Random sampling for indoor flight

G.C.H.E. de Croon,[*] E. de Weerdt, C. de Wagter, B. Remes, and R. Ruijsink
Aerospace Software and Technologies Institute, Technical University of Delft,
Delft, the Netherlands

ABSTRACT

A challenging problem in the research field of Micro Air Vehicles is to achieve vision-based autonomous indoor flight. Approaches to this problem currently hardly make use of *image appearance features*, because these features generally are computationally expensive. In this article, we demonstrate that the broadly applicable strategy of random sampling can render the extraction of appearance features computationally efficient enough for use in autonomous flight. Random sampling is applied to a height control algorithm that estimates the height at which an image is taken by processing small image patches. The patches are extracted at random locations in the image. We vary the specific number of image patches to directly influence the trade-off between processing time and the accuracy of the height estimation. The algorithm is first tested on image sets and then on videos taken from a real platform. Subsequently, the algorithm is tested on a 15-gram ornithopter in an office room. The experiments show that very few image patches (0.56% of all possible patches) are already sufficient for the task of height control.

1 INTRODUCTION

Micro Aerial Vehicles (MAVs) hold a promise to observe places that are either too small or too dangerous for humans to enter. However, autonomous indoor flight remains a challenging and largely unresolved problem: even basic capabilities such as obstacle avoidance are hard to attain. The lightest of MAVs (see Fig. 1) cannot carry sensors such as a miniature laser range finder to achieve indoor flight [1, 2]. Instead, they can only carry a passive sensor such as a camera onboard. At the moment, there are two main approaches for achieving indoor flight on the basis of onboard camera images.

The first approach accurately estimates the state of the MAV (3D position and attitude). Such a state estimate can be obtained by 'matching' camera images to known locations in a 3D-model of the environment [3, 4, 5, 6, 7]. However, the algorithms for learning and using such a model are currently still computationally expensive. They require heavier, more energy consuming processors for onboard processing.

[*]Email addresses: g.c.h.e.decroon@tudelft.nl, microuav@gmail.com

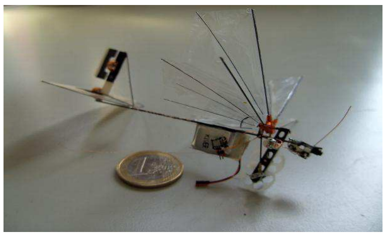
Figure 1: The *DelFly Micro* is a 3.07-gram ornithopter with a wing span of 10 cm. Autonomous flight will require fast image processing with little computational power available.

The second approach is a bio-inspired approach to autonomous flight that is computationally more efficient. Typically a state estimate is abandoned altogether and the MAV directly responds to incoming visual inputs [8, 9]. Generally, optic flow is used [10, 11, 12], since it has been shown to play an important role in insect flight [13, 14]. However, both the optics and the optic flow algorithms of flying robots are inferior to their natural counterparts. As a consequence, autonomous flight with optic flow is limited to environments with sufficient texture.

While optic flow is commonly used in efforts for reaching autonomous flight, image appearance has been largely neglected. Image appearance features could be useful for autonomous indoor flight, since they can capture information complementary to optic flow. For example, the absence of texture (a fail-case for optic flow) can be successfully detected by extracting appearance features. The little interest in appearance features is mainly due to the fact that their extraction is computationally expensive.

The **first contribution** of this article is to show that the broadly applicable strategy of *random sampling* can render the extraction of appearance features fast enough for use in indoor flight. Instead of extracting samples at all possible image locations, the strategy extracts samples at a random subset of locations. The higher computational efficiency comes at the cost of an acceptable loss in accuracy.

The **second contribution** of the article is that a novel height estimation algorithm is introduced for use in indoor flight. The algorithm captures the distribution of textures and / or colors in a still image, and classifies it as belonging to a certain height.

The remainder of the article is organised as follows. In Section 2, the height estimation algorithm is described. In Section 3, the computational efficiency and accuracy of random sampling is investigated in the context of the height estimation algorithm. Subsequently, the selection of the algorithm's parameters is explained in Section 4 and the offline tests we performed in Section 5. Then, we focus on the implementation of the algorithm for controlling a 15-gram ornithopter in Section 6. We explain the experimental setup, the controller, and evaluate the results of the algorithm in flight. Subsequently, we discuss the generalization of the proposed algorithm to unknown rooms in Section 7. Finally, we draw our conclusion in Section 8.

2 Height estimation algorithm

The height estimation algorithm processes the images from a forward pointing camera. The algorithm uses appearance differences between images taken at different heights, assuming little variance in the pitch of the MAV. Simply put: an image taken close to the ceiling is different from an image taken close to the floor. The algorithm is based on the work of [15]. They showed that the computationally efficient *texton* method performed better than computationally intensive filtering methods (such as Gabor filters) on a texture classification task. We use the texton method, since it has a rather good performance on image classification tasks and it is amenable to a random sampling approach.

