{bmatrix}^{-1}
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}
\]

6.1.6. Efficient linear-time Inlier Selection

The widely used [FB81] RANSAC approach for inlier selection based on geometric verification was replaced in this work with a new constant-time, histogram based approach. The benefits are a linear-time estimation of the inlier set (walking through all features of the current frame only once) and the selection of inliers per dimension.
6. **Natural Feature based Localization**

**Yaw Estimation**

Since the orientation assignment of each feature point is equally expensive as the keypoint detection itself, the estimation of the camera yaw angle should not rely on it. Therefore the vector between point pairs is used. The direction of the vector on the map and the direction of the vector in the frame provide the yaw offset of the camera. All estimated directions are inserted into a 360 degree histogram and then dominant direction is calculated as average of the values in the peak bins of the histogram. The estimated yaw angle is used in the further estimation of X and Y.

Project the homogeneous frame coordinates onto the plane parallel to the ground plane and calculate the vector between the frame point pair

\[
\vec{p} = \begin{bmatrix} p_{2x} \\ p_{2y} \\ p_{2z} \end{bmatrix} - \begin{bmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \end{bmatrix}
\]

Scale the frame vector to unit length:

\[
\vec{p} = \frac{\vec{p}}{\sqrt{p_x^2 + p_y^2 + p_z^2}}
\]

Calculate the vector between the map point pair

\[
\vec{m} = \begin{bmatrix} m_{2x} \\ m_{2y} \\ m_{2z} \end{bmatrix} - \begin{bmatrix} m_{1x} \\ m_{1y} \\ m_{1z} \end{bmatrix}
\]

Scale the map vector to unit length:

\[
\vec{m} = \frac{\vec{m}}{\sqrt{m_x^2 + m_y^2 + m_z^2}}
\]

The final yaw estimate \( \psi \) is obtained by calculating the angle between the two vectors:

\[
\psi = \arctan2(p_y, p_x) - \arctan2(m_y, m_x)
\]

This estimate is inserted into the histogram of all yaw estimates, from which the peak indicates the true camera yaw angle \( \psi \).

**Scale / Z Estimation**

The Z-value / scale can be estimated from two properties: Using the focal length and the feature size/scale estimated by CenSurE or the distance between point pairs. The latter technique proved to be more robust, because the distance between points is larger relative to the camera focal length and thus provides a better depth resolution.

The first step is to project the homogeneous frame coordinates onto the plane parallel to the ground plane and calculate the vector length between the frame point pair and map point pair.
6.2. Absolute Pose from two 3D-2D Correspondences

\[ \vec{p} = \begin{bmatrix} p^2_x \\ p^2_y \\ p^2_z \end{bmatrix} - \begin{bmatrix} p^1_x \\ p^1_y \\ p^1_z \end{bmatrix} \]

Calculate the frame vector length:

\[ l_p = \sqrt{p^2_x + p^2_y + p^2_z} \]

Calculate the vector between the map point pair

\[ \vec{m} = \begin{bmatrix} m^2_x \\ m^2_y \\ m^2_z \end{bmatrix} - \begin{bmatrix} m^1_x \\ m^1_y \\ m^1_z \end{bmatrix} \]

Calculate the map vector length:

\[ l_m = \sqrt{m^2_x + m^2_y + m^2_z} \]

The final z estimate is obtained by the factor of the two vector length:

\[ z = \frac{l_m}{l_p} \]

The estimates from all point pairs are inserted into the Z-histogram, as shown in Fig 6.1

![Figure 6.1: Z estimations histogram](image)

**X and Y Estimation**

The X position of the camera can be estimated from the fronto-parallel view, which is compensated for yaw and scaled correctly, from a single point. This provides a densely filled histogram and good resolution. The same applies to the Y-position estimation, which is decoupled from X.

\[
\begin{bmatrix} t_x \\ t_y \end{bmatrix} = z \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix}
\]

