Machine Learning-based Indoor Localization for Micro Aerial Vehicles

Volker Strobel

August 17, 2016

Abstract

Widespread applications, ranging from surveillance to search and rescue operations, make Micro Air Vehicles (MAVs) versatile platforms. However, MAVs have limited processing power due to their small size and cannot fall back on standard localization techniques in the indoor environment. To address this issue, an efficient on-board localization technique using machine learning was developed in the scope of this thesis.

The vision-based 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 (k-NN) 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 positional ambiguities. To predict the performance of the proposed algorithm in different environments an evaluation technique is developed. 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, enabling comparisons between environments and the identification of critical positions within an environment. To compare maps before modifying an environment, a software tool was created that 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 promising results. In a triggered landing setting, the MAV correctly landed in specified areas. The map evaluation technique was applied to various high-resolution images to identify suitable maps.

The presented approach is based on three pillars: (i) a shift of processing power to a 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 can build a foundation for efficient localization in various GPS-denied environments.

Contents

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 efficient indoor localization for MAVs combining computer vision and machine learning techniques.

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 [24, 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 [7, 4, 2]. Additionally, many commercially available drones are already equipped with cameras. 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 not only lightweight but also robust to external influences like magnetic fields. This minimizes points of failure. However, this reduced physical payload is not without cost:

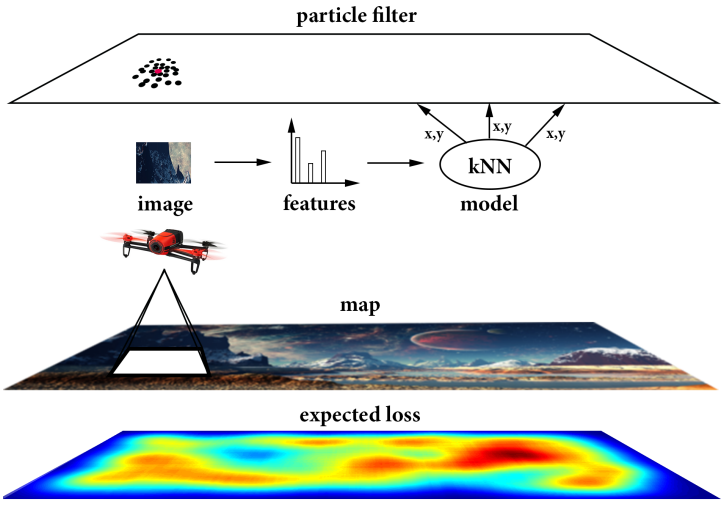

Figure 1.1: The figure illustrates the proposed system from a high-level perspective. A feature vector—the texton histogram—is extracted from the current camera image of the UAV. The feature vector is forwarded to a machine learning model that uses a k-Nearest Neighbors algorithm to output x, y-position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

it must be traded off against the higher computational payload for the on-board CPU. Vision-based position estimation is usually a time-consuming and memory-intense procedure. 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.

The developed framework uses a computationally efficient machine learning approach 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

in a pre-flight phase. This is then followed by finding a homography—a perspective transformation—between them to locate the current image in the map. As an alternative images can be aligned with high-precision position estimates from a motion tracking system.

In the next step, the complexity of these images is reduced by determining their histogram of textons—small characteristic image patches [40]. New images can then also be encoded by texton histograms and matched to images with known x, y-positions using the k-Nearest Neighbors (k-NN) algorithm. The k-NN estimates are passed to a particle filter to neatly aggregate the estimates and solve positional ambiguities. The computational effort of the approach can be adjusted by modifying the amount of extracted patches and used particles, resulting in a trade-off between accuracy and execution frequency. Figure 1.1 summarizes the algorithm.

In the presented approach, 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.

