Optical flow for self-supervised learning of obstacle appearance

H.W. Ho*, C. De Wagter*, B.D.W Remes*, and G.C.H.E. de Croon*

***Abstract*— We introduce a novel setup of self-supervised learning (SSL), in which optical flow provides the supervised outputs. Optical flow requires significant movement for obstacle detection. The main advantage of the introduced method is that after learning, a robot can detect obstacles without moving - reducing the risk of collisions in narrow spaces. We investigate this novel setup of SSL in the context of a Micro Air Vehicle (MAV) that needs to select a suitable landing place. Initially, when the MAV flies over a potential landing area, the optical flow processing estimates a 'surface roughness' measure, capturing whether there are obstacles sticking out of the landing surface. This measure allows the MAV to select a safe landing place and then land with other optical flow measures such as the divergence. During flight, SSL takes place. For each image a texton distribution is extracted (capturing the visual appearance of the landing surface in sight), and mapped to the current roughness value by a linear regression function. We first demonstrate this principle to work with offline tests involving images captured on board an MAV, and then demonstrate the principle in flight. The experiments show that the MAV can land safely on the basis of optical flow. After learning it can also successfully select safe landing spots in hover. It is even shown that the appearance learning allows the pixel-wise segmentation of obstacles.**

I. INTRODUCTION

For many missions, Micro Air Vehicles (MAVs) will have to land autonomously. Therefore, MAVs need to identify a safe place which is defined in this study as a relatively flat surface with an allowable inclination and most importantly it is free of any obstacles underneath the vehicle. Many existing approaches use active sensors such as a laser range finder or use multiple cameras [1], [2], [3], [4], [5] to estimate the distance to many points on the landing surface. While they can provide accurate and redundant measurements, their perception range is limited and they are heavy and costly for small MAVs. Therefore, use of a monocular camera is preferable as it is light-weight and has low power consumption.

The current approaches using a single camera for autonomous landing are mainly visual Simultaneous Localization and Mapping (SLAM) and bio-inspired solutions using optical flow. The first method locates all the desired features in the field of view [6], [7] and determines the vehicle's location and 3D-structure of the landing surface at these points. Although its computational efficiency and accuracy have been improved over the years [8], [9], it still uses more computational resources than strictly necessary. The second method is inspired by flying insects, which heavily rely on optical flow for navigation. Biologists first found that honeybees perform a grazing landing by keeping the ventral flow (lateral velocities divided by height) constant [10], [11], [12], [13]. This approach guarantees a soft landing but does not control its vertical dynamics. To deal with that, recent studies proposed to use time-to-contact (height divided by vertical velocity) [14], [15], [16].

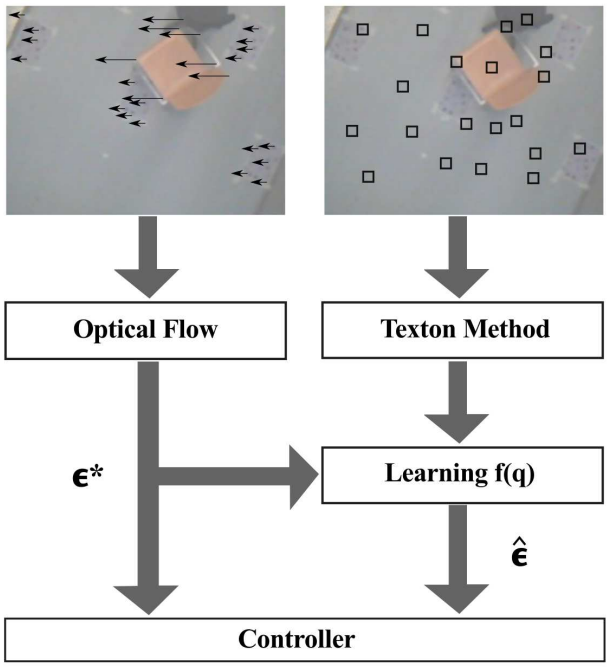


Fig. 1: Overview of the novel self-supervised learning (SSL) setup. The MAV starts flying using a roughness measure ϵ^* extracted from optical flow (*left*). A function is learned that maps appearance features from a *still* image to the roughness $\hat{\epsilon}$ (*right*). After learning, the MAV can determine whether there are obstacles below based on a still image, allowing landing site selection in hover.

Indeed, the bio-inspired approach using optical flow is currently a major research topic in autonomous landing of MAVs [17], [18], [19]. However, in order to sense the obstacles with optical flow during landing, the vehicle obviously needs to move. It would be both safer and more efficient to detect the obstacles underneath the vehicle in hover.

