Thesis: Machine Learning-based Indoor Localization for Micro Aerial Vehicles

Volker Strobel

August 13, 2016

Abstract

Widespread applications, ranging from surveillance to search and rescue operations, make Micro Air Vehicles (MAVs) versatile platforms. However, due to their small size, MAVs have limited processing power and cannot fall back on standard localization techniques. The contributions of this thesis are two-fold. On the one hand, in the scope of this thesis, an efficient vision-based onboard localization technique using machine learning was developed. The development, software and hardware implementation, and results of the localization system are presented. The light-weight approach estimates x, y-coordinates within a known and modifiable indoor environment. Its computational power is scalable to different platforms, trading off speed and accuracy. Histograms of textons—small characteristic image patches—are used as features in a k-Nearest Neighbors (kNN) algorithm. Several possible x, y-coordinates that are outputted by this regression technique are forwarded to a particle filter to neatly aggregate the estimates and solve ambiguities. On the other hand, an evaluation technique is developed that predicts the performance of the proposed algorithm in different environments. It compares actual texton histogram similarities to ideal histogram similarities based on the distance between the underlying x, y-positions. The technique assigns a loss value to a given set of images, allowing for comparisons between different environments and positions within the environment. A software tool creates synthetic images that could be taken during an actual flight.

We conducted several flight tests to evaluate the performance of the approach. A comparison of the localization technique with the ground truth showed an average error of 50 cm. In a triggered landing setting, the MAV correctly landed in specified areas, in ten out of ten trials. The method was tested on 46 high-resolution images to identify well-performing maps.

The presented approach is based on three pillars: (i) a shift of processing power to an pre-flight phase to pre-compute computationally complex steps, (ii) lightweight and adaptable algorithms to ensure real-time performance and portability to different platforms, (iii) modifiable environments that can be tailored to the proposed algorithm. These pillars build the foundation for efficient localization in various GPS-denied environments.

Chapter 1

Introduction

In the world of automation, micro aerial vehicles (MAVs) provide unprecedented perspectives for domestic and industrial applications. They can serve as mobile surveillance cameras, flexible transport platforms, or even as waiters in restaurants. However, indoor employment of these vehicles is still hindered by the lack of real-time position estimates. The focus of this thesis is, thus, the development of accurate and fast indoor localization for MAVs combining computer vision and machine learning techniques.

Since precision and reliability are crucial for safe flight, autonomous indoor navigation of an MAV is a challenging task. While unmanned aerial vehicles (UAVs) for outdoor usage can rely on the global positioning system (GPS), this system is usually not available in confined spaces and would not provide sufficiently accurate estimates in cluttered environments. If sufficient computational and physical power is available, a typical approach to estimate a UAV's position is by using active laser rangefinders [18, 5]. Although this approach is used in some simultaneous localization and mapping (SLAM) frameworks, it is usually not feasible for MAVs because they can carry only small payloads. A viable alternative are passive computer vision techniques. Relying on visual information scales down the physical payload since cameras are often significantly lighter than laser rangefinders [6, 4, 2]. Additionally, many commercially available drones are already equipped with cameras.

Figure 1.1 summarizes the proposed algorithm. In contrast to other existing approaches, it does not rely on additional information, such as data from the inertial measurement unit (IMU). The only required tool is a camera, which minimizes possible points of failure. Cameras are lightweight and robust to many external influences, such as magnetic fields.

However, this reduced physical payload is not without cost: it must be traded off against the higher computational payload for the onboard CPU. Vision-based

Figure 1.1: The figure illustrates the proposed system from a high-level perspective. The current camera image of the UAV is used to extract an image feature vector—the texton histogram. The feature vector is forwarded to a machine learning model that uses a k-Nearest Neighbors algorithm to output x, y-position estimates. These estimates are passed to a particle filter, which filters position estimates over time and outputs a final position estimate (green circle). The expected loss shows regions in the map where a lower accuracy is expected. The average expected loss can be used as "fitness value" of a given map.

position estimation is usually a time-consuming and memory-intense procedure. For example, a standard technique for 3D pose estimation extracts keypoints of the current camera image and a map image and then determines a homography between both keypoint sets. While this approach has been used for visual SLAM for MAVs [6], the pipeline for accurate feature detection, description, matching, and pose estimation is CPU-intensive [21]. One way to overcome this problem is to process the data on a powerful external processor by establishing a wireless connection between the MAV and a ground station. Such off-board localization techniques often lack the versatility to function in changing environments, though, due to factors—such as the bandwidth, delay, or noise of the wireless connection—interfering with the system's reliability.

In the approach proposed in this thesis, computational power will be shifted to an offline training phase to achieve high-speed during live operation. In contrast to visual SLAM frameworks, this project considers scenarios in which the environment is known beforehand or can be even actively modified. The environment is non-dynamic and planar, therefore, the UAV will make use of texture on the bottom or ceiling of the environment. This opens the door for improving the proposed algorithm by changing the map. On the basis of desired characteristics of a given map, an evaluation technique was developed that determines the suitability of an environment for the proposed approach. This technique allows for spotting distant regions with similar image features, which could lead to deteriorated performance. The evaluation can be performed using a given map image or recorded images during flight. In the former case, synthetic images will be generated from the map image that simulate images taken during flight.

It uses a rather simple stimulus-response behavior to estimate the position, which circumvents the requirement to store a map in the UAV's 'mind'. To assign x, y-coordinates to images in a training set, keypoints in the current image and a map image are detected. This is then followed by finding a homography between them. In the next step, the complexity of these images is reduced by *sparse encoding*: images are represented by determining their histogram of textons—small characteristic image patches [27]. New images can then also be encoded by texton histograms and matched to images with known x, y-positions using the k-Nearest Neighbors (kNN) algorithm. The computational effort of the approach can be adjusted by modifying the amount of extracted patches, resulting in a trade-off between accuracy and execution frequency. While the kNN algorithm is one of the simplest machine learning algorithms, it offers several advantages: it is non-parametric, allowing for the modeling of arbitrary distributions. Its capability to output multiple predictions with an associated confidence allows for neat integration with the proposed particle filter. Finally, computational complexity can be modified by changing the size of the datasets.

The goal is to estimate x, y-coordinates in real-time and on-board of the UAV. It is assumed that the UAV flies at an approximately constant height, such that the estimation of height is not necessary. It is intended to further reduce the size of MAVs to that of an insect, hence necessitating lightweight and scalable position estimation algorithms.

1.1 Contributions

The first contribution of this thesis is a machine learning-based indoor localization system that runs in real-time on board of an MAV, paving the way to a fully autonomous system. In contrast to existing *active* approaches, the proposed *passive* approach only uses a monocular downward-looking camera. Since computer vision-based localization approaches yield noisy estimates, a variant of a particle filter was developed that aggregates estimates over time to produce more accurate predictions. It handles the estimates of the kNN algorithm in an integrative way and resolves position ambiguities. The method is an absolute position system and does not suffer from error accumulation over time.

The second contribution is a map evaluation technique that predicts the suitability of a given environment for the proposed algorithm. To this end, a synthetic data generation tool was developed that creates random variations of an image. The tool simulates different viewing angles, motion blur, and lighting settings; the generated synthetic images are labeled with x, y-coordinates based on the 3D position of the simulated camera model.

The developed software is made publicly available. It encompasses (i) the localization algorithm as part of the Paparazzi project, which consists of the texton-based approach in combination with a particle filter (ii) software for augmenting an image with synthetic views, (iii) a script for evaluating a map based on histograms and corresponding x, y-positions.

1.2 Research Questions

The research questions addressed in this thesis are:

- Can 2D positions be estimated in real-time using a machine learning approach on a limited processor in a modifiable indoor environment? Why is this important and what follows?
- Can we predict the suitability of a given map for the proposed localization approach? Why is this important and what follows?