2.1 Description

The texton method starts with the gathering of an image set that represents the type of space in which the MAV has to fly. The images in the set need to be taken at H different heights[1]. After constituting the image set, we commence the learning of a *dictionary* of n textons (also referred to as *visual words*). To this end, we extract small image samples of size $w \times h$ pixels from each image in the set. The extraction locations are drawn from a uniform distribution over the entire image. The extracted samples are clustered by means of a Kohonen network (cf. [16]). There are other - more advanced - clustering techniques (see [17] for a MATLAB©toolbox with most recent methods), but the Kohonen network has the advantage of learning the clusters in an iterative fashion. It finds clusters quickly, and can be stopped as soon as the clusters seem to cover the different samples sufficiently.

We can use the dictionary to represent an image as a histogram g of visual words. From the image, s image samples are extracted. For each sample, we determine which visual word i in the dictionary is closest (Euclidian distance), and increment the corresponding bin in the histogram $g(i)$. Normalisation of the histogram results in a maximum likelihood estimate $\hat{p}$ of the probability p of each word in the image: $\hat{p}(i) = g(i)/s$.

The so-formed probability distributions are used as feature vectors in the learning and classification of different heights. We divide the image set in subsets of images taken at the same height $h \in 1, 2, \ldots, H$. For all images in each subset, the probability distributions are estimated. Then, the average probability for each word $\overline{p_h(i)}$ and the corresponding standard deviation $\sigma(p_h(i))$ are calculated.

The classification of an image starts with the extraction of s image samples. This leads to the estimate $\hat{p}$, which can be compared with the learned $\overline{p_h}$. We made use of two very basic classification methods to classify $\hat{p}$: Naive Bayes (NB) and a variation of Nearest Neighbour (NN). We vary on the NN method, since it is normally slow at execution time. Instead of comparing a new point with all points in the training set, we only compare it with the H 'centroid' points $\overline{p_h}$ (the average of all points $\hat{p}$ observed for a height h during training). Algorithm 1 contains pseudocode for the classification.

Algorithm 1 Algorithm to classify an image as being of height class $h \in \{1, 2, \ldots, H\}$

```
empty the histogram $g$
for $i$ = 1 to $s$ do
    pick a random location $(x, y)$ in the image
    extract an image sample centered on the location
    for $j$ = 1 to $n$ do
        determine the Euclidian distance of the sample to word $j$
    end for
    select the closest word, $k$
    increment bin $g(k)$
end for
for $i$ = 1 to $n$ do
    $\hat{p}(i) \leftarrow g(i)/s$
end for
if NN then
    for $j$ = 1 to $H$ do
        determine the Euclidian distance of $\hat{p}$ to $\overline{p_j}$
    end for
    $h \leftarrow j$, for the $\overline{p_j}$ closest to $\hat{p}$
else if NB then
    for $j$ = 1 to $H$ do
        determine the probability of $\hat{p}$ being generated by $p_j$, according to independent Gaussian distributions for each visual word $k$, i.e., $\sim\mathcal{N}\left(\overline{p_j(k)}, \sigma(p_j(k))\right)$
    end for
    $h \leftarrow j$ of the $p_j$ leading to the largest probability
end if
```

3 Random sampling

If the height estimation algorithm evaluates all possible image patches for estimating p, it is too slow for application to indoor flight. In order to reduce the computational effort of the algorithm, random sampling can be employed: a limited number of samples is then extracted at uniformly distributed locations in the image. In this section, the effects of random sampling on the computational effort and on the accuracy of the estimate $\hat{p}$ are evaluated.

3.1 Computational effort

Given a classification method, the computational effort of algorithm 1 depends on the number of words n and the number of extracted samples s. The computational effort c is approximately:

$$c \approx snW + n + HnC = n(sW + 1 + HC) \qquad (1)$$

[1]In principle, height estimation could be approached as a regression problem. The choice for classification is due to practical advantages concerning the formation of training data.

, where W is the cost of calculating the Euclidian distance between two single bins, and C is the cost of comparing two single bins with the chosen classification method. During execution, n is fixed, but s can be varied freely.

Fig. 2 shows the mean processing times and corresponding standard deviations of a MATLAB implementation of the texton method on a data set of 94 gray-scale images. The MATLAB-implementation used to obtain these results is available online[2]. The number of samples s is varied from 50 to 2500 with steps of 50. For the experiments, $n = 30$ and $w = h = 5$. The processing times are measured on a 2.26 GHz laptop.

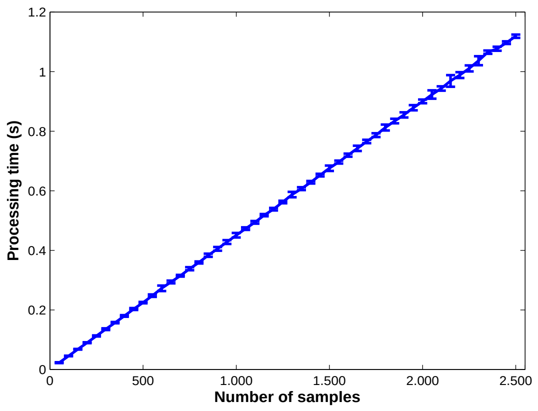

Figure 2: Mean processing times of a MATLAB-implementation of TMG for a number of samples s from 50 to 2500. The error bars indicate the standard deviation.