A possible solution for this is to learn the visual appearance of the obstacles, preferably without human supervision. *Self-supervised learning* (SSL) can be used to this end. Previous work focused on autonomous driving cars, where stereo vision, laser scanners, or bumpers provided supervised

*All authors are with the Micro Air Vehicle laboratory of the Faculty of Aerospace Engineering, Delft University of Technology, 2629HS Delft, The Netherlands h.w.ho@TUDelft.nl, G.C.H.E.deCroon@TUDelft.nl

outputs to learn the appearance of obstacles on the road [20], [21]. In [22], optical flow was used for tracing back the obstacles in time when they are far away. The supervised outputs were still provided by stereo vision and bumpers.

Here we propose using optical flow for the SSL of obstacle appearances to identify a safe landing place for MAVs. An overview of the proposed method is shown in Fig. 1. The major difference with previous work on SSL is that it is the first time *optical flow* is used for generating the supervised outputs. Additionally, it is applied to a flying vehicle that moves in 3-D space and thus has a completely different viewpoint than the cars in previous research[20], [21], [22]. This heavily reduces the prior on where obstacles can be located.

We will show that this novel setup of SSL permits an MAV to start flying with optical flow based behaviors (necessitating movement), but that after a while it can land completely based on the visual appearance of obstacles in still images (in hover). The remainder of the article is set up as follows: In Section II, we describe the proposed vision algorithm estimating surface roughness (ϵ^* and $\hat{\epsilon}$) to detect obstacles. Section III presents the results and discussion of both optical flow- and appearance- based landing experiments. Then, Section IV explains generalization of our SSL method to different environments. Finally, a conclusion with future works is drawn in Section V.

II. Self-supervised learning of obstacle appearance using optical flow

In this section, we explain (a) a computationally efficient method to estimate information from the optical flow field, which will allow the MAV to determine if a landing spot is safe, and (b) a SSL method of obstacle appearance using this information.

A. Surface roughness estimation from optical flow field

The vision algorithm we proposed to determine a safe landing spot is based on early findings in [23]. The algorithm was developed in previous research [24] to estimate the slope of the landing surface by assuming that (a) a pinhole camera model pointing downward is used, (b) the surface in sight is planar, and (c) the angular rates of the camera can be measured and used to de-rotate the optical flow. Under these assumptions, the equation of the optic flow vectors can be expressed as follows:

$$u = -\omega_x + (\omega_x a + \omega_z)x + \omega_x b y - a\omega_z x^2 - b\omega_z xy, \quad (1)$$

$$v = -\omega_y + \omega_y a x + (\omega_y b + \omega_z)y - b\omega_z y^2 - a\omega_z xy, \quad (2)$$

where u and v are the optical flow vectors in x and y image coordinates system, respectively shown in Fig. 2. $\omega_x = V_x/h$, $\omega_y = V_y/h$, and $\omega_z = V_z/h$ are the corresponding velocities in X, Y, and Z direction scaled with respect to the height h. Slope angles of the surface, α and β are the arctangent of a and b, respectively in the equations. In this work, the sparse corner detection method using FAST [25], [26] integrated with Lucas-Kanade tracker [27] is implemented to compute the optical flow. Note that since the computation of optical flow is not the primary focus of this paper but rather the concept of using it with SSL, other methods computing optical flow can also be used.

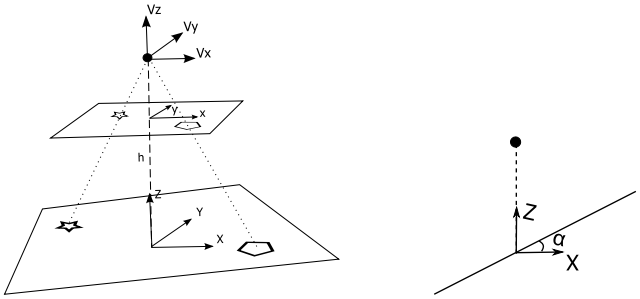


Fig. 2: *Left*: A pin hole model. *Right*: An inclined ground surface with slope α.