1.3 Outline

The remainder of this thesis is structured as follows. Chapter 2 surveys existing indoor localization approaches related to this thesis. In Chapter 3, the developed texton-based approach is presented and its components, the kNN algorithm and the particle filter, are introduced. Details about the synthetic data generation tool and map evaluation technique are also given. Chapter 4 describes the setup of the on-ground and in-flight experiments. The results of the experiments are presented in Chapter 5 and discussed in Chapter 6. Finally, we draw our conclusions and indicate future research directions in Chapter 7.

Chapter 2

Related Work

This chapter will discuss advantages and disadvantages of different approaches for indoor localization and contrast them to the proposed method. While there is a wide range of methods for indoor localization, from laser range scanners over depth cameras to RFID tag based localization, only methods that use the same technical setup—a monocular camera—are discussed. Two types of localization are distinguished: local techniques and global techniques [15]. Local techniques need an initial reference point and estimate a robot's coordinates based on the change in position over time. Once they lost track, the robot's position can typically not recovered. The approaches also suffer from "drift" since errors are accumulating over time. Global techniques are more powerful and do not need an initial reference point. They can recover when temporarily losing track and address the *kidnapped robot problem*, in which a robot is carried to an arbitrary location [14].

In general, comparing the accuracy and run-time of different localization methods is difficult: target systems and test environments are often too different to draw comparisons. The annual Microsoft Indoor Localization Competition[1] aims at setting a standardized testbed for comparing near real-time indoor location technologies. However, since the competition does not require lightweight platforms and allows the use of external infrastructure such as WiFi routers, no vision-only approach has yet been presented at the competition.

[1] `https://www.microsoft.com/en-us/research/event/microsoft-indoor-localization-competition-ipsn-2016/`

2.1 Vision-based Localization Methods

2.1.1 Fiducial Markers

Fiducial markers (Figure 2.1), which are often employed in augmented reality applications [20, 16], have been used for UAV localization and landing [13, 24]. The markers encode information by the spatial arrangement of black and white or colored image patches. Their corners can be used for estimating the camera pose at a high frequency. The positions of the markers in an image are usually determined with *local thresholding*. Local thresholding is a simple method for separating objects—salient image regions—from a background. Its output is a binary image with two states: foreground (markers) and background. Several widespread marker libraries exist, including ArUco and ARToolKit [20]. In several fiducial marker libraries, such as *ArUco* [17], marker positions are further refined by removing improbable shapes, yielding an adjusted version of possible marker positions.

An advantage of fiducial markers is their widespread utilization, so that one can fall back on experience of many studies; several libraries are technically mature and open-source. Given adequate lighting conditions, markers can be used in a wide variety of environments. This makes them suitable for indoor localization.

A drawback of the approach is that motion blur, which frequently occurs during flight, can hinder the detection of markers [3]. Furthermore, partial occlusion of the markers through objects or shadows break the detection; each marker needs to be fully in the camera view [19]. Another downside is that markers might be considered as visually unpleasant and may not fit into a product or environmental design [8]. They offer little flexibility, since one has to rely on predefined marker dictionaries; additionally, marker-based approaches require to *always* modify the environment. Like most vision-based approaches, the detection of markers is prone to changes in lighting setting and may not work in low-contrast

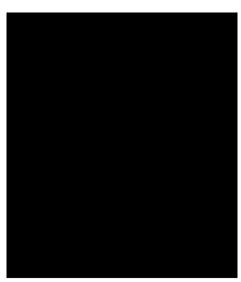

Figure 2.1: Examples of fiducial markers of the ArUco library.

2.1.2 Homography Determination & Keypoint Matching

A standard approach for estimating camera pose is detecting and describing keypoints of the current view and a reference image [26], using algorithms such as Sift [22], followed by finding a homography between both keypoint sets. A keypoint is a salient image location described by a feature vector. Depending on the algorithm, it is invariant to different viewing angles and scaling. The Sift algorithm transforms an image into a set of image features. It works in four subsequent stages:

1. *Maxima detection:* The image is convolved with the *Difference of Gaussian* blob detector. By varying the variance of the Gaussian distribution, the maxima—potential keypoints—across different scales and spaces can be detected.

2. *Refinement of keypoints:* The potential keypoints are refined using stability measures.

3. *Orientation detection:*

(i) keypoint detection and (ii) keypoint description. In the first stage, salient image locations are detected . To achieve scale invariance, the keypoints are scanned at a wide range of scales and the scale where the interest point features.

, i.e the 2D space wise signal variances, meet certain stability criteria, is selected as the one that gets coded by the descriptor.

By finding a homography, that is a perspective transformation between the keypoints of the current view and a reference image, the current view can be located in the reference image.

While this approach is used in frameworks for visual Simultaneous Localization and Mapping (SLAM), the pipeline of feature detection, description, matching, and pose estimation is computationally complex. Therefore, ground stations for off-board processing or larger processors are usually needed for flight control.

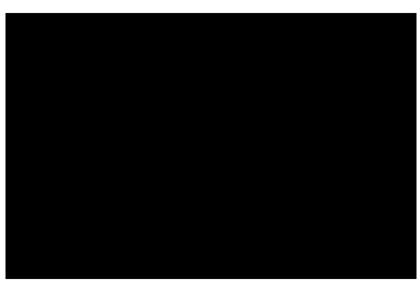

Figure 2.2: Perspective transformation between keypoints of the current image (left) and the reference or map image (right).

The approach has been used in absolute localization for UAVs: Blösch et al. [6] employ it on a $3.5\,m \times 2\,m$ area and achieve a root mean square (RMS) positional error below $10\,cm$ in x, y, z-direction. Calculations are executed on a powerful ground station, which is connected to the UAV with a USB cable. While subsequent research has brought the algorithm on board of the UAV [1], the required processing power is still too complex for small MAVs [9].

write about accuracy

2.1.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are specialized for image processing [**lecun1998gradient**] and have outperformed classical approaches for a large amount of computer vision challenges [**dosovitskiy2014discriminative**]. They consist of multiple neuron layers, which are able to represent increasing levels of abstraction [**lecun1998gradient**]. Usually, the training of such networks requires a considerable amount of time, in the range of hours and days. However, the prediction often takes only few milliseconds, shifting computational effort from the test phase to the training phase. CNNs been used as more robust alternative to SIFT in case of image perturbations [**dosovitskiy2014discriminative**], however, wit h larger computation time. In recent work, **kendall2015posenet** present the first framework for regressing camera positions based on CNNs [**kendall2015posenet**].

The approach works remarkably fast and is robust to different lighting settings, motion blur, and varying camera intrinsics.

Not really, only on powerful computers!

2.1.4 Optical Flow

Optical flow algorithms are biologically inspired methods for navigation—taking inspiration from insects and birds [25]. They estimate the motion based on the shift of corresponding image keypoints in successive images. Gradient based approaches, keypoint-based methods, and more specific methods have been put forth. They belong to the class of local localization techniques and can only estimate the position relative to an initial reference point. The approaches suffer from accumulating errors over time and typically do not provide a means for correcting these errors.

for example

Chao et al. [7] compare different optical flow algorithms for the use with UAV navigation. The approaches are computationally rather complex [23]. To render on-board odometry feasible for small MAVs, McGuire et al. [23] introduce a lightweight optical flow variant. The algorithm uses compressed representations of images in the form of edge histogram to calculate the flow.

2.2 Texton-based Machine Learning Methods

Textons are small characteristic image patches that can be used as image features. Varma et al. [27] originally introduced textons for classifying different textures, showing that they outperform computationally more complex algorithms, like Gabor filters [27]. For the classification, the approach compares texton histograms between a training set and the test sample and the class of the closest training sample is assigned to the test sample. To extract histograms in the *full sampling* setting, a small window—a kernel—is convolved over all image positions and the frequency of textons is calculated. Instead of convolving a kernel over the entire image, the kernel can be applied at randomly sampled image position [10]. Modifying the number of samples allows for adjusting the computational effort, resulting in a trade-off between accuracy and execution frequency. A disadvantage is that it discards all information about the spatial arrangement of textons—it does not make use of the *Where* of the information, just of the *What*, which can result in different images with similar histograms.