6.2. Absolute Pose from two 3D-2D Correspondences

After the successful selection of the inlier set, the final pose is estimated using all inliers to obtain a least squares solution. Two points are however sufficient in the minimal case with this algorithm.
Solve for camera pose $t = [t_x t_y t_z]^T$ with 3D-2D correspondences using pixel coordinates $u = [u v 1]^T$ and 3D points $X_n = [X Y Z 1]^T$:

$$u \times PX = 0,$$

the cross-product of $u$ and $PX$, which is expressed as the multiplication of the skew-symmetric matrix for $[u v 1]^T$ with $PX$.

$$\begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 & x_c \\ \sin(\psi) & \cos(\psi) & 0 & y_c \\ 0 & 0 & 1 & z_c \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0$$

Two solution approaches are presented here, one expresses the angle $\psi$ using sine and cosine, the other using a tangens substitution for sine and cosine.

### 6.2.1. Least-squares solution using sine-cosine constraint

The above problem can also be solved directly without substitution, using the $\sin^2(\psi) + \cos^2(\psi) = 1$ constraint to recover the metric scale instead of substituting sine and cosine initially. This formulation eliminates the tangens expression and results in the system $Ax = 0$, which will give two exact solutions for two points and two least-squares solutions for more points. Additional equations can be stacked to $A$ like in the formulation above.

$$\begin{bmatrix} 0 & -1 & v & -X & -Y & vZ \\ 1 & 0 & -u & -Y & X & -uZ \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \sin(\psi) \\ \cos(\psi) \\ 1 \end{bmatrix} = 0$$

The system can be solved using SVD. As it is a homogenous system but the solution should be in metric scale, the scaling factor $\lambda$ has to be calculated as final step. This can be easily done by using the sine-cosine constraint $\sin^2(\psi) + \cos^2(\psi) = 1$. The solution vector $V$ contains $x, y, z$ and sine / cosine. The sine-cosine constraint results in the scaling factor $s$ being:

$$s = \sqrt{\frac{1}{\sin^2(\psi) + \cos^2(\psi)}}.$$
6.2. Absolute Pose from two 3D-2D Correspondences

6.2.2. Least-squares solution using tangens substitution

This solution is following the formulation of [KBP11]. It is computationally however more expensive and therefore only included for completeness. By substituting $q = \tan \psi_z$ sine and cosine can be substituted by expressions $\cos \psi_z = \frac{1 - q^2}{1 + q^2}$ and $\sin \psi_z = \frac{2q}{1 + q^2}$. The resulting rotation matrix becomes:

$$(1 + q^2)R_y(q) = \begin{bmatrix}
1 - q^2 & -2q & 0 \\
2q & 1 - q^2 & 0 \\
0 & 0 & q^2 + 1
\end{bmatrix}$$

Expanding $u \times PX = 0$, one obtains three equations per point, of which the first two are linearly independent:

$$\begin{bmatrix}
t_zv - 2Xq - t_y + Y(q^2 - 1) + Zv(q^2 + 1) \\
t_x - 2Yq - t_zu - X(q^2 - 1) - Zu(q^2 + 1) \\
X(2qu + v(q^2 - 1)) + Y(2qv - u(q^2 - 1)) + t_yu - t_xv
\end{bmatrix} = 0$$

These two equations per point are stacked to the matrix $A$, allowing to calculate the pose from two points and resulting in a least-squares solution for more than two points. To obtain the pose, the homogeneous system $Ax = 0$ is solved:

$$\begin{bmatrix}
0 & -1 & v & Y + Zv & -2X & -Y + Zv \\
1 & 0 & -u & -X - Zu & -2Y & X - Zu
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
q^2 \\
q \\
1
\end{bmatrix} = 0$$