By re-writing (1) and (2) into matrix form as shown below, the parameter vectors $\mathbf{p_u} = [p_{u1}, p_{u2}, p_{u3}, p_{u4}, p_{u5}]$ and $\mathbf{p_v} = [p_{v1}, p_{v2}, p_{v3}, p_{v4}, p_{v5}]$ can be estimated using a maximal likelihood linear least squares estimate within a robust random sample consensus (RANSAC) estimation procedure [28]:

$$u = \mathbf{p_u}[1, x, y, x^2, xy]^T, \quad (3)$$

$$v = \mathbf{p_v}[1, x, y, y^2, xy]^T. \quad (4)$$

The estimated parameters provide important information for bio-inspired navigation: (a) ventral flow ($\omega_x = p_{u1}, \omega_y = p_{v1}$), (b) divergence ($D \triangleq \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = p_{u2} + p_{v3}$), and (c) time-to-contact ($\tau = 2/D$).

Lastly, the estimation using RANSAC returns the number of inliers and the fitting error. If there are obstacles sticking out of the landing surface, their optical flow vectors will not fit with the second assumption of a planar landing surface. This leads to a higher fitting error, ϵ^* which can thus be interpreted as a measure of *surface roughness*:

$$\epsilon^* = \epsilon_u + \epsilon_v, \quad (5)$$

with ϵ_u and ϵ_v sum of absolute errors of the RANSAC estimation in (3) and (4) divided by the number of tracked corners. Thus, we can use ϵ^* to detect obstacles near the ground surface by fitting the optical flow field. Note that (1) and (2) can be simplified by neglecting the second-order terms if the MAV only moves laterally. Therefore, linear fitting of the optical flow field can be used. To obtain a unique solution of either (3) or (4), we need at least three tracked features.

B. Self-supervised learning of obstacle appearance

Having to move close to obstacles in order to detect them represents a risk. It would be desirable to detect obstacles while hovering. To do this, the outputs from optical flow algorithm can be used as a supervised output for a self-supervised learning (SSL) method that maps visual appearance features to a roughness value. To illustrate the whole

process flow, an overview of the proposed SSL algorithm is presented in Fig. 1. Since the left part of the figure has been described in Section II-A, this section will discuss more about the right part of the figure.

1) Texton distributions for appearance representation: In this study the visual appearance is described using the *texton* method [29], based on the extraction of small image patches. With this method, first a *dictionary* is created consisting of *textons*, i.e., the cluster centroids of the small image patches. For our implementation, we follow our previous work in [30], and learn the dictionary with Kohonen clustering [31]. After creation of the dictionary, an image's appearance can be represented as a texton distribution. To this end, a number of image patches are randomly extracted from an image and per patch the closest texton can be added to a corresponding bin in a histogram. By normalizing it with the number of patches, a maximum likelihood estimate of the texton probability distribution is obtained. In previous research, the texton method has been used to calculate the appearance variation cue [32] and to learn how to recognize heights and obstacles [30] but it was never applied to SSL.

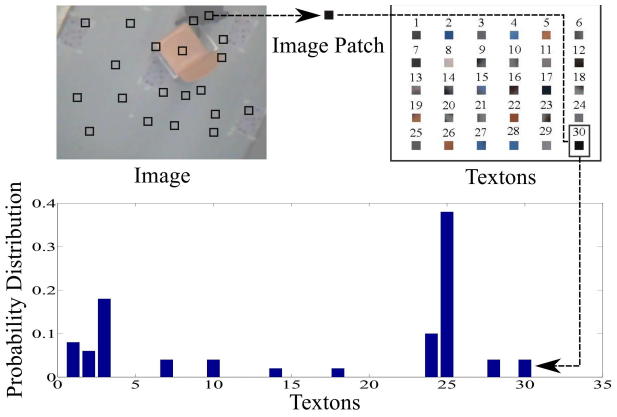


Fig. 3: Texton Method: A number of image patches is randomly selected from an image. The patches are then compared to textons in a dictionary to form a texton probability distribution of the image.

2) Predicting the roughness value with appearance: Secondly, we learn the texton distributions representing visual appearances of obstacle/ non-obstacle surfaces which can be differentiated by the roughness values from a set of on-board images. In order to do this, we can use a regression method to model a relationship between the texton distributions (regressors) and the roughness ϵ^* (regressand). To show feasibility and reliability of the relationship, various regression methods (such as Linear, Ridge, LASSO, Kernel smoother, Pseudo-inverse, Partial least squares, k-nearest neighbor regressions) were tried out to perform the learning using *prtools* [33] in MATLAB. A dataset consists of the texton distributions and the roughness ϵ^* are randomly sampled to get 80% training set and the rest for test set. Since the results from the regression methods are almost the same, we only plot the results of roughness $\hat{\epsilon}$ estimated using the Linear, k-nearest neighbor, and Pseudo-inverse regression methods compared with the ϵ^* in Fig. 4 while their normalized Root Mean Square Errors (RMSE divided by the range of the regressand) on the test set are $\sim 10\%$.