De Croon et al. [9] use textons as image features to distinguish between three height classes during flight. Using a nearest neighbor classifier, their approach achieves a height classification error of approximately 22 % on a hold-out test set. This enables a flapping-wing MAV to roughly hold its height during an experiment . In another work, De Croon et al. [11] introduce the *appearance variation cue*, which is based on textons, for estimating the proximity to objects [11]. Using this method, their MAV is able to avoid obstacles in a $5m \times 5m$ office space and achieves an AUC of up to .97 on rather distorted images.

Chapter 3

Methods

This section describes the ideas behind the developed approach, the hardware and software implementations and the design of the experiments. An overview of the entire process is sketched out in Figure 3.17. All developed software is made publicly available, the corresponding links are given in the respective sections.

3.1 Hardware and Software of the Platform

In a first setting, the commercially available Parrot AR.Drone2.0 was equipped with an Odroid XU-4 single board computer, a Logitech 525 HD webcam, a WiFi module. Figure 3.18 shows the setup. The camera images were processed on the more powerful Odroid processor and the resulting x, y-estimates were sent over a USB data link to the MAV flight controller. The Odroid processor has a full operating system (Ubuntu 15.04) and allows for running arbitrary Linux software. However, the additional weight through the modifications of the system resulted in unstable flight performance. Therefore, we modified the system to execute the localization algorithm directly on board of the MAV and not on the additional Odroid processor. To this end, the software had to be ported from the the high-level language Python to the low-level language C without being able to rely on many existing software libraries. Finally, this removed the need for additional payload and made flight performance very stable. Also, it reduced the need to buy and attach an external processor and a point of failure. Another advantages is that the framework can be easily ported to any UAV supported by the paparazzi software. The major disadvantage is that the on-board processors of many MAVs have a lower performance than the Odroid processor.

We decided to use the quadcopter *Parrot Bebop Drone* as prototype for all our tests. In general, quadcopters allow for navigating in arbitrary directions without changing their yaw angle. Additionally, several models show stable flight behavior, resulting in noise-free images. The *Parrot Bebop Drone* is equipped with a lithium-ion polymer battery that allows for approximately 11 minutes flying time. The UAV's dimensions are $28 \times 32 \times 3.6$ cm and it weighs 400 g. The UAV is equipped with two cameras: a front camera and a downward-looking bottom camera. The proposed approach makes use of the bottom camera only. This camera has a resolution of 640×480 pixels with a frequency of 30 frames per second. The UAV's processor is a Parrot P7 dual-core CPU Cortex A9 with a tact rate of 800 Mhz. It is equipped with 8 GB of flash memory. It runs a Linux operating system. The full specifications of the UAV can be found at `http://www.parrot.com/products/bebop-drone/`.

The original Bebop SDK was replaced with Paparazzi. Paparazzi is an "open-source drone hardware and software project encompassing autopilot systems and ground station software". Its modular approach allows for combining functions regarding stabilization, localization, and control of UAVs. The proposed approach is implemented as module in Paparazzi's computer vision framework. Since low-level routines like accessing camera information or attitude control for different platforms are already implemented in Paparazzi, the proposed module can be readily used across different platforms. Modules are written in the C programming language and are cross-compiled on the host PC to make them suitable for the UAV's processor. Afterwards, they are uploaded to the microprocessor of the UAV, allowing for autonomous execution of the compiled software. A downlink connection—from the UAV to the groundstation—allows for monitoring the state of the aircraft (e.g., speed, altitude, position, battery status).

While the goal of the proposed system is to achieve an autonomous system, monitoring the system is important for evaluation and safety considerations. Figure 3.1 shows the default ground control station of Paparazzi. For collecting data, real-time visualization of the position estimates and enabling future users to keep track of the system states, a graphical user interface (GUI) has been developed in the scope of this thesis. The GUI is displayed in Figure 3.2.

The GUI displays the position estimate of the proposed framework, the ground truth position based on OptiTrack, the texton histogram, the Euclidean distances between OptiTrack and the system, the uncertainty of the system, and the correlation between the uncertainty and the Euclidean distance between the Optitrack and treXton estimates. The GUI can be accessed using the *Tool* menu in paparazzi.

Figure 3.1: The ground control station of the Paparazzi software. It displays information about the status of the UAV and allows for controlling the vehicle.

Figure 3.2: Screenshot of the GUI. It displays the position estimate of the proposes framework, the ground truth position based on OptiTrack, the texton histogram, the Euclidean distances between OptiTrack and the system, and the uncertainty of the system (based on the spread of particles).

3.2 Pillar I: Training Set Generation

A main idea of the presented method is to shift computational effort to an offline phase, before the MAV is employed in a fixed environment. This shift is accomplished by using a light-weight machine learning approach instead of a computationally more complex algorithm, such as homography finding based on salient image keypoints. Supervised machine learning methods need a training set to find a mapping from features to target values. In a computer vision-task, the features represent image characteristics, such as average pixel values or the amount of oriented gradients.

The proposed method is based on histograms of *textons*. To create the training set, a convenient way is to align image features—texton histograms—with high-precision position estimates from a motion tracking system. These systems can track rigid bodies at a high frequency within an error of few millimeters. This allows for creating high-quality training data sets. Even low quality images can be mapped to the corresponding x, y-position.

This is achieved by saving the texton histogram on the MAV's hard disk. The corresponding position from the motion tracking system is broadcast to the UAV via the ground station. The ground station receives the positions from the motion tracking system. The major disadvantages of the approach are that motion tracking systems are usually expensive and time-consuming to move to different environments.

As an alternative, we sought a low-budget and more flexible solution. While the homography-based approach (Section 2.1.2) shows high matching quality for non-blurry images, it suffers from lacking robustness to noise and the high processing time requirements. To exploit the advantages, and straighten out the disadvantages, we used the approach in a preprocessing step to obtain labeled training data of a known environment. This allows to shift computational effort from the flight phase to a pre-flight phase—paving the way for autonomous flights of MAVs with limited processing power. It is based on two associated cornerstones that are performed in an offline phase: the creation of an *orthomap* and finding a homography. The required image dataset for both cornerstones can be obtained by using in-flight pictures of a manual flight or by taking pictures with a hand-held camera.

For creating a map, the images from the dataset have to be stitched together to get a hyperspatial image of the scene. The stitched image has a higher resolution than the single images and contains a greater range of detail. The stitching step includes challenges: a subset of the recorded images might be distorted and perspective transformations can impede the stitching process. Certain software packages allow for orthorectifying the images by estimating the most probable viewing angle based on the set of all images. However, since a downward-looking camera is attached to the UAV, most images will be roughly aligned with the z-axis, given slow flight.

For the stitching process, the software Microsoft Image Composition Editor (ICE) was used. The stitching process can be time-consuming and error-prone. In some environments, an image from a top view point can be taken capturing the entire area with high-resolution camera, circumventing the need for stitching together multiple images. Yet another method starts with an existing image and modifies the environment accordingly—for example by painting the floor or printing posters–to correspond to the image. In all cases, we will describe the environment as *map image*.

Keypoints of the current image and the map image are detected and described using the SIFT algorithm. The keypoint sets are further refined using Lowe's ratio test [22]. This is followed by a matching process, that identifies corresponding keypoints between both images. The matching uses a 'brute-force' matching scheme and every keypoint is compared to every other keypoint. These matches allow for finding a homography between both images. For determining the x, y-position of the current image, the center of it is projected on the reference image using the homography matrix. The pixel position of the center in the reference image can be used to determine the real world position by transforming the pixel coordinates to real-world coordinates, based on the scale factors C_x and C_y, with $C_x = \frac{width(R)}{width(I)}$ and $C_y = \frac{height(R)}{height(I)}$, where W is the real-world representation and I the digital pixel image. This yields a dataset of images, labeled with x, y coordinates and the number of matches. This process already introduces noise into the dataset, depending on the quality of the keypoint matches.

