Efficient Global Indoor Localization for Micro Aerial Vehicles

V. Strobel*[1], R. Meertens[2], and G.C.H.E. de Croon†[2]

[1]IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
[2]Micro Air Vehicle lab, Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands

Abstract

Indoor localization for autonomous micro aerial vehicles (MAVs) requires specific localization techniques, since GPS is usually not available. We present an onboard computer vision approach that estimates 2D positions of an MAV in real-time. The global localization system does not suffer from error accumulation over time and uses a k-Nearest Neighbors (k-NN) algorithm combined with a particle filter to predict positions based on textons—small image patches. The performance of the approach can be predicted by an evaluation technique that compares environments and identifies critical areas within them. In flight tests, the algorithm had a localization accuracy of approx. 0.6 m in a 5 m×5 m area at a runtime of 32 ms on board of an MAV. Its computational effort is scalable to different platforms, trading off speed and accuracy.

1 Introduction

Accurate onboard localization is a key challenge for micro aerial vehicles (MAV). In confined spaces, specific localization algorithms are essential, since the Global Positioning System (GPS) is usually not available. While light-weight MAVs could be employed in various indoor tasks, they cannot fall back on standard localization approaches due to their limited payload and processing power. To address this issue, this paper presents an efficient indoor localization technique (Figure 1). Our contribution is a machine learning-based indoor localization system that runs onboard of an MAV paving the way to an autonomous system. In the presented approach, computational power is 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 even be actively modified. The approach is based on the occurrence of textons, which are small characteristic image patches. With textons as image features and a k-Nearest Neighbors (k-NN) algorithm, we obtain 2D positions in real-time within a known indoor environment. A particle filter was developed that handles the estimates of the k-NN algorithm and resolves positional ambiguities. We consider settings in which the MAV moves at an approximately constant height, such that the estimation of height is not necessary. In contrast to existing approaches that use *active* sensors, the developed approach only uses a *passive* monocular downward-looking camera. While carrying active sensors, such as laser range finders, is too demanding for a light-weight MAV, onboard cameras can typically be attached. Additionally, we developed a technique for evaluating the suitability of a given environment for the presented algorithm. It identifies critical areas and assigns a global loss value to an environment. This allows for comparing different potential maps and identifying regions with low expected localization accuracy. The developed global localization system does not suffer from error accumulation over time. Onboard processing helps to reduce errors and delays introduced by wireless communication, and ensures a high versatility on the way to an autonomous system. We evaluated the the approach in flight experiments.

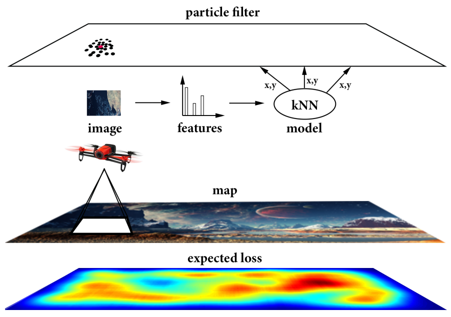

Figure 1: The figure illustrates the presented system from a high-level perspective. The feature vector—the texton histogram—that is extracted from the current camera image is forwarded to a machine learning model that uses a k-Nearest Neighbors algorithm to output k x, y-position estimates. These estimates 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 remainder of this paper is structured as follows. Sec-

*vstrobel@ulb.ac.be
†g.c.h.e.decroon@tudelft.nl

tion 2 surveys existing indoor localization approaches. In Section 3, the developed texton-based approach is presented and its components, the k-NN algorithm and the particle filter, are introduced. Section 4 describes the setup and results of the flight experiments. The results are discussed in Section 5 and we draw conclusions in Section 6.

2 Related Work

While a wide range of methods for indoor localization exists, we only consider methods in this section 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 [14]. 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 [13].

2.1 Optical Flow

Optical flow algorithms estimate the apparent motion between successive images. The most popular optical flow methods are gradient based approaches and keypoint-based methods [4]. Optical flow methods belong to the class of *local* localization techniques and most approaches are computationally rather complex [4].