1.1 Problem Statement and Research Questions

The goal of this thesis is to develop a fast localization technique for MAVs. Therefore, we formulated the following problem statement:

> **Problem statement:** *How can x, y-coordinates be estimated in real-time and on-board of an MAV?*

It is assumed that the UAV flies at an approximately constant height, such that the estimation of height is not necessary. Since it is intended to fur-

ther reduce the size of MAVs, lightweight and scalable position estimation algorithms are needed. The problem was addressed by combining computer vision and machine learning techniques for achieving real-time position estimates. We focus on the following research questions (RQs):

- **RQ 1:** "*Can 2D positions be estimated in real-time using a machine learning approach on a limited processor in a modifiable indoor environment?*"

  Real-time position estimates can pave the way for autonomous flight of MAVs in various indoor environments; pursuing an "on-board design" to make the MAV independent of an external ground station is an important step for security and versatility.

- **RQ 2:** *"How can we predict and evaluate the suitability of a given map for the developed localization approach?"*

  Computer vision techniques are commonly limited to environments with sufficient and informative texture. If an environment can be evaluated before actually flying in it, the performance of the approach can be predicted and possible dangers prevented.

1.2 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 an autonomous system. In contrast to existing *active* approaches, the developed *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 k-NN algorithm in an integrative way and resolves position ambiguities. The method is a global localization 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 presented 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 autopilot system [8], 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.

1.3 Thesis 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 k-NN 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 and results of the on-ground and in-flight experiments. The results are then discussed in Chapter 5. Finally, we draw our conclusions and indicate future research directions in Chapter 6.

Chapter 2

Related Work

This chapter discusses advantages and disadvantages of different approaches for indoor localization. While a wide range of methods for indoor localization exists, from laser range scanners over depth cameras to radio-frequency identification tag (RFID) based localization, we only discuss methods that use the same technical and conceptual setup—localization with a monocular camera.

One distinguishes two types of robot localization: local techniques and global techniques [21]. Local techniques need an initial reference point and estimate coordinates based on the change in position over time. Once they lost track, the position can typically not be 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 [20].

Target systems and test environments are often too different to draw comparisons: factors, such as the size of the environment, the speed of the robot or camera, or the processor play crucial roles for the evaluation. Therefore, comparing the accuracy and run-time of different localization methods is difficult.

2.1 Vision-based Localization Methods

2.1.1 Fiducial Markers

Fiducial markers (Figure 2.1), which are often employed in augmented reality applications [27, 22], have been used for UAV localization and landing [19, 35]. The markers encode information in 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. Marker positions are then often further refined by removing improbable shapes, yielding an adjusted version of possible marker positions [23].

An advantage of fiducial markers is their widespread use, leading to technically mature open-source libraries, including ArUco [23] and ARToolKit [27]. Given adequate lighting conditions, markers can be used in a wide variety of environments [26]. 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 [26]. Another downside is that markers might be considered as visually unpleasant and may not fit into a product or environmental design [10]. They offer little flexibility, since one has to rely on predefined marker dictionaries. Additionally, marker-based approaches always require the modification of the environment. Like most vision-based approaches, the detection of markers is prone to changes in lighting conditions and may not work in low-contrast settings [26].

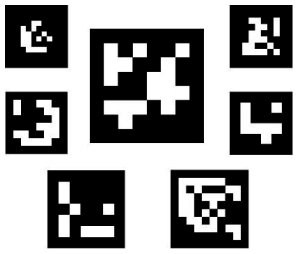

Figure 2.1: Examples of fiducial markers of the ArUco library.

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 [38], using algorithms such as Scale-invariant feature transform (Sift) [32], followed by finding a homography—a perspective transformation—between both keypoint sets (Figure 2.2). 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 using gray-scale images as input:

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 by removing maxima with small contrast and non-discriminative edges.
3. *Orientation assignment:* A histogram of the gradient orientations around the keypoint is created. The most frequent value indicates a the keypoint orientation.
4. *Keypoint description:* The local image gradients are transformed into a feature vector by describing pixels around a radius of a keypoint.

To locate the current view in the reference image, keypoints from one set are matched with their nearest neighbor in the other set using the Euclidean distance between their feature vectors. Based on the matched keypoint descriptions, a homography is calculated between the coordinates of both keypoint sets. This allow for locating the current view in the reference image. The calculation of the homography matrix (H) needs four matches between both keypoint sets. Usually many more points are available, leading to an overdetermined equation. The solution to H is then computed by minimizing the errors of between all the projected keypoints in a least-square sense.