Figure 3.3: This figure shows the created orthomap of a rather texture-rich floor. It is stitched together using 100 single images and represents a real world area of approximately 8×8 meters. Image distortions, non-mapped areas, and slightly skewed seams at several points are visible.

3.3 Pillar II: Texton-based Approach

In this section, the core of the proposed algorithm, the implementation of the proposed texton framework is described. The proposed approach requires a dictionary of textons, such that the histograms can be determined. The histograms are then used as features for the k-Nearest Neighbors (kNN) algorithm. The outputs of this regression technique are possible x, y-coordinates for a given image.

Importantly, for generating the training dataset, no subsampling was used. Therefore, the training dataset will not show any variance, if the same images were used.

In the following pseudo code, M represents the number of particles, z_t^x the output of the texton framework at time t, and f_t^x the estimated flow at time t.

3.3.1 Texton Dictionary Generation

For learning a suitable dictionary for an environment, image patches were clustered. The resulting cluster centers—the prototypes of the clustering result—are the textons [28]. The clustering was performed using a competitive learning scheme with a winner-take-all strategy. In the beginning, the dictionary is initialized with 20 random patches from the first image, which form the first guess for cluster centers. Then, new images patches are extracted and compared to each texton in the tentative dictionary using the Euclidean distance. The most similar texton to the current patch is declared as the "winner". This texton is then adapted to be more similar to the current patch, by calculating the difference in pixel values between the texton and the current images patch, and updating the texton with a learning rate of $\alpha = 0.02$. The first 100 images of each dataset were used to generate the dictionary. For each image, 1000 randomly selected image patches of size $w \times h = 6 \times 6$ were extracted, yielding $100,000$ image patches in total that were clustered. An example of a learned dictionary can be found in Figure 3.4. Different maps and environmental settings require different textons. If one would use the same dictionary for each map, it might happen that the histogram has only a few non-zero elements, and thus, cannot represent the variance in the map. While we set the number of textons to 20 for all maps, this parameter is also map-dependent, and ideally, if time allows, could be adapted to the given map.

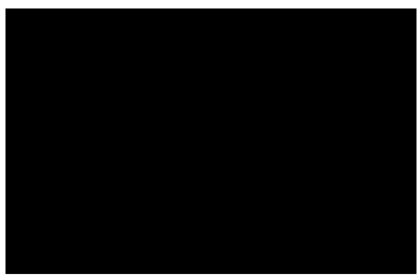

Figure 3.4: 20 textons. The grayscale textons are six pixels high and 6 pixels wide.

3.3.2 k-Nearest Neighbors (kNN) algorithm

The k-Nearest Neighbors (kNN) algorithm is the machine learning-core of the proposed algorithm. For a given histogram that is derived from the current camera image, the k-Nearest Neighbors (kNN) algorithm measures the similarity of this histogram to all histograms in the training dataset and outputs the k most similar training histograms and the corresponding x, y-positions.

While the kNN algorithm is one of the simplest machine learning algorithms, it offers several advantages: it is non-parametric, allowing for the modeling of arbitrary distributions. Its capability to output multiple predictions allows for neat integration with the proposed particle filter. Its simplicity comes together with transparency: it allows for spotting the possible sources of error such as wrongly labeled training examples.

While the naive approach in using kNN for regression calculates the mean of the k outputs, we decided to use a more complex method. This motivation is visualized in Figure 3.6: If $k = 2$ and the output values are distant to each other, averaging them would yield a value in the middle, which is with high certainty not the correct position. Over time, however, the ambiguity, can be resolved, when both estimates of the kNN model fall together. Compared to the Kalman filter, which is displayed in Figure 3.5, the full Bayesian filter can immediately find the correct position. Since a full Bayesian filter is computationally complex, a variant that is based on Monte Carlo sampling was used: the particle filter. A more detailed description of the filtering technique can be found in the next section.

Figure 3.5: Four time steps of a Kalman filter.

Figure 3.6: Four time steps of a Bayesian filter.

It often outperforms more sophisticated algorithms. A frequent critique regarding the kNN is its increasing computational complexity with an increasing size of the training dataset. However, its time complexity can be reduced by storing the training examples in an efficient manner, such as a binary tree structure. However, all of our training datasets were below 1000 images, resulting in fast predictions based on a list structure.

Figure 3.7: The main part of the particle filter.

Figure 3.8: Resampling wheel for particle resampling.

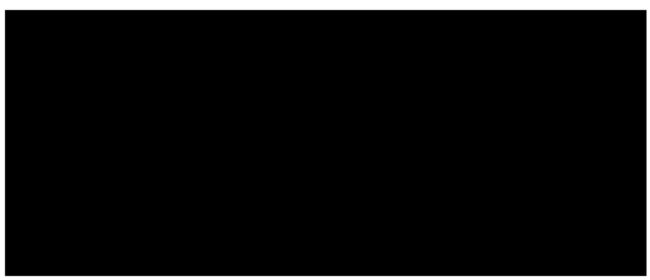

Figure 3.9: Initialize particles.

3.4 Filtering

Computer vision-based estimations are usually noisy or ambiguous, and so is the proposed system: beginning with the estimations of the homographies, the ground truth is already based on possibly faulty estimates. Texton histograms obtained during flight will not perfectly match the ones in the training data set: blur, lighting settings, viewing angles and other variables change the shape of the texton histograms.

To filter out outliers and smooth the estimations, a popular filter choice is the Kalman filter. However, the Kalman filter is not able to represent multimodal

Figure 3.10: High-level pseudocode for texton framework.

probability distributions. This makes it rather unsuitable for the presented approach: if the kNN model outputs two predictions, one would need to use average of these predictions and feed this value to the Kalman Filter. This approach can lead to biased predictions, especially, if the the model outputs belong to distant locations— to similar texton distributions at these positions.

Instead, the more powerful general Bayesian filter could simultaneously keep account of both possible locations and resolve the ambiguity as soon as one location can be favored. In this case, the predictions of the k neighbors are directly fed into the particle filter without averaging them first. The filter is able to smooth the estimations, handle uncertainty, and simultaneously keep track of several competing position estimations. Since the calculation of a full Bayesian filter is computationally intractable, a particle filter which is based on sampling was used as approximation.

In general, particle filters estimate the posterior probability of the state given observations. Specifically, this means that finding the position of the UAV can be described as $p(X_t \mid Z_t)$, where X_t is the state vector at time t (x, y-position, heading, speed, acceleration) and Z_t are the measurements ($z_1, ..., z_t$, where each z_i represents the x, y output of the proposed algorithm) up to time t. The state vector is *hidden*, since the variables cannot be measured directly, therefore the situation can be described using a hidden Markov model. Instead, noisy or ambiguous data can be obtained through the proposed algorithm.

The weighted particles are a discrete approximation of the posterior probability function (*pdf*) of the state vector.

Particle filters have several advantages. First, one can represent uncertainty by the variance of the state variables of the particles. Second, the particle filter allows for *sensor fusion*, and can integrate IMU data or optical flow into the position estimation. A major disadvantage is the rather high computational complexity. This can be circumvented by reducing the amount of particles (trading off speed and accuracy)—allowing for adapting the computational payload to the used processor.

The used particle filter is initialized using 100 particles at random x, y-positions. To incorporate the distances, the sensor model $p(z \mid x) = \frac{p(x|z)p(z)}{p(x)}$ is used, where $\mathbf{x} = ((x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k))^T$. A two-dimensional Gaussian model was used for each point. The parameters of the Gaussian have been determined by comparing of the positions based on the motion tracking system with the predictions of the proposed system. This results in values for the variances in x and y, the correlation ρ between x and y. The mean values μ were set to zero (no-systematic bias). Figure **??** shows the results of one such evaluation.

Figure 3.11: Dependence structure between similarity of histograms and measurement error. TODO: Check if it is the distance or the similarity and if I used cosine similarity or Euclidean distance.