2.2 Fiducial Markers

Fiducial markers have been used for UAV localization and landing [12, 21]. 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. An advantage of fiducial markers is their widespread use, leading to technically mature and open-source libraries. A drawback of the approach is that motion blur, which frequently occurs during flight, can hinder the detection of markers [1]. Furthermore, partial occlusion of the markers through objects or shadows break the detection. Another downside is that markers might be considered as visually unpleasant and may not fit into a product or environmental design [5].

2.3 Homography Determination & Keypoint Matching

A standard approach for estimating camera pose is detecting and describing keypoints of the current view and a reference image [22], using algorithms such as Scale-invariant feature transform (SIFT) [19], followed by finding a homography—a perspective transformation—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.

This *homography-based* approach is employed in frameworks for visual Simultaneous Localization and Mapping (SLAM) but the pipeline of feature detection, description, matching, and pose estimation is computationally complex [15]. While the approach has been employed for global localization for UAVs, the required processing power is still too high for small MAVs [7].

2.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a specialized machine learning method for image processing [18]. The supervised method has outperformed other approaches in many computer vision challenges [10]. 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 robust alternative for keypoint detection and description if images were perturbed [10] but needed more computation time than SIFT. In recent work, Kendall, Grimes, and Cipolla present a framework for regressing camera positions based on CNNs [15]. 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.

2.5 Texton-based Methods

Textons [24] are small characteristic image patches; their frequency in an image can be used as image feature vector. 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 [8].

De Croon et al. [7] use textons as 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. [9] introduce the *appearance variation cue*, which is based on textons, for estimating the proximity to objects [9]. Using this method, the MAV can avoid obstacles in a $5\,m \times 5\,m$ office space.

3 Methods

The pseudo code in Algorithm 1 shows a high-level overview of the parts of the framework. Details are given in the following subsections.

Algorithm 1 High-level texton framework

```
1: $t \leftarrow 0$
2: $\mathcal{X}_0 \leftarrow$ INIT_PARTICLES
3: while true do
4:     $t \leftarrow t + 1$
5:     $I_t \leftarrow$ RECEIVE_IMG_FROM_CAMERA
6:     $\mathcal{H}_t \leftarrow$ GET_TEXTON_HISTOGRAM($I_t$)
7:     $\mathbf{z}_t \leftarrow k$-NN($\mathcal{H}_t$)
8:     $\mathcal{X}_t \leftarrow$ PARTICLE_FILTER($\mathcal{X}_{t-1}, \mathbf{z}_t$)
9:     $x_t, y_t \leftarrow$ MAP_ESTIMATE($\mathcal{X}_t$)
10: end
```

3.1 Hardware and Software

We used the quadcopter *Parrot Bebop Drone* as a prototype for all our tests. The developed approach uses the bottom camera only, which has a resolution of 640 × 480 pixels with a frequency of 30 frames per second.

3.2 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.

One possible way to create the data set is to align the images with high-precision position estimates from a motion tracking system, which yields high-quality training sets. 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. Out of the presented approaches in Section 2, the homography-based approach (Section 2.3) 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. With certain software packages the images can be orthrectified 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 [3]. For the stitching process, we used the freeware software Microsoft Image Composite Editor (ICE) [20]. Keypoints of the current image and the map image are detected and described using the SIFT algorithm. This is followed by a matching process, that identifies corresponding keypoints between both images. 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.

3.3 Texton Dictionary Generation

For learning a suitable texton dictionary for an environment, image patches were clustered. The resulting cluster centers—the prototypes of the clustering result—are the textons [25]. The clustering was performed with a Kohonen network [16]. The first 100 images of each dataset were used to generate the dictionary. From each image, 1 000 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. For our approach, we also used the color channels U and V from the camera to obtain color textons.

3.4 Histogram Extraction

The images from the preliminary dataset are converted to the final training set that consists of texton histograms and x, y-values. 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. 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 [8]. This sampling step speeds up the creation of the histograms and permits a trade-off between speed and accuracy. 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.

3.5 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 Euclidean 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 [17]: 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. Additionally, k-NN regression often outperforms more sophisticated algorithms [6]. 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 (see also Figure 6), time complexity can be reduced by storing and searching the training examples in an efficient manner, for example, with tree structures [2].