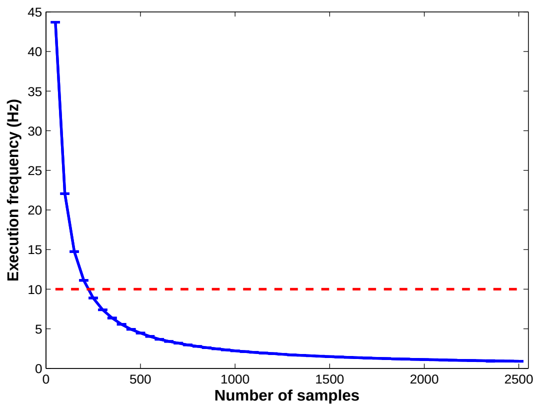

Figure 3: Mean execution frequencies and corresponding standard deviations of a MATLAB-implementation of TMG for a number of samples s from 50 to 2500 (blue solid line). The 10 Hz limit is shown by a red dashed line.

Fig. 2 shows that, as expected, the processing times increase linearly with the number of samples. Fig. 3 shows the execution frequencies corresponding to the different numbers of samples (blue line). It gives an idea of the numbers of samples that should be selected for the task at hand. For example, it is reasonable to state that for height control with indoor MAVs the execution frequency should at least be 10 Hz (red dashed line). The figure shows that this frequency is reached by extracting $\sim$225 or fewer samples. In addition, the execution frequency need not be higher than the frame rate, which is typically 30 Hz. Of the tested numbers of samples, only 50 samples resulted in an execution frequency higher than 30 Hz (43.7 Hz).

[2]http://www.bene-guido.eu/guido/

The full sampling processing time is not included in Fig. 2, since it falls ouside of its scope. The image size used for the experiments is 720×480, leading to 345,600 possible samples. The average processing time for full sampling in such relatively large images amounts to 165.78 seconds with a standard deviation of 1.68 seconds. This is $\sim$ 6376 times the processing time of the texton method with 50 samples. Of course, extracting a lower number of samples not only decreases the computational effort but also decreases the accuracy of the distribution estimates $\hat{p}$.

3.2 *Accuracy*

Here the effect of random sampling on the accuracy of the estimate $\hat{p}$ is analyzed in the context of the maximum likelihood estimate of the visual word distribution in the image. This distribution is a categorical distribution, and can be fully determined by extracting all samples from the image. Our analysis consists of determining the relation of the number of samples and the expected L1-distance between the maximum likelihood estimate and the actual distribution in the image.

We determine the probabilities for distances between the estimated and actual distributions for the case with replacement. Extracting a fixed number of s samples from random image locations results in word occurrences $g = \langle g_1, g_2, \ldots, g_n \rangle$. The vector g follows a multinomial distribution and it has the following well-kown probability formula:

$$P(g) = \frac{s!}{g_1!g_2!\ldots g_n!} p_1^{g_1} p_2^{g_2} \ldots p_n^{g_n}, \tag{2}$$

with $\sum g_1 + g_2 + \ldots + g_n = s$. For a given number of samples s, this formula allows one to iterate over all possible vectors g while determining the distance d between the estimated distribution $\hat{p} = g/s$ and the actual distribution p. The probability for distance $d = d(\hat{p}, p)$ can then be incremented by $P(g)$. Iterating over all g permits to calculate $P(d)$, the probability distribution of the distances between the estimated and actual distribution.

Although the above method is not elegant, it is tractable for a limited number of samples, since there are also a limited number of possible distances. To illustrate the effects of random sampling, we apply the method to the categorical distribution with $n = 6$: $p = \langle 0.5, 0.1, 0.1, 0.05, 0.05, 0.2 \rangle$. Fig. 4 shows the distribution $P(d)$ (y-axis) for $s = \{50, 100, 150, 200, 250, 300\}$ (x-axis), where illuminance represents high (white) to low (black) probabilities. The black line indicates the mean distance to the actual distribution and the white dashed line the 95th percentile.

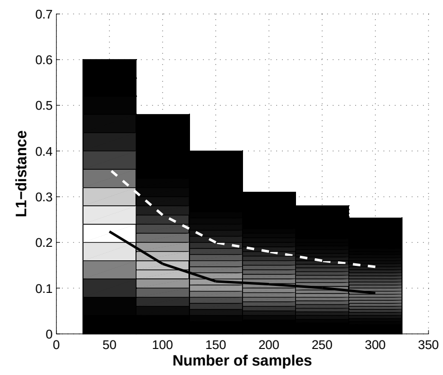

Figure 4: Relation between the number of samples used in random sampling and the accuracy of the estimate. The y-axis shows the L1-distance between the estimated and the true distribution, the x-axis shows the number of samples, $s = \{50, 100, 150, 200, 250, 300\}$. The intensity represents the probability, where white indicates high probability and black a low probability. The black line indicates the mean distance from the actual distribution, the white dashed line the 95th percentile of the distribution.

Two main observations can be made from Fig. 4. First, as to be expected, the mean of the distribution gets closer to the actual distribution as the number of samples increases. Second, this effect obeys the law of diminishing returns, so that *a modest number of samples already reduces the probability for 'large' distances considerably*.