Figure 3.12: Dependence structure between similarity of histograms and measurement error in y-direction

This allows to make use of the information in all k neighbors and keep track of a multimodal distribution. While keeping track of a multimodal distribution allows for incorporating several possible states of the system, the system needs one best position estimate for guidance and stabilization after each iteration. Using a weighted average of the modes would again introduce the problem, that the weighted average could fall into a low density region. Therefore, the maximum a posteriori estimate, as described in [12] was used. This approach uses the following formula to obtain the MAP estimate:

Therefore, the final position estimate is equal to the position of one of the particles.

Figure 3.13: Measurement model showing the delta x and delta y positions TODO: Add Gaussian approximation!

$$s_k^{MAP} = \arg\max_{s_k} p(s_k \mid Z_k) \tag{3.1}$$

$$= \arg\max_{s_k} p(z_k \mid s_k) p(s_k \mid Z_{k-1}) \tag{3.2}$$

Therefore, the MAP estimator in our case is

$$s_k^{MAP} = \arg\max_{s_k} \lambda^M(s_k) \tag{3.3}$$

with

$$\lambda^M(s_k) = p(z_k \mid s_k) \sum_{j=1}^{M} p(s_k \mid s_{k-1}^j) w_{k-1}^j \tag{3.4}$$

This function is now only evaluated at a finite, chosen number of states, the particles, using

$$\hat{s}_k^{MAP} = \arg\_s_k \in \{s_k^i \mid i = 1, \ldots, N\} \lambda^M(s_k) \tag{3.5}$$

In this formula, $p(z_k \mid s_k)$ is the likelihood of the particle s_k given the current measurement z_k. In our setting, this probability is equal to the weight of the particle z_k, therefore $p(z_k \mid s_k) = w_k^i$.

The estimation of *uncertainty* is a core part of the proposed approach, being important for safety and accuracy. Therefore, uncertainty was modeled using the spread of the particles.

In every time step, the particles of the filter get updated based on the optical flow estimates. These estimates are noisy, as illustrated in Figure **??**. Additionally, optical flow estimates aggregate noise over time, since each estimate is dependent on the previous one, leading to drift (*relative position estimates*). In contrast, the machine learning-based method makes independent predictions. While this allow for avoiding accumulating errors, the predictions do not dependent on each other, and might be 'jumping' between two points. To combine the advantages of both methods, and leverage out the disadvantages, the particle filter is used.

An idea was to include the similarity to the neighbors as confidence value, thus reducing the measurement noise, if a high similarity between current histogram and a training histogram is achieved. However, we found no correlation between these variables. Figures 3.11 and 3.12 displays the dependence structure.

Another idea—if the homography-based approach is used for labeling—was to use the amount of detected keypoints (K) as confidence value. However, again, no linear relationship between K and the error in $x-direction$ (X) and the error in $y - direction$ (Y) could be found (Figure 3.11): $\rho_{K,X} = 0.10$, $\rho_{K,Y} = 0.01$.

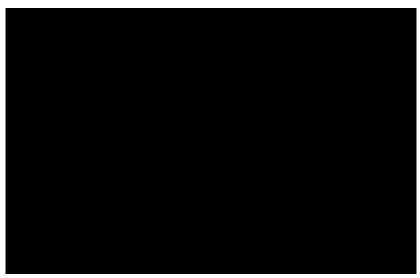

Figure 3.14: Dependence between number of keypoints and error in x-direction.

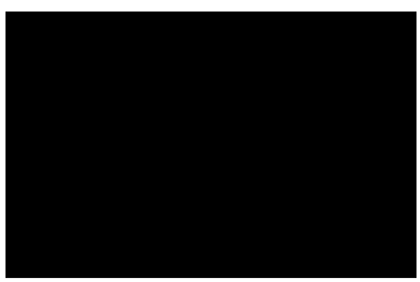

Figure 3.15: Dependence between number of keypoints and error in y-direction.

Figure 3.16: Training dataset generation if the motion tracking system is used. The texton histograms of the camera images during flight are extracted and aligned with the highly accurate position estimates of the motion tracking system. The result is a high-quality training set of texton histograms and corresponding x, y-positions.

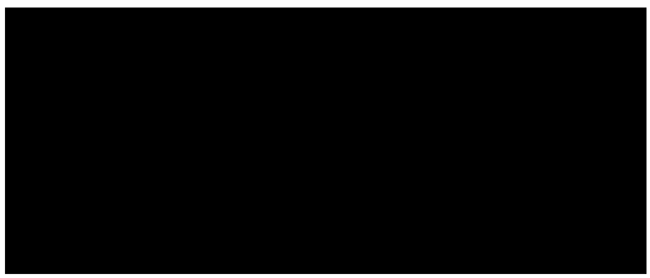

Figure 3.17: The figure illustrates the workflow of the proposed approach. After obtaining images from an initial flight, the recorded images are stitched together to create an orthomap. The same images are used to detect and describe their keypoints using SIFT, followed by finding a homography between the keypoints of the flight images and the orthomap. The center of the homography is used as x, y coordinate for labeling the training set. Additionally, the number of detected matches is saved for the corresponding x, y estimate. Each image of the initial flight data set can be used and a fixed amount of small 6×6 pixel image patches can be extracted. These can be labeled with the nearest texton, using a distance measure, for example, Euclidean distance. Finally, a classifier can be trained using texton histogram as feature vector and the corresponding x, y coordinate as target value. This process allows shifting computational burden of the keypoint detection and homography finding to a faster machine learning approach.

Figure 3.18: Comparison of an unmodified Parrot AR.Drone.2.0 (left) and a modified version (right). The modified one was equipped with an Odroid XU-4 single board computer, a Logitech C525 HD camera, a WiFi module, and a USB connection between the Odroid board and the AR.Drone.2.0 flight controller.

3.5 Pillar III: Map evaluation

3.5.1 Synthetic Data Generation

In the scope of this thesis, an application to simulate different camera positions during flight was created. It generates synthetic image patches based on perspective transformations of a map image. Examples of generated images are displayed in Figure 3.19. The application allows for comparing and predicting the performance of different maps. The software is written in C++ and OpenCV 3.0.0.

The software is able to generate a specified amount of image patches using random values for rotational angles, translational shifts, as well as blur, contrast and brightness intensity. The values are sampled from different probability distributions, see Table 3.1 for a summary. An additional graphical user interface (GUI) displays the result of applied transformations and saves the generated images.

To simulate camera movements in 3D space, a 2D to 3D projection of the image is performed first. Then, by building separate rotation matrices R_x, R_y, and R_z around the axes x, y, and z, the rotations can be performed separately. Next the rotation matrix R is created by multiplying the separate matrices, i.e., $R = R_x \times R_y \times R_z$. The 3D translation matrix is multiplied by the transposed rotation matrix. This step is crucial to rotate the *camera model* and not the image itself. Finally, after performing all steps, a projection from 3D space to 2D is applied, to obtain the transformed image.

Distribution					
Uniform ($\mathcal{U}$)			Normal ($\mathcal{N}$)		
Parameter	Min	Max	Parameter	M	STD
Yaw	-5	5	Roll	90	3
Translation X	-800	800	Pitch	90	4
Translation Y	-500	500	Brightness	2	0.1
Height	100	700			
Blur	1	10			
Contrast	2	3			

Table 3.1: The table shows the used distributions for the different parameters of the synthetic data augmentation tool.

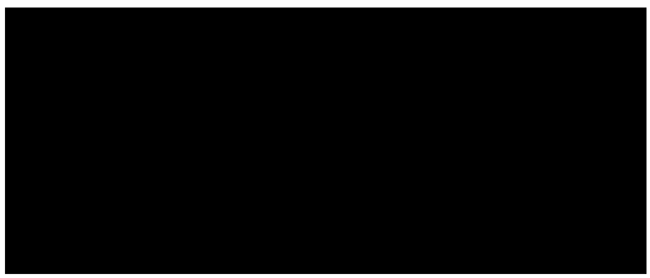

Figure 3.19: Six images generated by the synthetic data generation tool.