3.6 Filtering

Our approach uses a filtering method that is able to capture *multimodal distributions*. Given an adequate measurement model, a general Bayesian filter can simultaneously maintain multiple possible locations and resolve the ambiguity as soon as one location can be favored. In this case, the predictions of the k neighbors can be directly fed into the filter without averaging them first. However, a general Bayesian filter is computationally intractable. Therefore, a variant based on random sampling was used: the particle filter. While its computational complexity is still high compared to a Kalman filter, one can modify the amount of particles to trade off speed and accuracy and adapt the computational payload to the used processor.

The weighted particles are a discrete approximation of the probability density function (*pdf*) of the state vector (x, y-position of the MAV). Estimating the filtered position of the MAV can be described as $p(X_t \mid Z_t)$, where X_t is the state vector at time t and $Z_t = \mathbf{z}_1, ..., \mathbf{z}_t$ are all outputs of the k-NN algorithm up to time t, with each $\mathbf{z}_i$ representing the k x, y-outputs of the algorithm at time i.

The used particle filter is initialized with particles at random x, y-positions. To incorporate the measurement noise for each of the k estimates from the k-NN algorithm, we developed a two-dimensional Gaussian Mixture Model (GMM) as measurement model. The GMM is parameterized by the variances $\Sigma^{[j]}, j \in \{1, \dots, k\}$ that are dependent on the rank j of the prediction of the k-NN algorithm (for example, $j = 2$ is the second nearest neighbor). The variance matrix $\Sigma^{[j]}$ specifies the variances of the deviations in x-direction and y-direction and the correlation ρ between the deviations. The values for $\Sigma^{[j]}$ were determined by calculating the variance-covariance matrix for the difference between the ground truth T from the motion tracking system and the predictions P_j of the k-NN algorithm: $\Sigma^{[j]} := \text{Var}(T - P_j)$.

The used *motion model* is solely based on Gaussian process noise and does not consider velocity estimates, headings, or control inputs. Its mean and variance are dependent on the expected velocity of the MAV. We used the forward difference $T_t - T_{t-1}$ to estimate the average movement and its variance-covariance matrix Σ_{process} between timesteps t and $t-1$.

In the following pseudo code of the developed particle filter (Algorithm 2), $\mathcal{X}$ is the list of particles, f the two-dimensional Gaussian probability density function, $z_t^{[i]}$ the ith neighbor from the kNN prediction, $x_t^{[m]}$ the mth particle at time t, and $w_t^{[m]}$ its corresponding weight. The "resampling wheel" [23] performs the importance resampling step.

With the GMM, the information of all k neighbors can be used, yielding a possibly multimodal distribution. While a multimodal distribution allows for keeping track of several possible positions, certain subsystems—for example a control loop—often need *one* point estimate. Using a weighted average of the particles would again introduce the problem that it could fall into a low density region (an unlikely position). Instead, we used a maximum a posteriori (MAP) estimate, as described by Driessen and Boers [11]. The estimation of *uncertainty* was modeled using the spread of the particles—as expressed by their variance in x-direction and y-direction.

Algorithm 2 Particle filter update

```
1:  procedure PARTICLE_FILTER($\mathcal{X}_{t-1}, z_t$)
2:      ▷ Initialize particle list
3:      $\mathcal{X}_{\text{temp}} := \varnothing$
4:      for $m = 1$ to $M$ do
5:          ▷ Add random process noise (motion model)
6:          $x_t^{[m]} \leftarrow x_t^{[m]} + \mathcal{N}(0, \Sigma_{\text{process}})$
7:          ▷ Iterate over $k$-NN preds (measurement model)
8:          $w \leftarrow 0$
9:          for $i = 1$ to $k$ do
10:             ▷ Gaussian Mixture Model
11:             $w \leftarrow w + f(z_t^{[i]}; x_t^{[m]}, \Sigma_{\text{measurement}}^{[i]})$
12:         $\mathcal{X}_{\text{temp}} := \mathcal{X}_{\text{temp}} \cup (x_t^{[m]}, w)$
13:     ▷ Importance resampling
14:     $\mathcal{X}_t \leftarrow$ RESAMPLING_WHEEL($\mathcal{X}_{\text{temp}}$)
15:     return $\mathcal{X}_t$
```

3.7 Map evaluation

The performance of the developed method depends on the environment: a texture-rich environment without repeating patterns will be better suited than a texture-poor environment. To assess if the algorithm will work in a given environment, we propose an evaluation scheme that compares different environments and areas within an environment. This scheme assigns a global fitness value or global loss value to a "map"—expressed as dataset $\mathcal{D}$ consisting of N texton histograms h_i and corresponding x, y-coordinates $\text{pos}_i = (x_i, y_i)$. The fitness value is intended to be proportional to the accuracy that can be expected when using this dataset as training set for the developed localization algorithm. The scheme allows for inspecting the dataset and detecting regions within the map that are responsible for the overall fitness value.