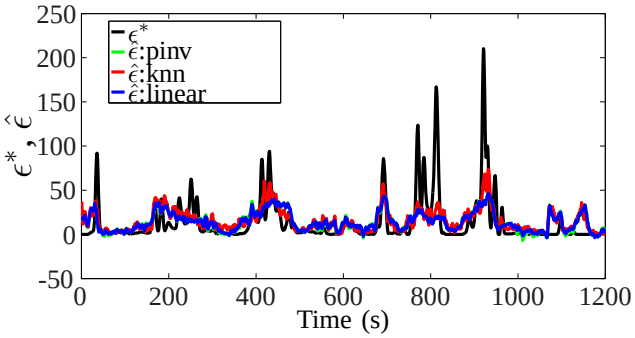


Fig. 4: Comparison of the roughness $\hat{\epsilon}$ estimated using linear, k-nearest neighbor (knn), and Pseudo-inverse (pinv) regression methods with the roughness ϵ^* estimated from optical flow algorithm.

Since they all give reasonably good results, the linear regression method is used for this study, due to its simplicity and computational efficiency. After obtaining the texton distributions $\mathbf{q}$ of m number of visual words and the roughness ϵ^* for n images, a linear regression model expressed in (6) can be trained.

$$f(q_i) = \rho_1 q_{i1} + ... + \rho_m q_{im} + \lambda, \quad i = 1, ..., n \qquad (6)$$

where λ is a bias and ρ are the regression coefficients, which are optimized so that $f(\mathbf{q}) \approx \epsilon^*$.

III. Results and Discussion

Before doing the experiment, a learning procedure needs to be performed: (a) training of a texton dictionary in the MAV by flying around the landing area, (b) collecting a dataset of texton distributions computed based on the trained dictionary and roughness ϵ^* using optical flow algorithm during the second flight, and (c) learning the regression model using the dataset[1].

A. Experimental platform and processing time

A Parrot AR.Drone 2.0 is used as a testing platform for this study. It uses 1GHz 32 bit ARM Cortex A8 processor and runs Linux operating system. It is equipped with a downward-looking camera running up to $60fps$ which is of particular interest to us for the landing purpose.

Instead of using the original Parrot AR.Drone program, an open-source autopilot software, Paparazzi Autopilot is used because it allows us to have direct access to the sensors and control the MAV. We created a computer vision module in Paparazzi Autopilot to capture and process images on-board the MAV and test our proposed algorithm in flight tests.

To examine the efficiency of the both algorithms, we measured the times taken by each process of the algorithms.

[1]Although we have chosen for this multi-phase learning procedure in this article, there is no fundamental reason prohibiting the different phases to be executed simultaneously during operation.

Since these times depend on the number of samples, we computed the average processing time required for each stage of both algorithms divided by number of samples used as shown in Table I. In our experiments, the maximum number of corners in optical flow algorithm and the number of samples used in SSL are both set to 25. Note that this value can be tuned to include more or less information from the images, however, higher value needs more computational time. The measured processing times (see Table I) show that both algorithms are computationally efficient and manage to be executed on-board the MAV.

TABLE I: Average processing time per sample for each stage of the vision algorithms

Optical Flow	Corner Detection	Corner Tracking	Flow Fitting	Sum
Time (ms)	1.2241	0.3794	0.4081	**2.0116**
SSL	Distribution Extraction			Sum
Time (ms)	0.8503			**0.8503**

B. Results of roughness estimate while navigating

We flew an MAV in the indoor flight arena with various obstacles and evaluated whether the obstacles can be localized based on the vision outputs. Fig. 5 presents the obstacles detection using the roughness on images taken from the on-board camera. This figure illustrates that both roughness estimates from optical flow (ϵ^*) and appearance ($\hat{\epsilon}$) are higher when there is an obstacle than when there is no obstacle on the landing surface. The result shows that the algorithm introduced in [24] and the proposed algorithm in this paper manage to detect the obstacles and thus identify safe spots to land.