The over-determined system can be solved using SVD to obtain the least squares solution. To solve for the camera position $t$, the quadratic equation $q^2 + q + b = 0$ has to be solved, which is $q^2 + a(4, 5) + a(4, 6) = 0$, gives two solutions. Since all the solutions for $t_x$, $t_y$ and $t_z$ are expressed in $q$, the correct solution is the one for which the camera is above the ground plane. Geometrically the two solutions are mirrored on the ground plane. To obtain the final values, the scaling of $(1 + q^2)$ has to be accounted for, leading to the final terms: $t_x = \frac{-a(1,5)q}{(1+q^2)}$, $t_y = \frac{-a(2,5)q}{(1+q^2)}$ and for z position: $t_z = \frac{-a(3,5)q-a(3,6)}{(1+q^2)}$. The calculation of the yaw angle $\psi$ is straightforward: $\psi = 2 \arctan(q)$. 

33
6. Natural Feature based Localization

6.2.3. C/C++ Implementation

The sine-cosine formulation has been selected as C/C++ implementation basis and the complete pipeline including histogram inlier selection and the final least squares optimization step was implemented onboard. The histogram assignment was implemented using standard C++ functions. The SVD implementation stems from GSL and for maximum accuracy IEEE-754 double precision floating point numbers were used. The actual decomposition was performed the function gsl_linalg_SV_decomp_jacobi from the GNU Scientific Library (GSL). Listing 6.1 shows how the hamming distance calculation gets severely optimized by the hardware-accelerated variant over the already optimized implementation of the Berkeley Software Distribution (BSD). The example shows 32 bit junks since this was the installed system on the helicopter, but there is also a 64 bit version of the instruction. The code is just intended to illustrate the fact that the hardware instruction saves significant processing time.

Listing 6.1: POPCNT Intel GCC

```c
int brief_frame;
int brief_map;
int i = brief_frame ^ brief_map;
#if (defined __SSE4_2__) && (defined _64_)
   // POPCNT method
   return _mm_popcnt_u32 (i);
#else
   // BSD optimized version
   i -= ((i >> 1) & 0x55555555);
   i = (i & 0x33333333) + ((i >> 2) & 0x33333333);
   i = (i + (i >> 4)) & 0x0F0F0F0F;
   i = (i * 0x01010101) >> 24;
   return i;
#endif
```

Test and Validation Setup

To assess the quality of the localization, tests were conducted in a Vicon\(^1\) motion capture system, providing 0.1 mm spatial resolution with 12 cameras at up to 250 Hz. As Figure 6.2 shows, deviation from the Vicon ground truth position is very little.

\(^1\)http://www.vicon.com/
6.2. Absolute Pose from two 3D-2D Correspondences

![Diagram of X-Y-Z position estimate]

**Figure 6.2.:** X-Y-Z position estimate
Figure 6.3: Yaw estimate
Mapping and Optimization

Since the measurements of the roll and pitch angles have an error, but no static offset and drift, using roll and pitch in the mapping process does not generate map drift. The general mapping pipeline is as follows: Whenever the localization quality is good (sufficient number of inlier features, small enough reprojection error) and a map update would be beneficial (map grid contains uncovered areas) the map is extended with features found in the current frame, but not matched to the map. This implies that those features are not represented yet in the map or that their earlier representation is so degenerated that matching was not possible. In both cases it is desirable to add the feature, as only salient features are added to the map. To allow continuous mapping while improving the map coverage over time, the algorithm establishes a 2D occupancy grid. The desired fillrate can be adjusted at runtime. Since the localization quality is related to the number of raw features in each grid cell, it is also a valid measure to adjust the flight path to not cross areas with insufficient localization.

7.1. Position Refinement

Since the focus during the localization step is on a robust position estimate even from a low number of correspondences, the position is refined with an additional RANSAC [FB81] step with reduced reprojection error threshold. The final camera pose is calculated based on this inlier set. The map positions of all features are then found using this camera pose by projecting the keypoint location in the camera frame onto the ground plane. Please note that,
7. Mapping and Optimization

for 3D depth cameras, the planar world assumption relaxes to a 3D world assumption, as Z-information about the feature coordinate is available.