The idea behind the global loss function L is that histograms h_i and h_j in closeby areas should be similar and the similarity should decrease with increasing distance of the corresponding x, y-coordinates pos_i and pos_j. Therefore, the approach is based on the difference between *actual* and *ideal* texton histogram similarities in a dataset. The ideal texton similarity distribution is modeled as a two-dimensional Gaussian distribution around each x, y-position in the dataset. Using this idea, a histogram is compared to all others by comparing expected similarities to actual similarities. This results in a loss value per sample of the dataset (local loss). Applying the algorithm to each sample in the dataset yields the global loss of a dataset.

The method uses the cosine similarity $CS(h_i, h_j) = \frac{h_i^T h_j}{||h_i|| \, ||h_j||}$ to compare histograms. The cosine similarity has the convenient property that its values are bounded between -1 and 1. In the present case, since the elements of the histograms are non-negative, it is even bounded between 0 and 1. Let $f(x;\mu,\sigma) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ describe the non-normalized one-dimensional Gaussian probability density function. Since we assume that the ideal similarity in x-position is independent of the y-position, the ideal two-dimensional similarity function $d_e(\text{pos}_i, \text{pos}_j; \Sigma)$ can be modeled as the product of the respective one-dimensional function f:$d_e(\text{pos}_i, \text{pos}_j; \Sigma) = f(x_i; x_j, \sigma_x) \cdot f(y_i; y_j, \sigma_y$ This function is also bounded between 0 and 1, which makes the functions d_e and CS—ideal similarity and actual similarity—easily comparable. In summary, we propose the following global loss function (L) for evaluating a given dataset ($\mathcal{D}$): $L(\mathcal{D}) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} CS(h_i, h_j) - f(x_i; x_j, \sigma_x) \cdot f(y_i; y_j, \sigma_y)$ The simple difference—in contrast to least absolute deviations or least square errors—ensures that similarities that are *less* similar than the ideal similarity *reduce* the loss. Therefore, a high variation in texture is always seen as “positive”. The variances σ_x and σ_y specify the dimension of the region, where similar histograms are desired. The lower their value, the more focused the ideal similarity will be, requiring a high texture variety for getting a low loss value. A high value might overestimate the suitability of a dataset. While the approach is relatively robust to the choice of the parameter values, we still need to find a heuristic for suitable values.

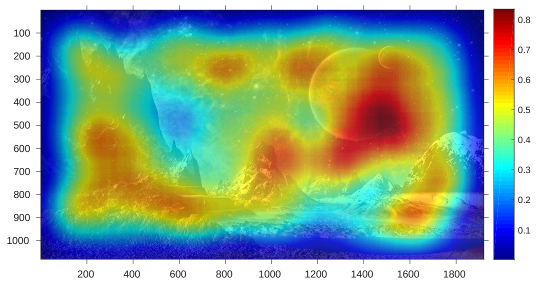

Figure 2: The figure shows the loss of a map: the regions that did not follow the ideal similarity pattern are displayed in red. For the visualization, the loss values per sample in the dataset were smoothed with a Gaussian filter. This assigns a loss value to each x, y-position of the map.

4 Analysis

In the experiments, the MAV was guided along flight plans using the motion tracking systen. If not otherwise stated, we used the following default values for the parameters in our framework:

parameter	value
samples in the histogram extraction step	400
textons in the dictionary	20
particles of the particle filter	50
histograms / images in the training set	800
histograms / images in the test set	415
neighbors in the k-NN algorithm	5

Map-dependent texton dictionaries were used and created by conducting an initial flight over the respective maps.