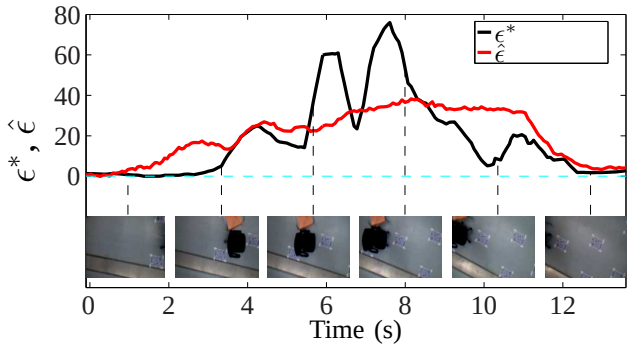


Fig. 5: Obstacles detection using vision output while navigating. *Left* axis indicates the values of the roughness ϵ^* from optical flow algorithm (*black line*) and $\hat{\epsilon}$ from SSL method (*red line*), respectively while the *bottom* part shows the on-board images with *black* and *yellow* obstacles.

C. Results of roughness estimate while hovering

To show the main advantage of the proposed SSL method, we hovered the MAV at several waypoints with obstacles and without obstacle on the ground and examined the roughness $\hat{\epsilon}$ and ϵ^*. Note that for the landing purpose we only took into account the presence of an obstacle in the center of the image. Therefore, we selected the image patches for building the texton distribution in a 160×120 pixel region in the image center only. Fig.6 shows a comparison between roughness ϵ^* from optical flow algorithm and $\hat{\epsilon}$ from SSL method while hovering. This figure shows that the SSL method is able to detect obstacles by giving higher value of $\hat{\epsilon}$ while the MAV is hovering above the obstacles. In contrast, the optical flow algorithm mostly gave faulty classifications due to the absence of lateral movement although there were some movements when approaching and leaving the waypoint which gave higher value of ϵ^* as shown in Fig.6. The results demonstrate that the proposed SSL method manages to detect obstacles even without having to move. [2]

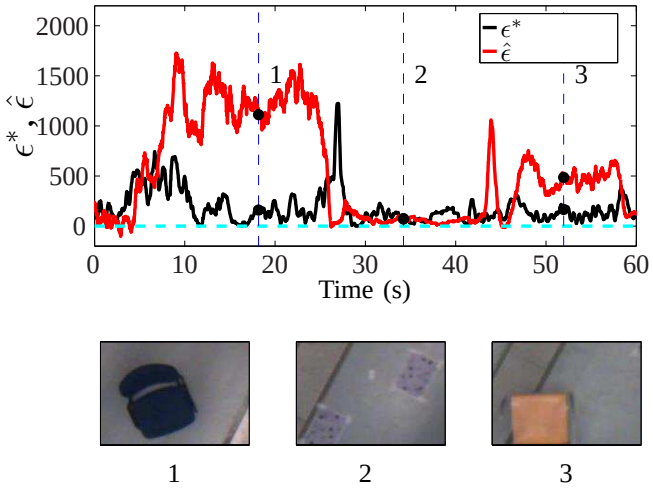


Fig. 6: Comparison of roughness estimates ϵ^* from optical flow algorithm (*black line*) and $\hat{\epsilon}$ from SSL method (*red line*) while hovering. Images at the bottom were taken from three different waypoints where the MAV hovered at the time indicated by 1, 2, and 3 during the experiments. Images 1 and 3 consist of *blue* and *yellow* chairs, respectively while image 2 does not have obstacle.

D. Results of roughness estimate during outdoor flight

Besides using 'standard' obstacles like chairs and artificial ground features in the indoor flight tests, we also investigated this proposed method in an outdoor environment. We flew the MAV in a place with trees (obstacles) and grass field (safe landing place). The roughness ϵ^* and $\hat{\epsilon}$ estimated from on-board images are shown in Fig. 7. Although the experiment was conducted in a windy condition, the obstacles can still be detected reasonably well for both methods.

E. Pixel-wise obstacle segmentation

The roughness value resulting from the optical flow processing is a *global* value for obstacle presence in the entire image. This study shows, after SSL, the MAV will not only be able to detect obstacle presence, but will even

[2]Experiment video: `https://goo.gl/Le5HT2`

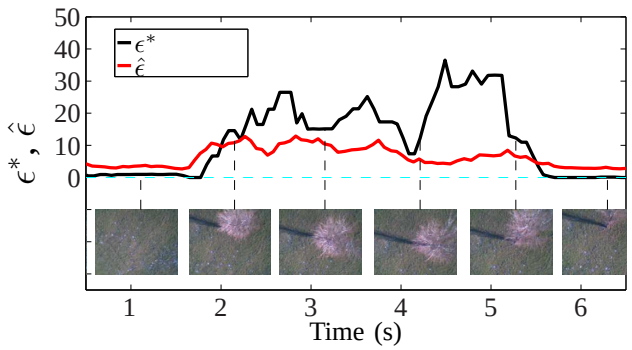


Fig. 7: Obstacles detection using vision output during outdoor flight. *Left* axis indicates the values of the roughness ϵ^* from optical flow algorithm (*black line*) and $\hat{\epsilon}$ from SSL method (*red line*), respectively while the *bottom* part shows the on-board images with trees and grass field.

be capable of pixel-wise obstacle segmentation. The basis for this capability is the *local* nature of the image patches involved in the construction of the texton distribution.