When the distribution is known, $P(d)$ can be determined to provide certainty bounds to the distance between the estimated and actual distribution. For example, for the categorical distribution with replacement given above, $P(d < 0.20 \wedge s = 150) = 0.95$. The problem is of course that the actual distribution is not available. Still, we can limit the 95% certainty bound from above by assuming the worst case scenario, which occurs when the entropy of the actual distribution is highest[3]. Fig. 5 shows the mean distances and the 95th percentiles for different distributions. The distributions have $n = 6$ bins and different entropies, mentioned in the figure. The distances are highest for the distribution with maximal entropy, $H = 2.585$.

4 Parameter Selection

The texton method involves a number of choices, including the setting of parameter values that influence the trade-off between computational efficiency and accuracy of the probability estimates. We made these choices on the basis of experiments on a hand-made image set. We created the set by walking up and down our office corridor holding the camera at three different heights ($H = 3$): close to the ground ($\sim$15 cm), at waist level ($\sim$100 cm), and close to the ceiling

[3]A formal proof of this matter is beyond the scope of this paper.

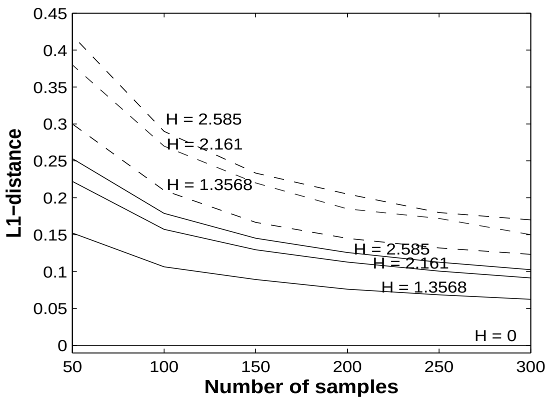

Figure 5: Relation between the entropy H and the mean L1-distances (solid lines) and the 95th percentile (dashed lines) for different numbers of samples. The highest entropy leads to the least accurate estimates.

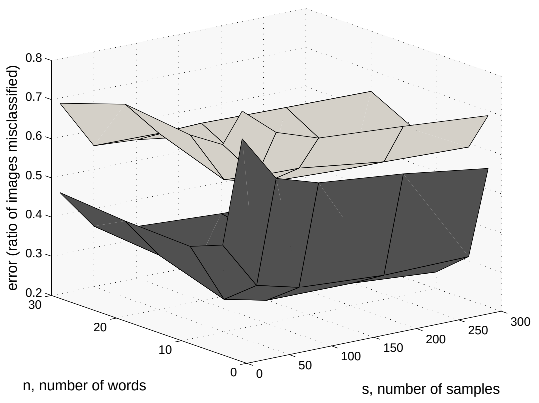

Figure 6: Mean error (expressed as the ratio of misclassified images) when using a separation in the image (dark grey), and no separation (light grey), for $s \in \{10, 50, 100, 200, 300\}$ and $n \in \{2, 5, 10, 20, 30\}$.

($\sim$250 cm). In order to get reliable results, we used a separate training set and test set (different walk with varying heights). Because of the random nature of the selection of the samples, we performed ten different training and test runs per parameter setting / choice. Since n and s are the most important parameters, we always tried out different values for these parameters.

One important choice is to divide the image into a bottom part and a top part. If an image is represented as a single probability distribution of visual words, all spatial information is lost. Coarse spatial information can be preserved by dividing the image in a bottom and a top part. The histogram and probability distribution then have double the size[4]. Figure 6 shows the difference in average error on the test set for the plain method (mean error shown in light grey) and the method with a bottom and top part (mean error shown in dark grey),

[4]The computational cost becomes: $c \approx sn^2W + 2n + 2HnC$

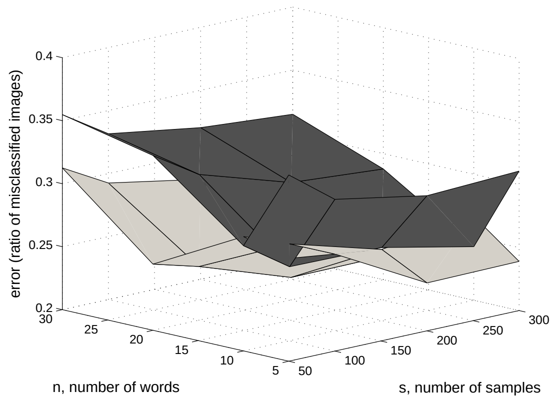

Figure 7: Mean error (expressed as the ratio of misclassified images) when using colour images (dark grey) and when using black-and-white images (light grey) for $s \in \{50, 100, 200, 300\}$ and $n \in \{5, 10, 20, 30\}$.

for $n \in \{2, 5, 10, 20, 30\}$ and $s \in \{10, 50, 100, 200, 300\}$. Dividing the image leads to a better average performance, at the cost of $n + HnC$ extra computational effort. Another interesting comparison is that between the use of colour images (dark grey mean) and black-and-white images (light grey mean), shown in Figure 7. Surprisingly, black-and-white images seem to lead to a slightly lower error than colour images for most configurations. This is most likely due to overfitting of the colour method.