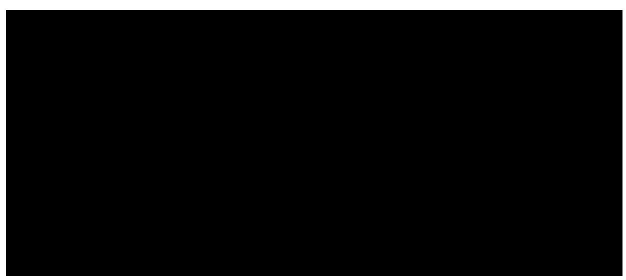

Figure 3.20: Camera model for the synthetic flight.

3.5.2 Evaluation Scheme

The performance of the proposed method depends on the environment it is used in: obviously, a texture-rich environment without repeating patterns will be better suited than an uniformly colored environment. Ideally, one would like to know if the proposed method is able to work in a given environment. Therefore, we propose an evaluation scheme for given maps. This scheme assigns a global fitness value to a given map, proportional to the expected accuracy it is expected to achieve in the physical world. The evaluation scheme allows to inspect the given map and to detect regions that are responsible for the overall fitness value. The evaluation of a map is difficult, since the obtained histograms during a real flight depend on many factors: motion blur, distance to the map and rotations proportional to the map.

In the first step of the map evaluation procedure, N different patches of a given map are generated using the tool *draug* (Section 3.5.1). We propose the following global loss function (L) for evaluating a given map ($\mathcal{M}$):

$$L(\mathcal{M}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \ell(d_a(h_i, h_j), d_e(h_i, h_j)) \tag{3.6}$$

$$\begin{aligned} \ell(x, y) &= x - y && (3.7)\\ d_a(h_i, h_j) &= \mathrm{CS}(h_i, h_j) && (3.8)\\ d_e(h_i, h_j) &= f_X(pos_i \mid \mu, \Sigma) = f_X(x_i, y_i \mid \mu, \Sigma) && (3.9)\\ & && (3.10) \end{aligned}$$

$$\begin{aligned} \mu &= pos_j = (x_j, y_j) && (3.11)\\ \Sigma &= \begin{bmatrix} \rho & 0 \\ 0 & \rho \end{bmatrix} && (3.12) \end{aligned}$$

The cosine similarity (CS) is defined as:

$$CS(h_i, h_j) = \frac{h_i^T h_j}{||h_i|| \, ||h_j||} \tag{3.13}$$

The cosine similarity has the convenient property that its values are bounded between -1 and 1. In the present case, since the elements of h_i and h_j are non-negative, it is even bounded between 0 and 1. The function f_x describes the non-normalized probability density function of the normal distribution: $f_X(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. This function is also bounded between 0 and 1, which makes the functions f_X and CS easily comparable.

The idea behind the global loss function L is that histograms in closeby areas should be similar and the similarity should decrease the further away two positions are. This is modeled as a 2-dimensional Gaussian with zero covariance. The variance is depended on the desired accuracy (ρ): the lower the variance, the more punctuated a certain location is but also the higher the risk that a totally wrong measurement occurs. The following visualization are based on color histograms (and not texton histograms) for easier visual analysis.

In future research, the "bad regions" could be optimized using an optimization approach such as an evolutionary algorithm. It also allows to show the similarities for a fixed position, and the loss for a fixed positions based on the expected similarity and the actual similarity.

Figure 3.21: TODO: add colorbar! Heatmap colors shows losses per region (smoothed using a Gaussian filter). Ideal histogram similarity for a given position. Histograms taken at positions close to $\mathbf{x} = (400, 300)$ should be similar to this histogram. The further away the position of a certain histogram, the lower the ideal similarity should be.

Chapter 4

Analysis

In this chapter, the setup of the conducted experiments is presented. We examine different parameter choices in Experiment 1 to 6 in on-ground experiments using recorded data. Afterward, the found parameters are used to show the validity during flight in Experiments 7 and 8. The experiments were conducted in TU Delft's CyberZoo—an indoor flight arena of size $w \times l \times h = 10\,m \times 10\,m \times 7\,m$, with relatively constant lighting settings due to primarily artificial lighting.

The choice of the parameters is dependent on the environment and the size of the training dataset. Therefore, there is no general optimal parameter. Instead, the parameters have to be adapted to the particular environment. Since the proposed algorithm is intended for known environments, this is always possible.

4.1 Analysis – Determining the Number of Textons

The developed framework allows to tune the computational complexity by modifying the number of extracted image samples. To increase the speed of the algorithm, the goal is to use as few samples as possible. To determine a suitable number of extracted samples, in this experiment, the average cosine similarity between $D = 20$ datasets of histograms is compared. Each dataset consists of $N = 10300$ histograms. The independent variable is the number of extracted image patches M. The histograms were generated using the same images. Due to the random sampling of the extracted image patches, the histograms of each datasets will differ. This deviation will be measured using the cosine similarity. Therefore, each of the D datasets was compared to all the other $D-10$ datasets and the average cosine similarity was determined as well as the standard deviation of the cosine similarity was measured. Comparing the cosine similarity between the histograms has the advantage that the number of samples can be determined independent of a specific task.

4.2 Analysis – Setting the Baseline for kNN and determining k

In a standard setting, the training error ϵ_t of a k=1-nearest neighbor algorithm is $\epsilon_t = 0$ because the nearest neighbor of the sample will be the sample itself, given that each feature vector is unique. However, in this scenario, we deal with random sampling such that each image will be represented by a slightly different histogram each time the histogram is extracted.

Figure 4.1: The created map that was stitched together using 445 images. A non-mapped area in the middle of the map can be seen, which is a result of the set flight path. An image distortion can be seen at the right-hand side, where the landing spot sign appears twice, while in reality, only one circle was visible.

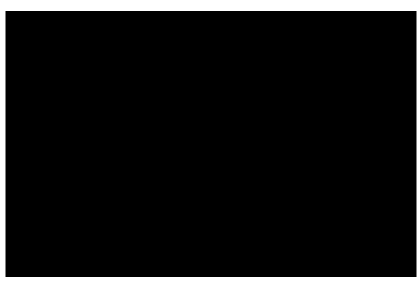

Figure 4.2: The estimates of the homography method compared to the ground truth of the motion tracking system. TODO: Legend

	x-position	y-position
Error in cm	31	75
STD in cm	73	369

Table 4.1: Error statistics for the homography method.

4.3 Experiment – Motion tracking system for Stabilization, Texton-based Approach for estimation

4.3.1 Training Set based on Motion Tracking System

In this experiment, a fixed route was set using the ground control station. While the stabilization and guidance were performed using the motion tracking system, the position estimates were performed onboard of the MAV using the texton-based approach. The Euclidean distances between the estimates of the motion tracking system and the texton-based approach were measured separately for the x- and y-direction. The experiment uses the particle filter and the textons only. The training dataset was composed of 500 images recorded at an height of approximately 1 m, recorded in the time span of one hour before the experiment. The corresponding x, y-coordinates were obtained from the motion tracking system.

4.3.2 Training Set based on Homography-finding Method

In this experiment, the training dataset was created using the homography-finding method. Apart from that, the settings are the same as in Experiment 4.3.1.

4.4 Experiment – Triggered Landing

In the triggered landing experiment, the UAV was programmed to land as soon as its position estimates are in the landing zone. The landing zone was defined as a circle with a radius of $60\,cm$. The x, y-coordinate of the center of the circle was specified in the flight plan. A safety criterion based on the variance of particles was introduced, such that the landing is only performed if the criterion holds. The criterion parameter was set to 60 cm in both $x-$ and y-direction. The experiment was conducted on a $5m \times 5m$ map and the ground truth was based on the position data from the motion tracking system. The training dataset was composed of 500 images recorded at an height of approximately 1 m, recorded in the time span of one hour before the experiment.