To show this, a sub-image with window size of 50×50 pixels of an image was moved across $x-$axis of the image for each line in $y-$axis with increment of 4 pixels until it covered the whole image. For each sub-image, the texton distribution was formed using 50 image patches and mapped to a roughness value with the regression function discussed above. Fig. 8 shows two still images (top) and the corresponding regression results. The obstacles clearly have a higher value in the regression maps (scaled to image size for viewing convenience). Note that this method with a moving window is used to show that our approach can also accurately segment the obstacles in an image. However it is computationally expensive since it processes almost every pixel in the image, and we actually do not need all the detailed information unless we need to land on a narrow place. Therefore, for the application on MAV, we used only one large window (160×120 pixels) located at the image center.

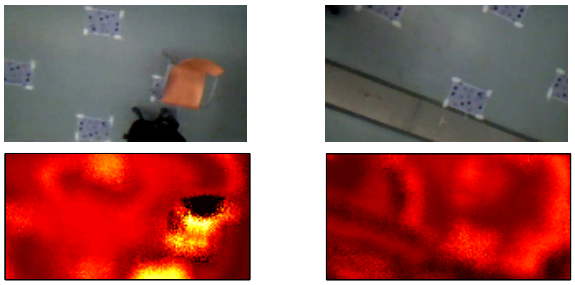


Fig. 8: Obstacle localization using roughness $\hat{\epsilon}$ from SSL method. This figure shows images with obstacles (*top left*) and without obstacle (*top right*) and roughness maps (*bottom*) for both images. The *light yellow* color in roughness map represents the presence of obstacles.

IV. Generalization of the SSL method to different environments

There is a main question remaining for the proposed method, i.e., how well the learned mapping from visual appearance to roughness will generalize to different environments. As in any learning scheme, this depends on the training and test distribution and on the learning method. In order to be successful, the training and test distribution should be sufficiently similar. In computer vision, this similarity does not only depend on the environment, but also on the visual features extracted from the images and how invariant they are to for instance rotation, scaling, and lighting changes.

In the context of SSL of obstacle appearance, we expect that features and learning methods can be found that generalize well over different environments and conditions. For instance, when we humans look at a Google maps image, we can discern obstacles such as trees and buildings rather well from areas that are more suitable for landing such as grass fields. Such a classification performance is also within reach of computer vision methods [34]. Of course, the computationally efficient texton distributions and simple learning methods used for our proof-of-principle are quite limited. However, even a limited generalization to a visually very different environment does not have to pose a problem in SSL. Two strategies are available to deal with this: (1) continuously learn the mapping when the MAV is moving enough with respect to the visual scene, and (2) detecting when the learned mapping is receiving different inputs and hence producing uncertain outputs. If the estimated outputs are uncertain, the MAV can rely again on optical flow and adapt its mapping to the new environment. In this section, we show that the uncertainty of outputs in a visually different environment can be evaluated with a Naive Bayes classifier and Shannon entropy.

A. Naive Bayes Classifier

Given a distribution of n textons ($\mathbf{q} = (q_1, \ldots, q_n)$) to be classified, the Naive Bayes classifier for k possible classes is given as in (7). In this study, we have two classes ($k = 2$), i.e. presence and absence of an obstacle, each of which can be represented by a distribution of $n = 30$ textons.

$$p(C_k|q_1, \ldots, q_n) \propto p(C_k) \prod_{i=1}^{n} p(q_i|C_k) \qquad (7)$$

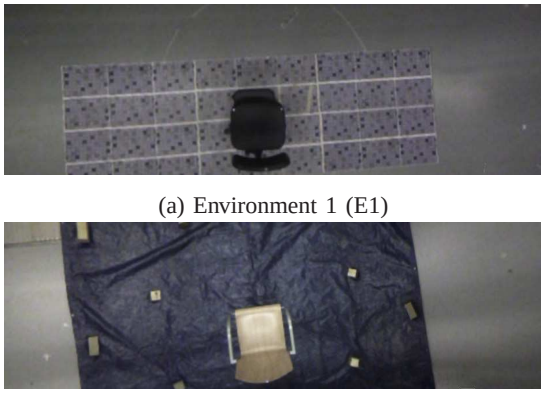
To create a Naive Bayes classifier, we first assign a roughness threshold, $\hat{\epsilon}_{th}$ ($= 20$) to classify the distributions based on its corresponding roughness, $\hat{\epsilon}$ into two classes labeled C_1 for obstacle (if $\hat{\epsilon} > \hat{\epsilon}_{th}$) or C_2 for non-obstacle (if $\hat{\epsilon} < \hat{\epsilon}_{th}$). Based on this dataset, learning of the Naive Bayes classifier was performed using *prtools* [33].