Similar experiments led us to set the sample size to 5×5, the image size to 160×120, and to choose the nearest neighbour algorithm. Concerning the last choice, the naive Bayes classifier can lead to a better performance, but has a larger standard deviation in the results. The choice for the nearest neighbour algorithm is a choice for reliability.

Importantly, all of the experiments showed roughly the same relation between the number of samples and the error: increasing the number of samples only has a modest influence on the error above 100 samples. In addition, they all demonstrated that for the height estimation it is unnecessary to have more than 10 words. The final result of the experiments is an error of around $0.22 = 22\%$. In the next subsection, we investigate whether this error is low enough to estimate the height of a flying MAV.

5 OFF-LINE EXPERIMENTS

Further experiments were performed to determine how well the classification of the algorithm correlated with the height of a flying MAV. The differences with a hand-made video include more realistic image shake, WLAN-noise, and realistic light differences. In this experiment, we used a coaxial toy helicopter (a modified version of the *Lama V4*) equipped with a small onboard colour camera and transmitter.

We illustrate the experimental setup in Figure 8. The pilot

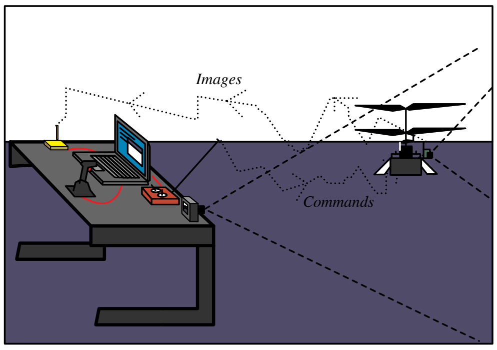

Figure 8: Setup of the experiment (see the text for details).

used the joystick to control the helicopter. The computer sent the joystick commands to an RC transmitter, which ported them through to the helicopter. The pilot flew the helicopter so that it attended different heights. During the flight, the helicopter sent its images to a laptop computer running our ground station software - *SmartUAV*. The software ran the texton method, with the settings as determined in the last subsection. The only difference was in the number of height levels, $H = 5$. An external camera filmed the helicopter. In this manner, we were able to get an impression of the height of the helicopter and compare it with the outcome of the texton method.

After the experiment, the videoframes of the external camera and the laptop computer were aligned. Then, straightforward motion detection was used to detect the helicopter in the external camera images, and its median y-location in the image was registered over time. Figure 9 shows the y-coordinate over time (red line), the height classification (green dotted line), and a smoothed version of the height classification (blue line). The figure shows that the uncertain and discrete classifications can be smoothed to give a 'continuous' signal, which correlates well with the y-coordinate of an external camera.

6 FLIGHT EXPERIMENT

After the success of the algorithm on the images of a flying platform, we tested the algorithm inside the control loop of a 15-gram ornithopter. In what follows, we explain the experimental setup (Subsection 6.1), the controller (Subsection 6.2), and we show the results of our flight test (Subsection 6.3)

6.1 *Experimental setup*

The experimental setup is the same as in Figure 8, but now we use the ornithopter as the flying platform. We perform the experiments in an office room, which underwent no other preparation than switching on the lights and pushing the furniture aside. For the online (in-the-loop) test, we film the ornithopter from two view points (see the map in Figure 11).

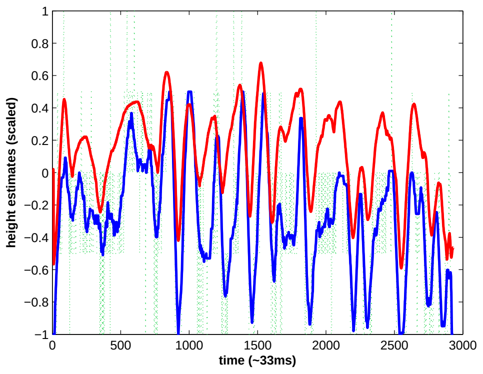

Figure 9: Comparison of the helicopter's y-coordinate in the external camera (solid line, scaled to $[-1, 1]$ with 1 being the top of the image), and the height estimated by the algorithm (grey line, scaled to $[-1, 1]$ with $h = 3$ mapped to 0). The black dashed line is a smoothed version of the grey line.

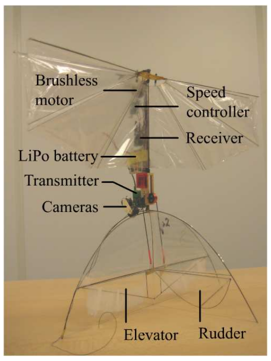

Figure 10: The 15-gram ornithopter used for the height control experiments.

In this way, we can reconstruct the 3D trajectory of the ornithopter during the test. The experiment starts by the pilot controlling the ornithopter via a joystick. He takes off, gives a trim to the throttle of the ornithopter, and then pushes the firebutton. When the firebutton is pressed, the computer is in full control of the throttle. During the height control experiment, the pilot continues to control the rudder of the ornithopter to avoid the walls. The elevator is constant throughout the experiment.

Figure 10 shows the ornithopter we have used for our experiments. It is a 15-gram ornithopter with two black-and-white cameras and transmitter on-board: only the forward pointing camera is used. Its X-tail allows it to perform vertical take-offs and landings. Powered by a LiPo battery, it can in principle perform a hovering flight for 15 minutes. We have to remark that the particular ornithopter used in the experiments of this article is part of a series produced in 2007.