4.5 Experiment – Determining the Frequency

The frequency of the proposed algorithm is determined by varying the number of samples in the texton-based approach, the number of textons, and the number of particles of the particle filter.

4.6 Experiment – Comparing different possible maps

For the map comparison, 50 possible maps have been collected using the search term 'wallpaper' in Google's image search. This search term was used, since it is (i) a general term, without any specific image categories, (ii) wallpapers are likely to have a high resolution, and (iii) wallpapers are likely to be visually pleasant since they are often used as desktop backgrounds. The criteria for the images were that their minimum resolution was 1920×1080. Images with a higher resolution were converted to 1920×1080. For each image, we generated 1000 random image patches using the simulation method described in Section **??**, followed by the histogram extraction method. This yielded a labeled dataset of histograms and corresponding positions. For each map, we determined the expected overall loss based on the method described in Section 3.5.1. The compared maps were then sorted according to their estimated loss.

Chapter 5

Results

5.1 Experiment – Determining the Number of Textons

Figure 5.1 displays the cosine similarity of histograms in relation to the number of samples and the corresponding standard deviations.

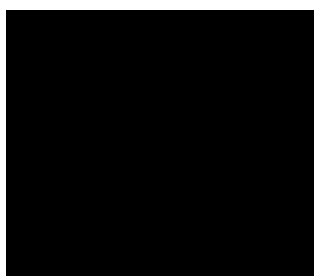

Figure 5.1: Cosine similarity of histograms in relation to the number of samples *Right*: Standard deviation of cosine similarity in relation to the number of samples. The squares indicate the positions at which the dependency was evaluated.

Figure 5.2: Estimates of the texton-based approach

5.2 Experiment – Real-Time Position Estimation

	x-position	y-position
Error in cm	46	54
STD in cm	56	71

Table 5.1: Estimates of the texton-based approach

5.3 Experiment – Triggered Landing

Table 5.2 shows the results of the triggered landings. Four out of six landings were correctly performed in the landing area. The mean distance of outliers was 16 cm.

	x-position	y-position
Error in cm	TODO	TODO
STD in cm	TODO	TODO

Table 5.2: Results of the triggered landings

5.4 Experiment – Comparing different possible maps

In Table 5.3, the results of the map evaluation procedure for the $N = 46$ maps are shown.

Statistic	Value
mean	0.57
median	0.55
standard deviation	0.14
max	0.99
min	0.24

Table 5.3: Results of the map evaluation procedure on synthetic data

Figure 5.3 shows the map with the highest global loss value and the map with the lowest global loss value.

Figure 5.3: Image with the lowest loss value; *Right*: Image with the highest loss value

5.4.1 Time requirements

Chapter 6

Discussion

In this chapter, the results of the experiments are discussed with regards to accuracy, frequency, and future improvements. Afterward, in the General Discussion, the research questions are addressed and discussed.

The comparison between sub-sampling and full sampling (Experiment 5.1) has shown that only a small part of the maximum amount of samples is necessary. In fact, $\frac{400}{640 \times 480} = 0.13\,\%$ of the maximum amount of samples suffices to achieve cosine similarities between the texton histograms larger than 99 %. This set the stage for large speed-ups during live operation. Additionally, the technique allows for further speed-ups depending on the processing power of the platform at hand.

In the real-time position estimation experiment (Experiment 5.2), the initial mean error was rather large: 130 cm in x-direction and 90 cm in y-direction. A more in-depth analysis revealed that the estimates of the particle filter were lacking behind the position estimates of the motion tracking system. This was due to the simple motion model of the particle filter that was based on Gaussian noise only. By shifting the estimates of the particle filter by six frames, that is approx. 0.46 seconds at a frequency of 13 Hz, the error could be reduced to 46 cm in x-direction and 54 cm in y-direction. This lag can be addressed by to strategies: (i) slower speed during flight, (ii) a better or more flexible motion model.

The triggered landing (Experiment 5.3) showed a high accuracy. While the majority of landings were triggered inside the landing zone, two out of the six landings were outliers. However, their distance to the landing area were rather small, with an average distance of 16 cm. A comparison without the safety criterion, showed, that the criterion can have advantages.

The evaluation of different maps using the synthetic data showed considerable differences between the evaluated images. The range of losses from 0.24 to 0.99

clearly shows the different suitabilities of different maps for the proposed algorithm. In order to visually evaluate the proposed map evaluation technique, a simple map was constructed with two repeating tiles. The image with the minimum and the one with the maximum loss value based on their color histogram are shown in Figure 5.3. The different patterns of the images are clearly visible: while the image with the minimum value fulfills the desired properties—closeby areas have similar color values, distant areas are dissimilar, the image with the maximum loss is mainly black resulting in similar histograms all over the place and leading to high loss values. This initial evidence can be taken to test the predictive power of the evaluation algorithm for texton histograms.

6.1 General Discussion

In this chapter, the results will be discussed with regard to error statistics, execution frequency, robustness, and scalability. To begin with, we recapitulate the research questions:

- R1: Can accurate 2D positions be estimated in real-time, using a machine learning-based approach on a limited processor in a modifiable indoor environment?
- R2: Is accurate real-world localization regression or classification possible when the training data comprises synthetic data only?
- R3: Can we predict the goodness of a given map for the proposed localization approach?

Regarding R1, the conducted experiments provide supportive evidence that a texton-based machine learning approach is able to accomplish real-time indoor localization. The proposed algorithm runs with as frequency of 30 Hz on a single board computer with limited CPU. Shifting processing power to an offline training step and relying on random sampling are the cornerstones for running the algorithm on processors with limited CPU. Despite the small ratio between extracted image patches (s) and the maximum amount of different image matches ($s*$), the accuracy is hardly affected, and only slightly improves when incorporating more textons.

Regarding research question R2, the initial idea—to use the synthetically generated images directly as training data—was not successful and not further followed up. This might be also the reason that only few projects have used synthetic images for real-world phenomena. The reality gap between the synthetic data and real-world data was huger than expected. Figure 15 shows an example of two image patches, one synthetically generated, one taken with the camera of the MAV. While the patches can be easily identified as similar for human eyes, the texton maps, where different colors represent different textons, are dissimilar. Blur, lighting settings, and camera intrinsics modify low-level features of the image to a too strong extent. A possible improvement might be to find a mapping from histograms of synthetic images to histograms of real images, by mapping 'synthetic textons' to 'real-world textons'.

Referring to R3, we found some initial evidence that the proposed map evaluation generalizes to the real-world. In contrast to R2, the generalization from the synthetic data to real-world data is of a different nature in this case. The requirement here is that maps that follow the ideal similarity distribution in the synthetically generated images also follow this distribution after being recorded with a camera. Or stated differently, for maps with a low loss value, distant image positions should not have similar histograms using the synthetic images nor the real-world images.

Despite the overall promising results, we noticed drawbacks of the proposed approach during the flight tests and directions for future research.

The accuracy—that is the difference between the estimates of the motion tracking system and the texton-based approach— could be further improved by incorporating more features, for example histogram of oriented gradients. Investigating further regression techniques, like Gaussian processes or Bayesian networks that can inherently handle space and time could be a worthwhile endeavor.

The developed method sets the stage for numerous future research directions and improvements. The current implementation assumes rather constant height (up to few centimeters) and no angular rotations of the MAV. While a quadroter can move in every direction without performing yaw movements, using the MAV on another vehicle could require arbitrary yaw movements. In order to limit the complexity of the dataset, a "derotation" of the incoming image could be performed to align it with the underlying images of the dataset. While the current approach normalizes each 5×5 image patch to unit mean and zero variance—giving robustness to different lighting conditions—this procedure could be further extended, for example by using specific color models.

While the current map evaluation approach used existing fixed images, it could also serve as a fitness function for an optimization approach—for example, an evolutionary algorithm—which modifies a given image. While the solution for a desired loss value of 0 or near 0 might be unique and independent of the original image, a higher loss value might change the initial image only to a certain extent, yielding an "improved version of the image", which is better suited for the proposed algorithm.