4.1 Baseline: Homography-based Approach

To find a baseline for our approach and to provide a homography-based training set, we used the homography-based approach to estimate x, y-coordinates in the same environment and based on the same images as the texton-based framework. The required hyperspatial image (Figure 3) of the environment was stitched together using 800 images and the software Microsoft ICE.

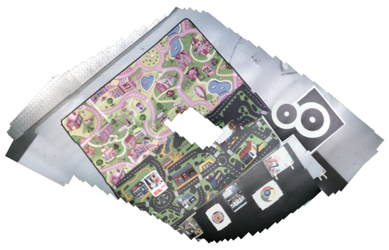

Figure 3: The created map (size: approximately 5×5 meters) that was stitched together using 800 images.

We estimated the x, y-coordinates of the 415 test images using the homography-based approach and compared the predictions to the ground truth. The predictions were not filtered. The results can be found in the following table.

	x-position	y-position
Error in cm	31	59
STD in cm	68	77

4.2 Training Set based on Motion Tracking System

In this experiment, the position estimates were calculated on board of the MAV using the texton-based approach with the particle filter. The Euclidean distances between the estimates of the motion tracking system and the texton-based approach were measured in x-direction and y-direction.

The training dataset was composed of 800 texton histograms with corresponding x, y-coordinates that were obtained from the motion tracking system. The images were recorded in a 5×5 meter area at a height of approximately one meter in a time span of one hour before the experiment to keep environmental factors roughly the same.

The results can be found in the following table. They are based on 415 images, which corresponds to a flight time of approximately 35 seconds.

	x-position	y-position
Error in cm	61	59
STD in cm	39	39

4.3 Training Set based on Homography-finding Method

In this experiment, the training dataset was created by estimating the x, y-positions of the 800 training images using the homography-finding method from the previous section and the same hyperspatial image. Apart from that, the settings are the same as in the previous experiment.

	x-position	y-position
Error in cm	54	97
STD in cm	41	61

4.4 Triggered Landing

For the triggered landing experiment, the MAV was guided along random flight paths, which covered a 5×5 meter area; during navigation, the MAV was programmed to land as soon as its position estimates were in a "landing zone": an x, y-position with a specified radius r. A safety criterion was introduced such that the landing is only performed if the standard deviations of the particles in x-direction and y-direction are below thresholds θ_x and θ_y. We set the parameters to $\theta_x = \theta_y = 60\,\text{cm}$. The x, y-coordinate of the circle was specified in the flight plan; the radius was set to $r = 60\,\text{cm}$. We performed six triggered landings; after each landing, the x, y-center of the zone was randomly set to another position in the map. For the texton framework, the same training set as in Experiment 4.2 was used.

Four out of six landings were correctly performed in the landing area. The distances of the two outliers were 14 cm and 18 cm, measured as distance to the circumference of the landing area.

4.5 Speed versus Accuracy Trade-Off

Adapting the frequency of the main loop of the developed approach to make it suitable for different platforms with varying processing power is one of its core parts. Figures 4 and 5 show the speed versus accuracy trade-off as a function of the used particles and of the used samples in the histogram extraction step, respectively. As a reference, the frequency using *full sampling* in the histogram extraction step was 0.1 Hz. The above stated *default* values were used for the *ceteris paribus* assumption, when varying the parameters. While the bottom camera of the Parrot Bebop Drone has a frequency of 30 Hz, the Paparazzi software currently only receives the images with a frequency of 12.5 Hz. Therefore, the baseline for the conducted experiment—the maximum achievable frequency—is 12.5 Hz. Figure 6 illustrates the frequency as a function of the used histograms in the k-NN algorithm. After having received the image, the processing time of the presented algorithm using the *default* parameter values is 32 ms, which includes the histogram extraction (16 ms) as well as the k-NN predictions, the filtering and the output of the best x, y-coordinate (16 ms).

5 Discussion

The flight tests show initial evidence for the real-world suitability of the method, which yields slightly less accurate results than the unfiltered homography-finding method. While we did not test the frequency of the homography-based approach on board of an MAV, on a desktop computer, it took 200 ms per image. Therefore, the developed algorithm runs at a much higher frequency. The triggered landing (Experiment 4.4) showed good accuracy: while most 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.