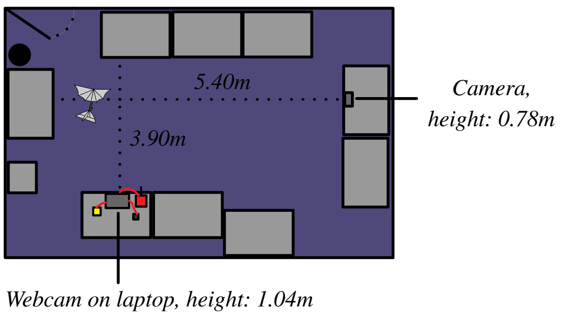

Figure 11: Map of the room used in the flight experiment. The pilot only controls the rudder of the ornithopter; the height is controlled autonomously. The ornithopter is filmed from two viewpoints: by the webcam of the laptop and the camera used in the previous experiment.

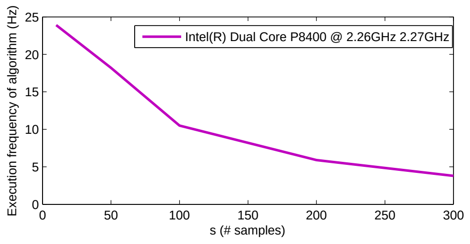

Figure 12: Frequency (in Hz) of the thread running the height estimation algorithm.

It was originally designed for 10 flying hours, but has already seen more than 200 flying hours; it is still able to hover, but only achieves a few minutes of hovering flight.

6.2 Controller

The controller used for the experiments is rather straightforward. The ground station receives an image, and uses Algorithm 1 to classify it as one of the heights $h \in \{1, 2, 3, 4, 5\}$. The height is scaled to the interval $[-1, 1]$, with height 3 mapped to 0. This height value is used as an error value in a regulator for the throttle. The throttle has a value in the interval $[-1, 1]$, with -1 being full throttle. Concerning the regulator: for the experiment with the ornithopter a P-controller was already sufficient to achieve height control, $P = 0.45$.

By running the controller and moving the ornithopter up and down (while being turned off), we could explore the trade-off between the number of samples s and the performance / computational effort. We noticed that the algorithm already performs sufficiently with only $s = 100$ samples, corresponding to the height algorithm running at ~10.5 Hz. To compare, with $s = 300$, the algorithm runs at ~3.8 Hz. Figure 12 shows the frequency of the algorithm on an Intel(R) Dual Core with 2.27 GHz for $n = 10$ and

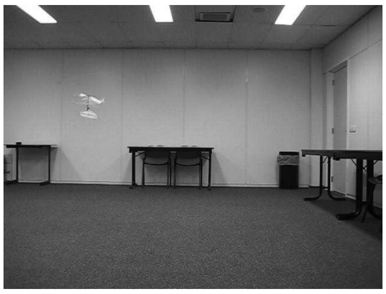

Figure 13: Picture of the height control experiment.

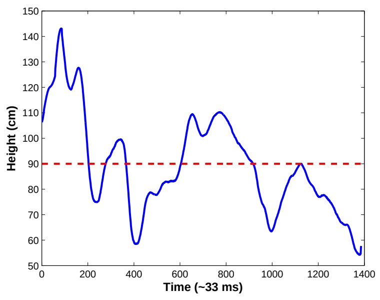

Figure 14: Solid blue line: height of the ornithopter over time. Dashed red line: approximative height of $h = 3$, towards which the height is regulated.

$s = 10, 50, 100, 200$, and 300. Note that this frequency was determined while also running the controller and video receiving / viewing / recording software. This is also the reason that the frequencies differ from those shown in Figure 3.

6.3 *Results*

The ornithopter successfully maintained a sane height during the experiment; it neither touched the floor nor got close to the ceiling (for a still of a video of the experiment, see Figure 13). To get more insight into the height during flight, we reconstructed the trajectory of the ornithopter as follows. First, we determined the median x- and y-coordinates of the motion detected in both (external) camera images[5]. Then, we used the knowledge of the experimental setup in determining the X,Y-coordinate at which the rays backprojected from the cameras intersect in the office room. Finally, we calculated the corresponding height Z.

Figure 14 shows the reconstructed height (Z) over time. The ornithopter stayed within an acceptable bandwidth around the height classified as $h = 3$ ($\sim$90 cm). Please note that the motion detection from the different view points leads to rather noisy estimates. As a consequence, the actual Z-coordinate may deviate from the shown (smoothed) trajectory in the order of $\sim$20 cm. The height approximately varies between 0.55m and 1.45m. The reader may have a look at the original experimental videos online at http://www.delfly.nl/.

[5]We omitted video frames in which the ornithopter was only visible to one of the cameras.

6.4 *Analysis*

The experiments showed that 100 samples sufficed for achieving height control. This represents only $\sim$0.56$\%$ of the total number of possible image samples. In this subsection, we perform a preliminary analysis of how many of the errors are actually caused by taking fewer samples. For this analysis, we need to differentiate between two types of classification errors: (1) if the actual distribution is misclassified, it is referred to as a generalization error, and (2) if the actual distribution would be classified correctly, random sampling can still lead to an estimated distribution that is classified differently - a sampling error. In the experiments, both types of errors were intermingled.