While this *homography-based* approach is employed in frameworks for visual Simultaneous Localization and Mapping (SLAM), the pipeline of feature detection, description, matching, and pose estimation is computationally com-

plex [28]. Therefore, ground stations for off-board processing or larger processors are usually needed for flight control. In this thesis, the homography-based approach is used in a pre-flight phase to assign x, y-coordinates to images. The approach has been employed for global localization for UAVs:

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

Blösch et al. [7] evaluate 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. Subsequent research has brought the algorithm on board of UAVs [1], achieving a frequency of 10 Hz with a 1.6 GHz on-board processor with 1 GB RAM. However, the required processing power is still too complex for small MAVs.

2.1.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specialized machine learning method for image processing [31]. The supervised method has outperformed other approaches in many computer vision challenges [17]. CNNs consist of multiple neuron layers, which represent increasing levels of abstraction [31]. While their training is usually time-consuming, predictions with CNNs often takes only few milliseconds, shifting computational effort from the test phase to the training phase. CNNs have been used as a robust alternative for keypoint detection and description if images were perturbed [17] but needed more computation time than Sift.

In recent work, Kendall et al. present a framework for regressing camera positions based on CNNs [28]. The method achieves an accuracy of ap-

proximately 50 cm in indoor environments with a spatial extent between $2 \times 0.5 \times 1\,m^3$ and $4 \times 3 \times 1.5\,m^3$. The approach is rather robust to different lighting settings, motion blur, and varying camera intrinsics. The approach predicts positions on a modern desktop computer in short time. However, in our implementation—employing the scientific computing framework Torch [11]—the approach was still computationally too involved for achieving real-time prediction on an Odroid XU-4 single board computer.

2.1.4 Optical Flow

Optical flow algorithms are biologically inspired methods for navigation—taking inspiration from insects and birds [37]. They estimate the apparent motion between successive images, for example, by comparing the positions of their keypoints [9]. Optical flow methods 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.

Chao et al. [9] compare advantages and disadvantages of different optical flow algorithms for the use with UAV navigation. Most approaches are computationally rather complex [33]. To render on-board odometry feasible for small MAVs, McGuire et al. [33] 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 Methods

Textons are small characteristic image patches; their frequency in an image can be used as image feature vector. Varma et al. [40] originally introduced textons for classifying different textures, showing that they outperform computationally more complex algorithms, like Gabor filters [40]. 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. A texton histogram is obtained by extracting patches from an image and comparing them to all textons in a "texton dictionary". The frequency of the most similar texton is then incremented in the histogram.

Texton histograms are flexible image features and their extraction requires

little processing time, which makes them suitable for MAV on-board algorithms. The approach allows for adjusting the computational effort by modifying the amount of extracted image patches, resulting in a trade-off between accuracy and execution frequency. A disadvantage is that it discards all information about the spatial arrangement of texton, so that different images can have the same histogram.

De Croon et al. [14] use textons as image features to distinguish between three height classes of the MAV during flight. Using a nearest neighbor classifier, their approach achieves a height classification accuracy of approximately 78 % 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. [16] introduce the *appearance variation cue*, which is based on textons, for estimating the proximity to objects [16]. Using this method, the MAV achieves a high accuracy for collision detection and can avoid obstacles in a $5m \times 5m$ office space.

In the scope of this thesis, an efficient *global* localization was developed that draws upon the lightweight character of texton-based approaches and combines their flexibility with the advantages of homography-based approaches.

Chapter 3

Methods

This section describes the ideas behind the developed approach, the hardware, and software implementations. The approach is based on three "pillars": (i) a shift of processing power to a 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 to get the most out of the approach. The pseudo code in Algorithm 1 shows a high-level overview of the parts of the framework. Details about the parts of the framework and the pillars will be given in separate sections.