The experiments show that with an increasing accuracy of the approach, the frequency of the algorithm decreases. However, the errors reach a plateau after which no large improvements can be expected at the lower end of parameter ranges. By optimizing the parameters, one can obtain localization errors below 50 cm with the developed approach.

While we compared the settings of different parameters, there are no generally optimal parameters for the presented framework: setting the number of textons, the number of images patches, or the number of neighbors is dependent on the environment and the size of the training dataset. The parameters have to be adapted to the particular environment.

The accuracy of our global localization technique could be further improved by combining it with a local technique. To this end, odometry estimates using optical flow or the inclusion of data from the inertial measurement unit (IMU) could be suitable.

Our current implementation assumes constant height up to few centimeters and only small rotations of the MAV. While a quadroter can move in every direction without performing yaw movements, other MAVs or the use of the front camera for obstacle avoidance could require them. The inclusions of images of arbitrary yaw movements into the dataset would inflate its size to a great extent. This could lead to a deterioration of the accuracy and increase the time-complexity of the k-NN algorithm. Instead, a "derotation" of the camera image based on IMU data could be performed to align it with the underlying images of the dataset.

6 Conclusion

This paper presented an approach for lightweight indoor localization of MAVs. We pursued an onboard design to foster real-world use. The conducted experiments underline the applicability of the system. Promising results were obtained for position estimates and accurate landing in the indoor environment. An important step in the approach is to shift computational effort to a pre-flight phase. This provides the advantages of sophisticated algorithms, without affecting performance during flight. The approach can trade off speed with accuracy to use it on a wide range of models. The map evaluation technique allows for predicting and improving the quality of the approach.

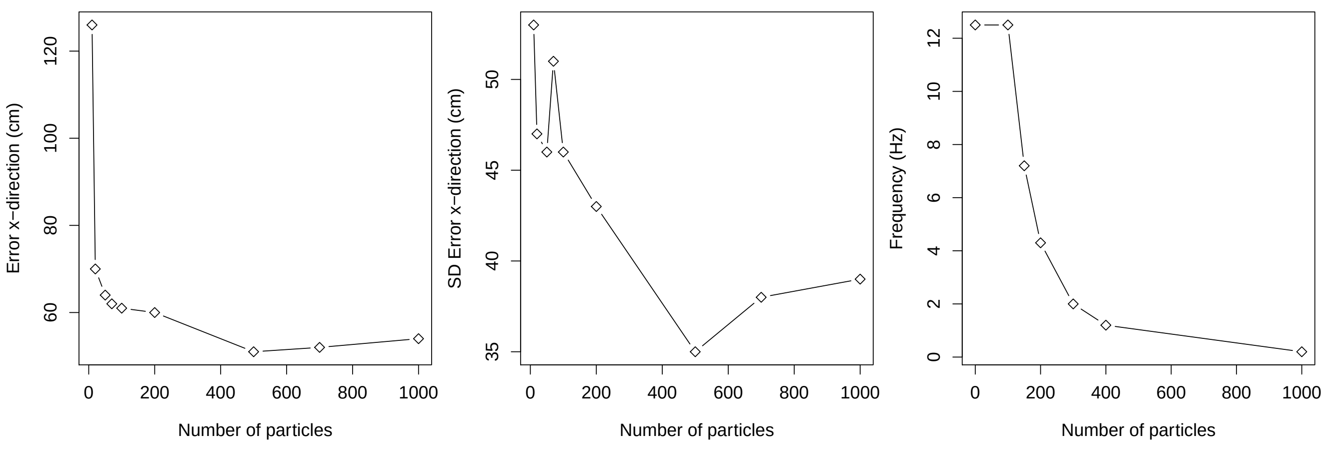

Figure 4: Speed versus accuracy trade-off in x-direction as a function of the number of used particles.