Here, we investigate the second type of error. Figure 15 contains the quantification of the sampling errors. For ten randomly selected images from the experiments, we isolated the sampling errors from the generalization errors as follows. First, each image was fully sampled and the resulting (actual) distribution classified as height h - which may differ from the actual height level at which the image was taken. Then, the image was classified multiple times with different numbers of samples. A classification was counted as a sampling error if it was not equal to h. The ten red thin lines show the relation between the sampling error and the number of samples for the ten images. The thick black line shows the same relation, but then for the case in which the texton distribution exactly corresponds to one of the learned distributions. Please note that random classification would result in an error of 0.8 (due to the 5 height levels). In addition, note that all misclassifications are confounded, while for height control a misclassification of $h = 1$ as $h = 2$ is less of a problem than a misclassification of $h = 1$ as $h = 5$. The main observation from Figure 15 is that a relatively low number of samples already considerably reduces the sampling error. After that, there are diminishing returns: increasing the number of samples from 50 to 100 has a larger performance impact than increasing from 100 to 150.

7 Discussion

The main goal of this article is to demonstrate how random sampling can render the extraction of appearance features fast enough for use in indoor flight. For platforms such as the DelFly II, such (fast) extraction of appearance features may turn out to be vital to autonomous flight. The reason for this is that more well-known methods such as optic flow encounter problems with the images that are significantly distorted and deteriorated by the high-frequency dynamics of the flapping movements. In this section, we discuss to what extent the presented height estimation algorithm generalizes to unknown rooms.

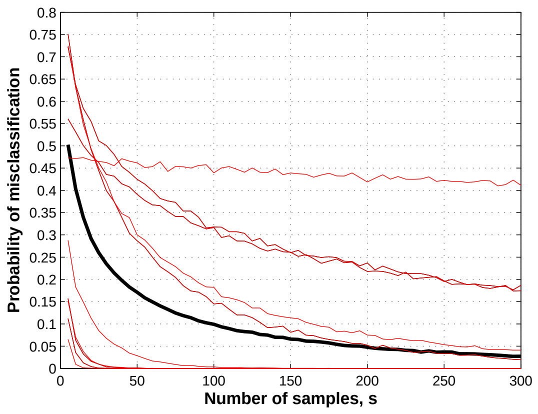

Figure 15: Misclassification due to random sampling, $H = 5$. Thick black line: average error when sampling from the learned distributions. Thin red lines: average errors for ten images from the height control experiment. See Subsection 6.4 for an explanation.

Specifically, we present the results of the texton method for height classification on a larger image set than the one discussed in Section 4. The set was created by walking around in different rooms and spaces while holding the camera at three different heights ($H = 3$): close to the ground ($\sim$15 cm), at waist level ($\sim$100 cm), and close to the ceiling ($\sim$250 cm). Fig. 16 shows nine example images, one for each different room. The figure illustrates that the set is quite challenging, since it is small and diverse. We performed k-fold tests, with $k = 9$ the number of rooms. As a consequence, the method was always trained on the images from eight rooms and then tested on the images from the ninth.

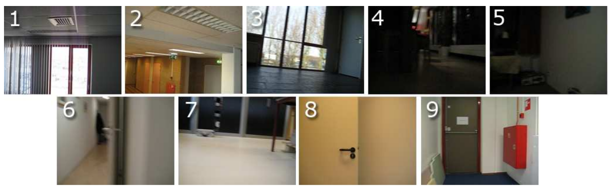

Figure 16: Example images from all nine rooms.

For this generalization task, the settings are slightly changed to $n = 30$, $s = 300$. All images are still transformed to gray-scale and divided in a bottom and a top part. Fig.17 shows the results of the experiments as nine confusion matrices. The row indices indicate the real classes (1 = low, 2 = medium, 3 = high), the column indices the classifications by the height estimation algorithm. Blue represents a low number of classifications for that class, red a high number of classifications. Perfect performance would be represented by a diagonal matrix with dark red on the diagonal and dark blue cells off the diagonal.

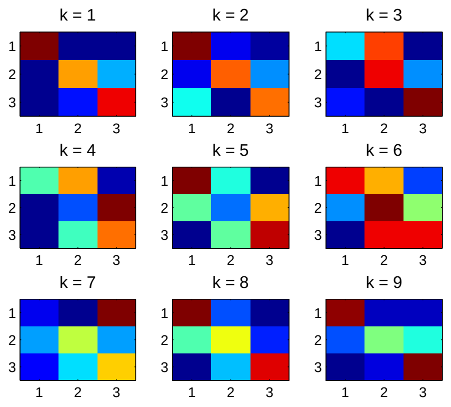

Figure 17: Confusion matrices for the texton method with $n = 30$ and $s = 300$ for nine different rooms.