The transformation from pixel coordinates in the camera frame \( u \) to the world frame is given by:

\[
X = \lambda RR_{\theta\phi} K^{-1} u + t
\]

Which is:

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} = z
\begin{bmatrix}
cos(\psi) & -sin(\psi) & 0 \\
sin(\psi) & cos(\psi) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
cos(\theta) & sin(\phi)sin(\theta) & cos(\phi)sin(\theta) \\
0 & cos(\phi) & -sin(\phi) \\
-sin(\theta) & cos(\theta)sin(\phi) & cos(\phi)cos(\theta)
\end{bmatrix}
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix}
\]

As the histogram based inlier selection provides an estimate about the variance, the map data of each feature can be augmented with the uncertainty of its position. This is possible as the localization in the map depends on the camera localization accuracy. To correctly scale the variance, the angle between camera center and the map surface has to be taken into account.

The final result of the mapping pipeline is shown in Fig. 7.1. The overall space is divided by the map grid. Each cell is assigned either the mapped or unmapped state. Mapped cells have more than a certain features located in their boundaries and therefore allow an overpassing MAV to visually localize itself. Cells meeting the threshold are marked in green. The current camera frame is indicated by the colored camera center and the red frame around it. Map feature point are marked with small blue circles. The underlying texture has been reprojected onto the ground plane based on the position obtained by visual odometry / local mapping. It shows the relative consistency of the data, even in the absence of an expensive global optimization step.

7.2. Hole Filling

As Fig. 7.1 shows, the map initially contains areas without sufficient feature coverage. While the occupancy grid approach prevents excessive amounts of features per grid cell, it also allows to fill in only the not well covered cells later. This strategy allows efficient, yet distributed map management over multiple vehicles without the need for more than exchanging mutually new found features.
7.2. Hole Filling

Figure 7.1.: Output of 2pt localization / mapping pipeline
7.3. Flight Planning

Crucial for safe flight is the availability of sufficient feature correspondences for a robust localization. Assuming that multiple vehicles operate in the same area and not all vehicles are actively mapping and can therefore fill gaps, the presented feature occupancy grid allows to route the non-mapping vehicles over areas where safe localization is provided. This offers the huge benefit of being able to operate heterogeneous swarms with larger vehicles with onboard mapping capabilities and smaller units just capable of basic flight control. The feasibility of onboard dynamic path planning based on a 2D occupancy grid has been demonstrated in earlier work in the PIXHAWK project in by Heng et al. [HMT+11].

7.4. Multi-Vehicle / Distributed Mapping

As the overall vision pipeline treats each map feature individually, it is easily possible to transmit parts of the map to other vehicles. With only 16/32 bytes descriptor size for each feature, this is even possible over weak radio links or in larger swarm networks. The individual keypoint handling and the small descriptor size is a key differentiator of this approach to keyframe-based approaches like PTAM or large descriptors like SURF. Even if SURF is quantized to 8-bit values, it is still a 64 byte descriptor, leaving alone the significant higher runtime of the SURF extraction. Distributed mapping is relevant for areas which cannot be easily covered by a single vehicle. This implies that vehicles either operate in different rooms in an indoor scenario or significantly spaced apart in an outdoor scenario. Fig. 8.5 gives an example of a cooperative mapping process. It shows the output of two concurrent mapping runs, exchanging feature information via the MAVLink protocol. This run also includes loop closing, as the two systems started at different locations and closed implicitly the loop once they entered the same area.

Therefore high data rate wireless links between several vehicles cannot be assumed and low-bandwidth communication has to be part of the distributed mapping model. Any scenario supporting high data rate links would immediately question the benefit of multiple vehicles operated in parallel.

It is for this property that the proposed feature extraction pipeline provides significant benefits over existing localization pipelines.
Experimental Results

The experiments for this thesis were conducted using the PIXHAWK Cheetah quadrotor. Runtime speeds were evaluated using onboard-cameras and the onboard processor. The floor texture/materials included gravel, street and grass as well as aerial imagery.