Algorithm 1 High-level texton framework

```
1: t ← 0
2: X_0 ← INIT_PARTICLES
3: while true do
4:     t ← t + 1
5:     I_t ← RECEIVE_IMG_FROM_CAMERA
6:     H_t ← GET_TEXTON_HISTOGRAM(I_t)
7:     z_t ← k-NN(H_t)
8:     X_t ← PARTICLE_FILTER(X_{t-1}, z_t)
9:     x_t, y_t ← MAXIMUM_A_POSTERIORI_ESTIMATE(X_t)
10: end
```

3.1 Hardware and Software

In our first approach, the commercially available Parrot AR.Drone2.0 was equipped with an Odroid XU-4 single board computer, a Logitech 525 HD webcam, and WiFi module. Figure 3.1 shows the setup. Instead of employing the AR.Drone2.0 processor, 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 can run 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 high-level language Python to the low-level language C without relying on many existing software libraries. However, this step removed the need for the additional payload and made the flight performance very stable. Also, it circumvented the effort of buying and attaching an external processor, which can be another point of failure. Another advantage 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.

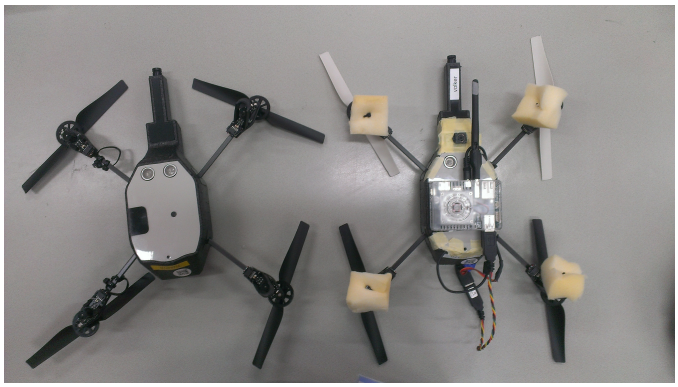

Figure 3.1: Comparison of an unmodified Parrot AR.Drone.2.0 (left) and a modified version (right). The modified one was equipped with an Odroid XU-4 single board computer, a Logitech C525 HD camera, a WiFi module, and a USB connection between the Odroid board and the AR.Drone.2.0 flight controller.

We decided to conduct all our tests with a quadcopter. Quadcopters allow for navigating in arbitrary directions without changing their yaw angle, show stable flight behavior, and often have high-resolution cameras. We used the *Parrot Bebop Drone* as a prototype. It is equipped with a lithium-ion polymer battery that lasts for approximately 11 minutes flying time. The UAV's dimensions are $28 \times 32 \times 3.6\,\text{cm}$ and it weighs $400\,\text{g}$. It has two cameras: a front camera and a downward-looking bottom camera. The developed 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 and runs a Linux operating system. The full specifications of the UAV can be found on its official website [36].

The original Bebop software development kit was replaced with the open-source autopilot software Paparazzi [8]. Paparazzi is used and advanced at the Micro Aerial Vehicle Laboratory at the TU Delft. The software provides a link between a ground station computer and the UAV to send commands and receive telemetry data. Furthermore, it provides functions for creating flight plans, plotting and logging telemetry data, and uploading firmware to the UAV. Its modular approach allows for combining functions regarding stabilization, localization, and control of UAVs, which are executed on board of the MAV. Paparazzi supports a wide range of commercially available aircrafts and associated hardware. Figure 3.2 shows the ground control station of Paparazzi.

The presented approach is implemented as a 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 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 to run them *on board*. A downlink connection—from the UAV to the ground station—permits monitoring the state of the aircraft, for example information about speed, altitude, position, or battery status.