In five out of nine test rooms, the texton method evaluates the true height levels as the most likely ones: rooms 1, 2, 6, 8, and 9. There are four rooms in which the method consistently misclassifies at least one level: rooms 3, 4, 5, and 7. One can see in Figure 16 that rooms 3, 4 and 5 have been filmed in very dim light. This obviously degrades the performance. The misclassifications in room 4 signify a shift up: the method classifies all levels as slightly higher than they are. In room 5, both the bottom and top level are well classified; only the medium level is interpreted as slightly higher. Such shifts in the height levels still have a chance of acceptable behaviour, since it may mean that the MAV just flies slightly higher in the room. In contrast, room 7 has many unacceptable misclassifications of the height, since the bottom level is interpreted as the highest level. In a control setting this would surely lead to a crash. The explanation for this unacceptable misclassification can be seen in Figure 16: uncommonly, this room has a white floor. With no white floor in the training set (rooms 1-6 and 8-9), the method misclassifies the corresponding images.

The results on the generalization experiment suggest that the proposed height estimation algorithm is especially suited for flight in rooms that have roughly the same appearance as the rooms in the training set. The features used in this article are robust to light intensity changes between different rooms or different days, but are sensitive to large intensity changes such as those between rooms with daylight and rooms with only artificial light.

8 Conclusion and future work

We conclude that appearance features can be extracted and processed fast enough for use in vision-based autonomous flight. The light-weight MAV *DelFly II* successfully used the height estimation algorithm to keep a sane height in an office room. To obtain successful height control, we did not prepare the room other than switching on the lights and putting the tables and chairs aside. The algorithm can be applied to almost any room, requiring only very limited training time ($< \sim 30$ minutes). The trained algorithm can generalize to rooms that have a similar appearance as the ones in the training set. However, rooms with a very different appearance (such as a white floor) lead to unacceptable misclassifications. Therefore, we conclude that the height estimation algorithm is most suited for known environments.

In future work, we will focus on the use of appearance features for obstacle avoidance. The most straightforward method to achieve obstacle avoidance is to provide a training set with example images close to or far from obstacles. A preliminary experiment leading to one minute of full autonomy of the mentioned ornithopter can be seen at: http://www.bene-guido.eu/guido/. In the preliminary experiment, a simple decision tree (cf. [18]) was used to classify images as close to or far from obstacles.

However, we will mainly focus on a more generalizable measure for obstacle avoidance. It turns out that the appearance variation of an image (as can be captured by the entropy of the texton distribution) is a good measure of obstacle incidence [19].

Acknowledgements

We thank Matthijs Amelink for helping us with the modification of the *Lama V4* toy helicopter.

References

[1] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *European Micro Air Vehicle conference and competititons, EMAV 2009, the Netherlands*, 2009.

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

[3] C. Kemp. *Visual Control of a Quad-Rotor Helicopter*. PhD thesis, Churchill College, University of Cambridge, 2006.

[4] A.J. Davison and D.W. Murray. Simultaneous localisation and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[5] S. Ahrens. Vision-based guidance and control of a hovering vehicle in unknown environments. Master's thesis, MIT, 2008.

[6] K. Celik, S.J. Chung, and A. Somani. Mono-vision corner slam for indoor navigation. In *IEEE International Conference on Electro/Information Technology (EIT 2008)*, pages 343–348, 2008.

[7] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision based mav navigation in unknown and unstructured environments. In *IEEE International Conference on Robotics and Automation*, 2010.

[8] A. Beyeler, J-C. Zufferey, and D. Floreano. 3d vision-based navigation for indoor microflyers. In *2007 IEEE International Conference on Robotics and Automation, Roma, Italy*, pages 1336–1341, 2007.

[9] S. Leven, J.-C. Zufferey, and D. Floreano. A minimalist control strategy for small uavs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, pages 2873–2878, 2009.

[10] F. Iida. Goal-directed navigation of an autonomous flying robot using biologically inspired cheap vision. In *Proceedings of the 32nd ISR (International Symposium on Robotics)*, 2001.

[11] F. Ruffier and N.H. Franceschini. Aerial robot piloted in steep relief by optic flow sensors. In *Intelligent Robotics and Systems (IROS 2008), Nice, France*, pages 1266–1273, 2008.

[12] A.M. Hyslop and J.S. Humbert. Autonomous navigation in three-dimensional urban environments using wide-field integration of optic flow. *AIAA Guidance, Control, and Dynamics*, pages 147–159, 2010.

[13] N. Franceschini, J.M. Pichon, C. Blanes, and J.M.Brady. From insect vision to robot vision. *Philosophical Transactions: Biological Sciences*, 337(1281):283–294, 1992.

[14] T.S. Collett. Insect vision: Controlling actions through optic flow. *Current Biology*, 12:615–617, 2002.

[15] M. Varma and A. Zisserman. Texture classification: are filter banks necessary? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume 2, pages 691–698, 2003.

[16] T. Kohonen. *Self-Organizing Maps, third edition, Springer Series in Information Sciences, Vol. 30*. Springer, 2001.

[17] L.J.P. van der Maaten. An introduction to dimensionality reduction using matlab, micc 07-07. Technical report, Maastricht University, the Netherlands, 2007.

[18] J.R. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[19] G.C.H.E. de Croon, E. de Weerdt, C. de Wagter, B.D.W. Remes, and R. Ruijsink. Real-time extraction of appearance features applied to robotic flight. In *Submitted.*