8.1. Fully Autonomous Flight

Fig. 8.1 shows fully autonomous flight using the localization pipeline presented in this work. The safety tether is an additional safety measure needed due to the absence of a proper safety net in lab. As can be seen from the L-shape, there is no force on the tether, it is thus not influencing the flight behavior. The flight trajectory estimated by the algorithm and used in the control loop is shown in Fig. 8.2. It is very obvious that the PID controllers for this craft are not well tuned and suffer from oscillation. The controller itself was however not part nor in the scope of this work. The plot clearly shows that the localization pipeline produces smooth results. Together with the correctness shown earlier in comparison with Vicon ground truth data and the fact that the system actually is capable of hovering with it, shows that the approach is very suitable for autonomous flight.
8. Experimental Results

Figure 8.1.: Fully autonomous hovering

Figure 8.2.: Localization output of autonomous hovering
8.2. Single Vehicle Mapping

The mapping results shown in Fig 8.3 use the projected image frames only for visualization. Individual keypoints are projected onto the map using the descriptor extracted from the frame. This approach has the benefit over re-extracting features from the projected texture that the image pixels are not modified. Therefore the same feature will be guaranteed to have the same appearance if seen again by the camera. The circular shape of the trajectory is consistent with the actual trajectory.

![Mapping results on outdoor gravel](image)

**Figure 8.3:** Mapping results on outdoor gravel

Fig. 8.4 shows the progress of a mapping run on aerial imagery. The yellow circle is the current camera position, the red border the current frame projected to the ground. Green marked areas contain sufficient features for safe localization, red areas have not been sufficiently mapped yet. This occupancy grid allows to plan a safe path, taking sufficient visual landmarks for localization into account.

8.3. Cooperative Mapping with multiple Vehicles

On average 0.48 kb/s with BRIEF-16 for 30 new features per frame, including keypoint position, response, size and MAVLink header overhead the total data rate is 1.02 kb/s. In contrast even the smallest existing descriptor, SURF-64 quantized to 8 bit per dimension, consumes 2.46 kb/s, making it more difficult to transmit over wireless networks with low bandwidth capabilities. This is very relevant in swarm applications where radio bandwidth is shared in a broadcast network. Fig. 8.5 shows the output of two vehicles localizing and mapping the same environment concurrently. Each vehicle can benefit
8. Experimental Results

from single features from the other vehicles. Please note that no attempt was made to enforce the global consistency of the map, as this work is focused on fast feature extraction, localization and local mapping.

8.4. Visualization and Flight Control

As the software is running onboard the helicopter, the algorithm output cannot be visualized directly on the screen. For this purpose the QGroundControl\(^1\) was adjusted to the needs of the localization pipeline. It allows to conveniently adjust algorithm parameters and to check output in the realtime graph.

8.5. Additional Feature Extraction Samples

The pipeline is capable of extracting features on very different natural floor surfaces. Some examples are more challenging, since less salient scenes will provide less robustness. A complete evaluation of which surfaces at which altitudes and focal length would be one possibility for future works. The pipeline was shown to perform well on the aerial image and the gravel, further tests with other textures would have exceeded the scope of this work.

\(^1\)http://qgroundcontrol.org
8.5. Additional Feature Extraction Samples

Figure 8.4: Mapping progress on aerial texture
8. Experimental Results

**Figure 8.5.** Cooperative Mapping, two vehicle maps shown side-by-side exchanging features

**Figure 8.6.** QGroundControl 2pt GUI with realtime plot
8.5. Additional Feature Extraction Samples

Figure 8.7.: Pipeline extracting features on gravel

Figure 8.8.: Pipeline extracting features on street
8. Experimental Results

Figure 8.9.: Pipeline extracting features on grass
Conclusion and Outlook

This master thesis covers the complete localization and mapping pipeline for a micro air vehicle. As typically the majority of processing time of a localization pipeline is spent on the feature extraction and matching, this thesis focused on the first steps to optimally reduce the processing time of previous state of the art approaches.