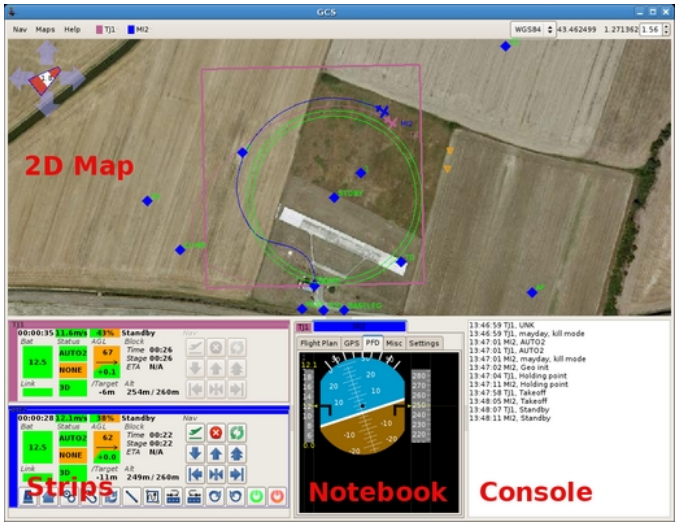

Figure 3.2: The ground control station of the Paparazzi software. It displays information about the status of the UAV and provides functions for controlling the vehicle (from PaparazziUAV wiki [42]).

3.2 Preliminary Dataset Generation

A main idea of the presented method is to shift computational effort to a pre-flight phase. Since the MAV will be used in a fixed environment, the results of these pre-calculations can be employed during the actual flight phase. Supervised machine learning methods need a training set to find a mapping from features to target values. In this first step, the goal is to label images with the physical x, y-position of the UAV at the time of taking the image. Therefore, a method for obtaining the physical position of the UAV is needed and GPS information is not available in the indoor environment. In the presented approach, the image is later converted to a texton histogram as described in the next section (Section 3.3).

One possible way to create the data set is to align the images with high-precision position estimates from a motion tracking system. The camera forwards 640×480 pixel images in Y'UV422 color space—a three-channel color space that encodes gray-scale information in the channel Y and color information in the channels U and V. The x, y-position is broadcast to the UAV via the ground station, which is connected to the motion tracking system. The data set is created by saving the image with the corresponding

position from the motion tracking system on the MAV's hard disk. The approach yields high-quality training sets since motion tracking systems can track rigid bodies at a high frequency within an error of few millimeters. Major disadvantages of the approach are that motion tracking systems are usually expensive and time-consuming to move to different environments. The workflow is illustrated in Figure 3.3.

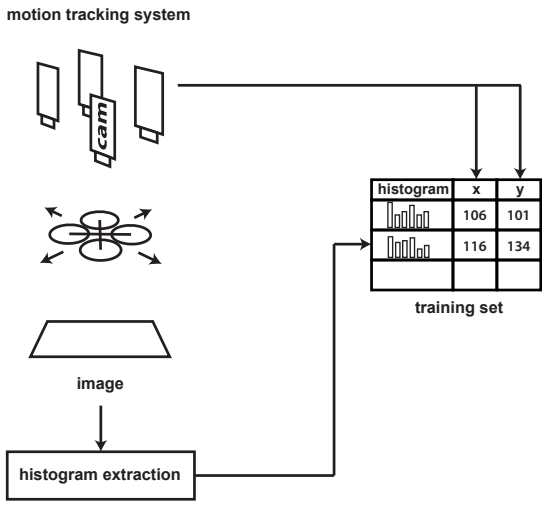

Figure 3.3: Training dataset generation if the motion tracking system is used. The texton histograms of the camera images during flight are extracted and aligned with the highly accurate position estimates of the motion tracking system. The result is a high-quality training set of texton histograms and corresponding x, y-positions.

As an alternative, we sought a low-budget and more flexible solution. Of the presented approaches in Chapter 2, the homography-based approach (Section 2.1.2) promises the highest flexibility with a good accuracy but also requires the most processing time. Since fast processing time is not relevant during the pre-flight phase, the approach is well-suited for the problem. The required image dataset can be obtained by using images gathered during manual flight or by recording images with a hand-held camera. To get a hyperspatial image of the scene for creating a map, the images from the dataset have to be stitched together. The stitched image has a higher resolution than the single images and contains a greater range of detail (Figure 3.5). With certain software packages the images can be "orthorectified"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 already be roughly aligned with the z-axis, given slow flight [7]. We used the freeware software Microsoft Image Composite Editor (ICE) [34] for the stitching process. However, this closed-source software does not publish details about its used techniques. As an open-source alternative, the panorama photo stitching software *hugin* [13] is available. In our tests, Microsoft ICE yielded better quality results.