B. Analysis on Two Different Environments

The MAV was flown in two visually different environments (E1 and E2) in which one different obstacle was placed as shown in Fig. 9. By following the procedure in Section III, a regression model was trained in E1 and then the texton distributions were logged for E1 and E2 by flying

the drone over the obstacles repeatedly. Here, we investigate how well the regression model trained in E1 performs in E2. Please note that despite the use of a similar object (a chair), the environments are visually very different (the chair being dark in E1 and bright in E2, the surface being grey in E1 and dark blue in E2). Fig. 10 plots the roughness with ground truth position of the obstacle in E2. In this figure, the ground truth positions of the obstacle are represented by the square boxes on the top of the figure where black areas indicate full visibility of the obstacle in the field of view of the camera while half visibility is shown using gray areas. It is a rather surprising observation that the learned model actually remains quite effective even when flying in a different environment. Although based on this result there is obviously some generalization, it would be better to re-train the model to adapt the new environment.

80% of the distributions in the E1 are used to train the Naive



(a) Environment 1 (E1)

(b) Environment 2 (E2)

Fig. 9: Images of two different environments stitched using on-board images

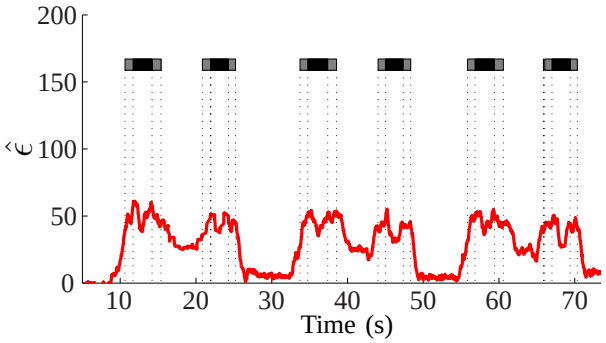


Fig. 10: Roughness estimate $\hat{\epsilon}$ in E2 using the regression model trained in E1 and ground truth position of the obstacle indicated by the square boxes (*top*) in which the *black* and *gray* areas represent full and half visibility of the obstacle, respectively.

Bayes classifier and the rest of the distributions are used for testing purpose. Both test sets from E1 and E2 are tested on the Naive Bayes classifier and the classification error, Err are computed using $Err = (FP + FN)/(n_{test})$, where FP and FN are the number of false positive and false negative, respectively from a total number of the test data, n_{test}. The errors on test sets for E1 and E2 are 4.6673% and 11.3697%, respectively. This error evaluates the dataset by the classifier without considering the class prior. The error is higher for the test set in E2, thus indicating that the generalization to E2 is indeed more difficult than the generalization to E1's test set.

The uncertainty of the Naive Bayes classification can be modeled with the Shannon entropy (8).

$$H = \sum_{k=1}^{2} p(C_k|\mathbf{q}) \log_2(p(C_k|\mathbf{q})) \tag{8}$$

In the top part of Fig. 11 and Fig. 12, the red line is the roughness. In the bottom part of these figures, the blue line shows the uncertainty of the outputs from the Naive Bayes classifier with the Shannon entropy. In Fig. 11, the uncertainty can be observed at the edges of the obstacles. It is reasonable as only a very small part of the obstacle was captured in the image. In Fig. 12, the entropy gives a more continuous uncertainty of the outputs due to the difference in visual appearance in E2. There is one part (e.g. at $25s$ to $30s$) where they both agree with their outputs because the field of view of the camera consists of largely the same gray ground on the right side of test field (see Fig. 9). By using this information, the MAV is able to detect the change of environment and trigger the optical flow approach to re-train the linear regression model so that it can adapt itself to the new environment.