This could allow to find a near optimal solution for a given regression technique and give insightful view in the underlying structure of certain regression techniques.

Currently, *draug* generates image patches based on drawing samples from parametric distributions. This was motivated by the fact that an ideal map should be independent of previous estimates and based on single images only—ideally requiring no filtering. In the future, the possibility to set flight routes by setting way points above an image could be included. This would allow to test the ability of the particle filter on synthetic flights.

Figure 6.1: Exemplifying the reality gap. *Top left*: image patch generated using draug. *Top right*: image patch taken with the MAV's camera after printing the patch. *Below left*: texton image of the synthetic image. *Below right*: Texton image of the real image. The texton images shows that corresponding regions get classified into different textons, resulting in different histograms. This makes the transfer from the synthetic data to the real world difficult.

The shift of the processing power could be further amplified by using a different regression technique. In the current implementation using kNN regression, larger training data sets are penalized due to a greater prediction time. However, the choice of a different regression technique is not as straightforward as it might seem. The technique should be able to output multiple predictions, since certain map regions might be ambiguous.

The presented approach is a vision-only approach. This makes it robust to external disruptions such as magnetic fields and reduces the amount of points of failure. Additionally, the approach can be used on different devices, such as handheld cameras. Still, future developments could incorporate data from the inertial measurement unit (IMU) in the particle filter's motion model.

Additionally, the time complexity of the algorithm can be further reduced with the aim of running the algorithm on fly-sized MAVs. Depending on the target platform, parallelization or threading could be used on multi-core systems to simultaneously compute texton histograms, make predictions and run the particle filter.

Chapter 7

Conclusion

This thesis presented a novel approach for fast and accurate indoor localization of MAVs. We pursued an onboard design without the need of an additional ground station to foster flexibility and autonomy. The conducted on-ground and in-flight experiments underline the real-world applicability of the system. Promising results were obtained for waypoint navigation, accurate landing, and stable hovering in the indoor environment. The used approach is based on three pillars.
The first pillar shifts computational effort from the flight phase to an offline preprocessing step. This allows for using sophisticated algorithms, without affecting performance during flight. The second pillar states that during flight the MAV lightweight algorithms should run with low-performing processors. These algorithms should be able to trade off speed with accuracy. This allows to use them on a wide range of models, from pico-drones to MAVs with a wing-span of over one meter. Examples of these adaptable algorithms are the particle filter and the texton-based approach. The third pillar is the known—and possibly—modifiable environment. This knowledge and flexibility allows to predict the quality of the used approach. In contrast to SLAM frameworks, in which the task is to simultaneous mapping and localization, the proposed approach is intended for various repetitive indoor activities.

The developed algorithms set the stage for absolute localization in various GPS-denied environments, such as homes, offices, or factory buildings. While the used platform for this project was the Parrot Bebop Drone, the characteristics of the proposed system generalize to smaller MAVs in a flexible and innovative way. We hope that our indoor localization approach will pave the way for various applications, such as delivery, search and rescue, or surveillance to support human operators in everyday life.

Bibliography

[1] Markus Achtelik et al. “Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments”. *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE. 2011, pp. 3056–3063.

[2] Spencer Ahrens et al. “Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments”. *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE. 2009, pp. 2643–2648.

[3] Eman R AlBasiouny, Amany Sarhan, and T Medhat. “Mean-shift-FAST algorithm to handle motion-blur with tracking fiducial markers”. *Computer Engineering & Systems (ICCES), 2015 Tenth International Conference on*. IEEE. 2015, pp. 286–292.

[4] Adrien Angeli et al. “2D simultaneous localization and mapping for micro air vehicles”. *European Micro Aerial Vehicles (EMAV)*. 2006.

[5] Abraham Galton Bachrach. “Autonomous flight in unstructured and unknown indoor environments”. PhD thesis. Massachusetts Institute of Technology, 2009.

[6] Michael Blösch et al. “Vision based MAV navigation in unknown and unstructured environments”. *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2010, pp. 21–28.

[7] Haiyang Chao, Yu Gu, and Marcello Napolitano. “A survey of optical flow techniques for UAV navigation applications”. *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*. IEEE. 2013, pp. 710–716.

[8] Hung-Kuo Chu et al. “Halftone QR codes”. *ACM Transactions on Graphics (TOG)* 32.6 (2013), p. 217.

[9] G.C.H.E. De Croon et al. “Design, aerodynamics, and vision-based control of the DelFly”. *International Journal of Micro Air Vehicles* 1.2 (2009), pp. 71–97.

[10] GCHE De Croon et al. “Sub-sampling: Real-time vision for Micro Air Vehicles”. *Robotics and Autonomous Systems* 60.2 (2012), pp. 167–181.

[11] G.C.H.E De Croon et al. “The appearance variation cue for obstacle avoidance”. *IEEE Transactions on Robotics* 28.2 (2012), pp. 529–534.

[12] Hans Driessen and Yvo Boers. "MAP estimation in particle filter tracking". *2008 IET Seminar on Target Tracking and Data Fusion: Algorithms and Applications*. IET. 2008, pp. 41–45.
[13] Daniel Eberli et al. "Vision based position control for MAVs using one single circular landmark". *Journal of Intelligent & Robotic Systems* 61.1-4 (2011), pp. 495–512.
[14] Sean P Engelson and Drew V McDermott. "Error correction in mobile robot map learning". *IEEE International Conference on Robotics and Automation, 1992 Proceedings*. IEEE. 1992, pp. 2555–2560.
[15] Dieter Fox et al. "Monte carlo localization: Efficient position estimation for mobile robots". *AAAI/IAAI* 1999 (1999), pp. 343–349.
[16] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". *Pattern Recognition* 47.6 (2014), pp. 2280–2292.
[17] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion". *Pattern Recognition* 47.6 (2014), pp. 2280–2292. ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/j.patcog.2014.01.005. URL: http://www.sciencedirect.com/science/article/pii/S0031320314000235.
[18] Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. "Towards a navigation system for autonomous indoor flying". *IEEE International Conference on Robotics and Automation, 2009 (ICRA'09)*. IEEE. 2009, pp. 2878–2883.
[19] Eva Hornecker and Thomas Psik. "Using ARToolKit markers to build tangible prototypes and simulate other technologies". *IFIP Conference on Human-Computer Interaction*. Springer. 2005, pp. 30–42.
[20] Hirokazu Kato and Mark Billinghurst. "Marker tracking and HMD calibration for a video-based augmented reality conferencing system". *2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), 1999 Proceedings*. IEEE. 1999, pp. 85–94.
[21] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "Convolutional networks for real-time 6-DOF camera relocalization". *arXiv preprint arXiv:1505.07427* (2015).
[22] David G. Lowe. "Object recognition from local scale-invariant features". *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.
[23] Kimberly McGuire et al. "Local Histogram Matching for Efficient Optical Flow Computation Applied to Velocity Estimation on Pocket Drones". *arXiv preprint arXiv:1603.07644* (2016).
[24] Parrot. *CES 2015 : Parrot Bebop Dance Choreography*. 2015. URL: https://www.youtube.com/watch?v=A_3UifFb45Y (visited on 10/15/2015).
[25] Franck Ruffier et al. "Bio-inspired optical flow circuits for the visual guidance of micro air vehicles". *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*. Vol. 3. IEEE. 2003, pp. III–846.

[26] Stephen Se, David Lowe, and Jim Little. "Global localization using distinctive visual features". *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002.* Vol. 1. IEEE. 2002, pp. 226–231.

[27] Manik Varma and Andrew Zisserman. "A statistical approach to texture classification from single images". *International Journal of Computer Vision* 62.1-2 (2005), pp. 61–81.

[28] Manik Varma and Andrew Zisserman. "Texture classification: Are filter banks necessary?" *2003 IEEE Computer Society Sonference on Computer Vision and Pattern Recognition, 2003 Proceedings.* Vol. 2. IEEE. 2003, pp. II–691.