Keypoints of the current image and the stitched map image are detected and described using the SIFT algorithm. The keypoint sets are further refined using Lowe's ratio test [32]. 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, its center 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 $C_x = \frac{width(W)}{width(I)}$ and $C_y = \frac{height(W)}{height(I)}$, where W is the real-world dimension and I the digital pixel image. Performing this step for all recorded images yields a preliminary dataset of images—that is later converted to a dataset of texton histograms—labeled with x, y coordinates. An illustration of the approach can be seen in Figure 3.4.

The stitching process can be time-consuming and error-prone. It can be impeded by distortions and perspective transformations of the recorded images. To circumvent the need for stitching together multiple images, an image with a high-resolution camera from a top view point can be taken that captures the entire area in some environments. 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. The homography-based process introduces noise into the dataset, since it only has a limited accuracy (Section 2.1.2) that depends on the quality of the keypoint matches.

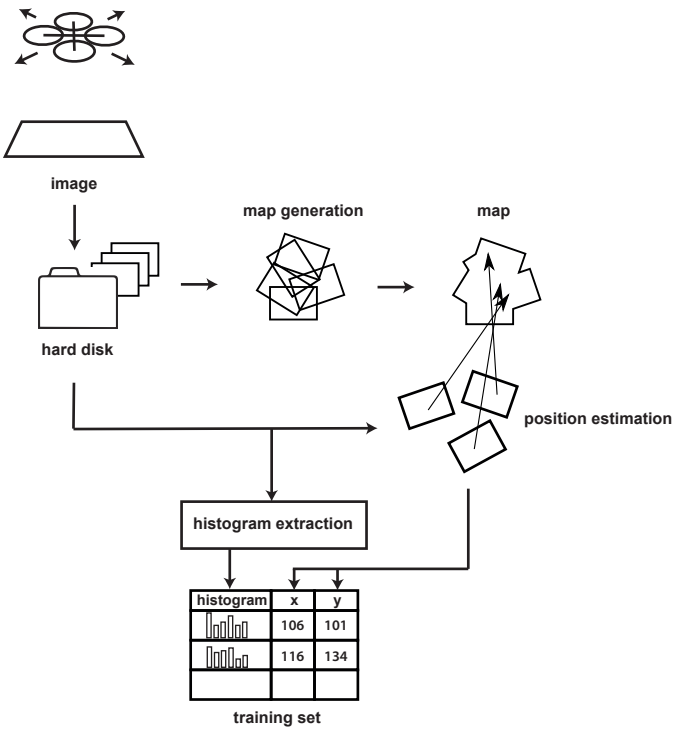

Figure 3.4: The figure illustrates the training set generation when applying the homography-based approach. Images from an initial flight are stitched together to create an orthomap. The same images are used to detect and describe their keypoints using SIFT, followed by finding a homography between the keypoints of the flight images and the orthomap to obtain x, y-coordinates per image. The training set is created by extracting texton histograms from the images.

3.3 Machine Learning-based Approach and Filtering

In this section, the core of the developed algorithm is described: the implementation of the texton framework, consisting of the texton dictionary generation, the extraction of the histograms, the k-Nearest Neighbors (k-NN) algorithm, and the particle filter. The dictionary of textons constitutes the base for determining the texton histograms. These histograms are used as features in the k-NN algorithm. The algorithm outputs k possible x, y-

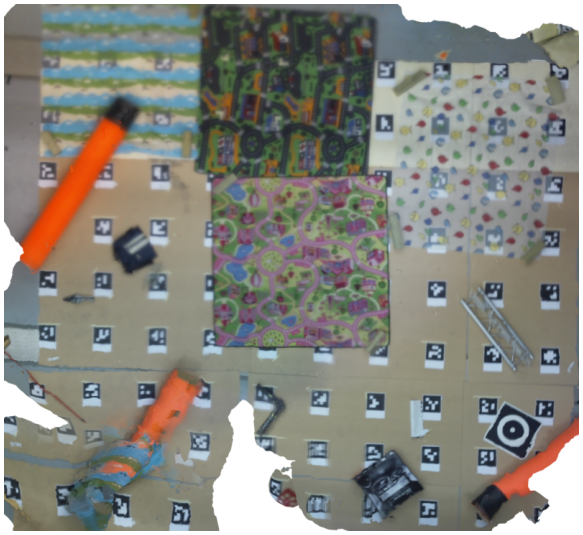

Figure 3.5: This figure shows the created orthomap of a texture-rich floor. It is stitched together using 100 single images and represents a real world area of approximately 8×8 meters. Image distortions, non-mapped areas, and slightly skewed seams at several points are visible.

coordinates for a given image, which are forwarded to the particle filter to yield a final position estimation.