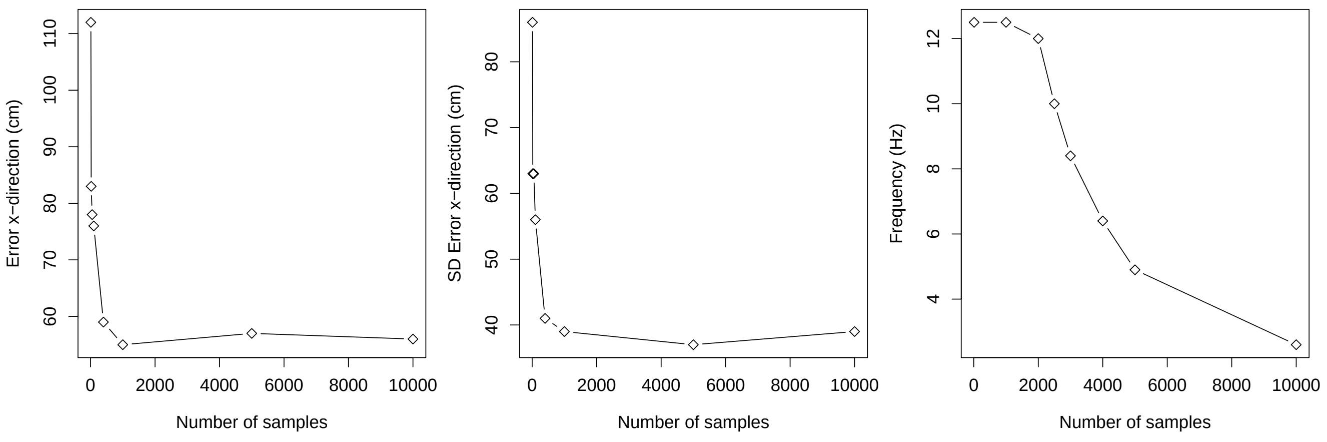

Figure 5: Speed versus accuracy trade-off in x-direction as a function of the number of used samples in the histogram extraction step.

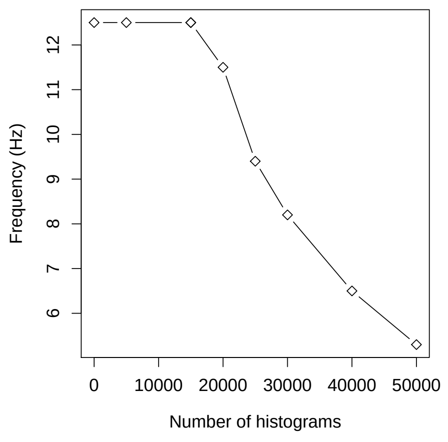

Figure 6: Frequency of the main loop as a function of the number of histograms in the training set.

REFERENCES

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

[2] Nitin Bhatia et al. "Survey of nearest neighbor techniques". In: *arXiv preprint arXiv:1007.0085* (2010).

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

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

[5] Hung-Kuo Chu et al. "Halftone QR codes". In: *ACM Transactions on Graphics (TOG)* 32.6 (2013), p. 217.

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

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

[8] G.C.H.E. De Croon et al. "Sub-sampling: Real-time vision for Micro Air Vehicles". In: *Robotics and Autonomous Systems* 60.2 (2012), pp. 167–181.

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

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

[11] Hans Driessen and Yvo Boers. "MAP estimation in particle filter tracking". In: *2008 IET Seminar on Target Tracking and Data Fusion: Algorithms and Applications*. IET. 2008, pp. 41–45.

[12] Daniel Eberli et al. "Vision based position control for MAVs using one single circular landmark". In: *Journal of Intelligent & Robotic Systems* 61.1-4 (2011), pp. 495–512.

[13] Sean P. Engelson and Drew V. McDermott. "Error correction in mobile robot map learning". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA 1992)*. IEEE. 1992, pp. 2555–2560.

[14] Dieter Fox et al. "Monte Carlo localization: Efficient position estimation for mobile robots". In: *Proc. National Conference on Artificial Intelligence/Annual Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI 1999)*. American Association for Artificial Intelligence, 1999, pp. 343–349.

[15] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "PoseNet: A convolutional network for real-time 6-DOF camera relocalization". In: *Proc. IEEE International Conference on Computer Vision (ICCV 2015)*. 2015, pp. 2938–2946.

[16] Teuvo Kohonen. "The self-organizing map". In: *Proc. IEEE* 78.9 (1990), pp. 1464–1480.

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

[18] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324.

[19] David G. Lowe. "Object recognition from local scale-invariant features". In: *Proc. IEEE International Conference on Computer Vision (ICCV 1999)*. Vol. 2. IEEE. 1999, pp. 1150–1157.

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

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

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

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

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

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