This work presented a new localization and mapping pipeline optimized on runtime speed and robustness on micro air vehicles. The reasonable assumptions of a semi-planar surface and the efficient use of inertial measurement information allowed to reduce the processing time of the computer vision pipeline significantly. The CenSurE detector and BRIEF descriptor have been used for the first time in a MAV based localization pipeline. The contributed orientation assignment was a critical and necessary improvement of the detector to allow autonomous flight. The new concept of histogram based inlier selection and position estimation shows similar results to the least squares approach but is computationally significantly reducing complexity. This work shows the first use of the two point algorithm in this domain and presents a more efficient expression in sine and cosine than previous approaches.

The unprecedented inclusion of IMU information in the feature extraction step allows to save 50% of the typical feature extraction time, reducing the extraction time from 60-80 ms to 30-40 ms on the onboard computer system. To the authors knowledge this is the first approach exploiting this information in a general feature extraction pipeline. This has also applications to other mobile robotics fields and mobile phone applications and effectively provides the same runtime performance as an upright descriptor.
9. Conclusion and Outlook

This thesis not only contributes the localization pipeline, but also shows valid experimental results in a realtime C++ implementation on a real system. The flight results prove the validity of the approach for autonomous flight and meet the state of the art in runtime performance and accuracy.

The mapping technique introduced in this work is very efficient at runtime and allows to fully initialize new features on their first observation. As the mapping approach does not imply global structures such as keyframes for features, multiple MAVs can easily exchange their maps on a per-feature basis. The key insight on this is that individual feature positions can be optimized, whereas a keyframe-based technique would have to optimize whole keyframes, which makes a distributed approach challenging.

It is particularly important to highlight that the overall goal was to achieve realtime speeds on an onboard computer for autonomous flight. Therefore any global optimization scheme of the map was out of scope of this work. Global optimization is not necessary for local flight control and can be conducted as a second thread onboard or even offboard. The design of the feature and map storage optimally supports the future introduction of a multithreaded global optimization scheme. Other possible extensions include the introduction of local submaps based on the current position to speed up matching. At the current map sizes this was however not necessary for realtime performance and will become only relevant if large maps with global optimization and e.g. GPS landmarks are included.

While the obtained results have the obvious impact that they enable autonomous flight using natural features, there might be a more general impact as well. The presented inertially aided orientation assignment for 2D features can be utilized on cell phones as well. As the main orientation is vertical in that case, the upright feature-angle can be directly estimated from the gravity vector.

I would like to thank Laurens Mackay and Petri Tanskanen in their important support in the operation of the flight tests and the adjustment of the onboard controllers to the new input. I would also like to thank Dr. Friedrich Fraundorfer and Prof. Marc Pollefeys for their ideas and comments.
A

Addition Technical Information

A.1. MAVLink communication protocol

The localization and mapping pipeline allows to distribute BRIEF features to multiple processes. This functionality was implemented using the MAVLink\(^1\) micro air vehicle communication protocol on top of the MAVCONN aerial middleware. MAVLink is a broadcast protocol with only eight bytes header overhead per message. It is particularly well suited for low-bandwidth radio modem communication, e.g. in a vehicle swarm. Unfortunately not enough vehicles were available to test swarm mapping, but the provided implementation (which was tested on one computer with many MAV instances running in parallel) is without changes ready for wireless multi-vehicle swarm mapping.

A.2. MAVCONN Aerial Middleware

As the MAVLink protocol only represents the message format and serialization library, the MAVCONN\(^2\) represents the actual MAV middleware solution. It is using LCM by Huang et al. [HOM10], but provides the crucial vision-IMU synchronization used throughout this work.

\(^1\)http://qgroundcontrol.org/mavlink/
\(^2\)http://pixhawk.ethz.ch/wiki/software/middleware
A. Addition Technical Information
Bibliography


[KBP11]  Z Kukelova, Martin Bujnak, and Tomas Pajdla. Closed-form so-