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 [41]. The clustering was performed using a competitive learning scheme with a "winner-take-all strategy," a simple variant of a Kohonen network [29]. In the beginning, the dictionary is initialized with $n = 20$ random image patches from the first image, which form the first guess for cluster centers. Then, a new image patch x is extracted and compared to each texton d_j in the tentative dictionary using the Euclidean distance. The most similar texton d_r is the "winner." This texton is then adapted to be more similar to the current patch by calculating the difference in pixel values between the current image patch and the texton and

updating the texton with a learning rate of $\alpha = 0.02$:

$$d_r := d_r + \alpha(x - d_r) \tag{3.1}$$

The first 100 images of each dataset were used to generate the dictionary. From each image, 1000 randomly selected image patches of size $w \times h = 6 \times 6$ pixels were extracted, yielding $N = 100\,000$ image patches in total that were clustered. An example of a learned dictionary of grayscale textons can be found in Figure 3.6. For our approach, we also used the color channels U and V to obtain color textons.

Different maps and environmental settings require different texton dictionaries. 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 $n = 20$ for all maps, this parameter is also map-dependent and should ideally be adapted to the given map.

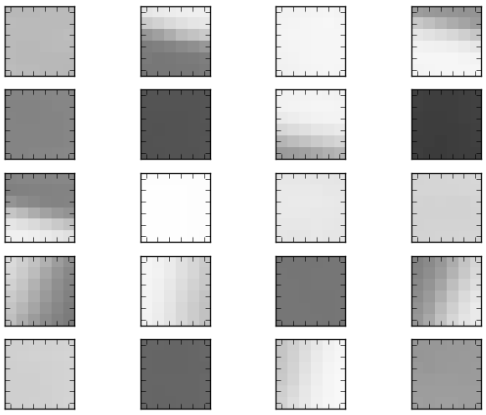

Figure 3.6: The figures shows a dictionary consisting of 20 grayscale textons ($w \times h = 6 \times 6$ pixels).

3.3.2 Histogram Extraction

The images from the preliminary dataset (Section 3.2) are converted to the final training set that consists of texton histograms and x, y-values. It is the

purpose of the conversion to obtain a more representative and dense description of an image, which should facilitate and speed-up recognition during the prediction step [25]. To extract histograms in the *full sampling* setting, a small window—or kernel—is convolved across the width and height of an image and patches are extracted from all positions. Each patch is compared with all textons in the dictionary and is labeled with the nearest match based on Euclidean distance comparing the pixels values in the channels Y, U, and V. The frequency of each label is reported in the corresponding "bin" of the texton histogram. The histogram is normalized by dividing the number of cases in each bin by the total number of extracted patches, to yield the relative frequency of each texton.

The convolution is a time-consuming step, since all possible combinations of width and height are considered: $(640 - w + 1) \cdot (480 - h + 1) = 301\,625$ samples are extracted. To speed up the time requirements of the histogram extraction step, the kernel can be applied only to randomly sampled image position instead [15]. This sampling step speeds up the creation of the histograms and permits a trade-off between speed and accuracy. One can see that the random sampling step introduces random effects into the approach. Therefore, for generating the training dataset, no random sampling was used to obtain high-quality feature vectors.

3.3.3 k-Nearest Neighbors (k-NN) algorithm

The k-Nearest Neighbors (k-NN) algorithm is the "machine learning-core" of the developed algorithm. Taking a texton histogram as input, the algorithm measures the distance 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 k-NN algorithm is one of the simplest machine learning algorithms, it offers several advantages [30]: it is non-parametric, allowing for the modeling of arbitrary distributions. Its capability to output multiple predictions enables neat integration with the developed particle filter. Its simplicity combines with transparency: it allows for spotting the possible sources of error such as wrongly labeled training examples. k-NN regression often outperforms more sophisticated algorithms [12]. A frequent point of criticism is its increasing computational complexity with an increasing size of the training dataset. While the used training datasets consisted of fewer

than 1000 images, resulting in short prediction times, time complexity can be reduced by storing and searching the training examples in an efficient manner, for example, with tree structures [6].

Chapter 4

Conclusion

This thesis presented a novel approach for lightweight indoor localization of MAVs. We pursued an on-board design to foster real-world use. The conducted experiments underline the applicability of the system. Promising results were obtained for real-time position estimates and accurate landing in the indoor environment.

The approach is based on three pillars that we identified for indoor localization for MAVs. The first pillar shifts computational effort from the flight phase to a preprocessing step. This provides the advantages of sophisticated algorithms, without affecting performance during flight. The second pillar states that on-board algorithms should be able to trade off speed with accuracy. This allows their use on a wide range of models. Examples of these adaptable algorithms are the texton-based approach and the particle filter. The third pillar is a known—and possibly—modifiable environment. This knowledge and flexibility allows for predicting and improving the quality of the approach.

The developed algorithms set the stage for global 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, including 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] Nitin Bhatia et al. "Survey of nearest neighbor techniques". *arXiv preprint arXiv:1007.0085* (2010).

[7] 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.

[8] Pascal Brisset et al. "The Paparazzi solution". *2nd US-European Competition and Workshop on Micro Air Vehicles 2006*. 2006.

[9] 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.

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

[11] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. “Torch7: A matlab-like environment for machine learning”. *BigLearn, NIPS Workshop*. EPFL-CONF-192376. 2011.

[12] *CPSC 340: Machine Learning and Data Mining*. URL: `https://www.cs.ubc.ca/~schmidtm/Courses/340-F15/L7.pdf` (visited on 08/10/2016).

[13] Pablo d’Angelo et al. *Hugin - Panorama photo stitcher*. URL: `http://hugin.sourceforge.net/` (visited on 08/10/2016).

[14] 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.

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

[16] 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.

[17] Alexey Dosovitskiy et al. “Discriminative unsupervised feature learning with convolutional neural networks”. *Advances in Neural Information Processing Systems*. 2014, pp. 766–774.

[18] 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.

[19] 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.

[20] 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.

[21] Dieter Fox et al. “Monte carlo localization: Efficient position estimation for mobile robots”. *AAAI/IAAI* 1999 (1999), pp. 343–349.

[22] S. Garrido-Jurado et al. “Automatic generation and detection of highly reliable fiducial markers under occlusion”. *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[23] 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.

[24] 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.

[25] Isabelle Guyon and André Elisseeff. "An introduction to feature extraction". *Feature extraction*. Springer, 2006, pp. 1–25.

[26] 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.

[27] 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.

[28] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "PoseNet: A convolutional network for real-time 6-DOF camera relocalization". *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2938–2946.

[29] Teuvo Kohonen. "The self-organizing map". *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[30] Mirosław Kordos, Marcin Blachnik, and Dawid Strzempa. "Do we need whatever more than k-NN?" *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2010, pp. 414–421.

[31] Yann LeCun et al. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[32] 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.

[33] 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).

[34] *Microsoft Image Composite Editor*. URL: http://research.microsoft.com/en-us/um/redmond/projects/ice/ (visited on 08/02/2016).

[35] Parrot. *CES 2015 : Parrot Bebop Dance Choreography*. 2015. URL: https://www.youtube.com/watch?v=A_3UifFb45Y (visited on 10/15/2015).

[36] *Parrot Bebop Drone*. URL: http://www.parrot.com/products/bebop-drone/ (visited on 07/14/2016).

[37] 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.

[38] 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.

[39] S. Thrun. *Artificial intelligence for robotics*. URL: `https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373` (visited on 03/10/2016).

[40] 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.

[41] 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.

[42] *Wiki PaparazziUAV*. URL: `https://wiki.paparazziuav.org/wiki/Main_Page` (visited on 07/14/2016).