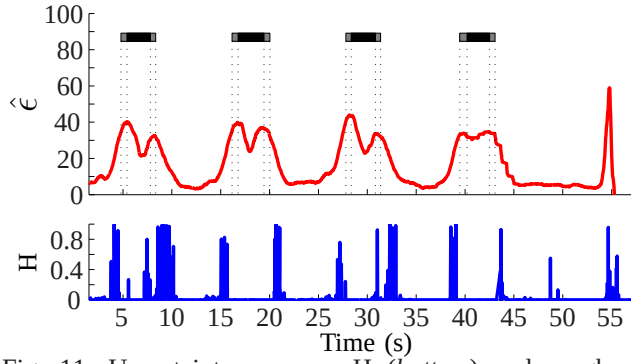


Fig. 11: Uncertainty measure H (*bottom*) and roughness estimate $\hat{\epsilon}$ (*top*) in E1.

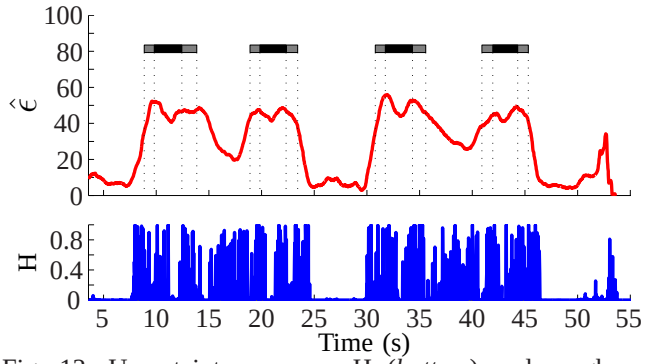


Fig. 12: Uncertainty measure H (*bottom*) and roughness estimate $\hat{\epsilon}$ (*top*) in E2.

V. CONCLUSIONS

We have introduced a novel setup for SSL, in which for the first time optical flow provides the supervised outputs. The surface roughness ϵ^* from the optical flow algorithm allows obstacle detection and safe landing spot selection when the MAV has lateral movement. A linear function is learned that maps texton distributions to ϵ^*. We have shown that $\hat{\epsilon}$ from SSL does not only manage to detect, but even segment obstacles, without having to move. Both methods led to successful landings in indoor experiments with a Parrot AR drone running all vision on-board. Additionally, we have investigated the generalization of the SSL method to different environments using a Naive Bayes classifier and Shannon entropy. Our future work will focus on: (1) re-training the model when detecting uncertain output, (2) examining the effect of height on the roughness, and (3) developing landing strategies based on the roughness for MAVs as estimated with appearance features.

REFERENCES

[1] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2009, pp. 733 219–733 219.

[2] S. B. Goldberg, M. W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *Aerospace Conference Proceedings, 2002. IEEE*, vol. 5. IEEE, 2002, pp. 5–2025.

[3] R. Brockers, Y. Kuwata, S. Weiss, and L. Matthies, "Micro air vehicle autonomous obstacle avoidance from stereo-vision," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014, pp. 90 840O–90 840O.

[4] J. Park and Y. Kim, "Landing site searching algorithm of a quadrotor using depth map of stereo vision on unknown terrain," *AIAA Infotech@ Aerospace 2012*, pp. 19–21, 2012.

[5] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles*, vol. 1, no. 4, pp. 217–228, 2009.

[6] K. Celik, S.-J. Chung, and A. K. Somani, "Mvcslam: mono-vision corner slam for autonomous micro-helicopters in gps denied environments," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.

[8] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *Robotics and automation (ICRA), 2010 IEEE international conference on*. IEEE, 2010, pp. 21–28.

[9] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[10] E. Baird, M. V. Srinivasan, S. Zhang, and A. Cowling, "Visual control of flight speed in honeybees," *Journal of Experimental Biology*, vol. 208, no. 20, pp. 3895–3905, 2005.

[11] E. Baird, M. V. Srinivasan, S. Zhang, R. Lamont, and A. Cowling, "Visual control of flight speed and height in the honeybee," in *From Animals to Animats 9*. Springer, 2006, pp. 40–51.

[12] N. Franceschini, S. Viollet, F. Ruffier, and J. Serres, "Neuromimetic robots inspired by insect vision," *Advances in Science and Technology*, vol. 58, pp. 127–136, 2009.

[13] F. Ruffier and N. Franceschini, "Aerial robot piloted in steep relief by optic flow sensors," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 1266–1273.

[14] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto, "The landing problem of a vtol unmanned aerial vehicle on a moving platform using optical flow," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1600–1605.

[15] D. Izzo and G. C. H. E. de Croon, "Landing with time-to-contact and ventral optic flow estimates," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1362–1367, 2012.

[16] E. Baird, N. Boeddeker, M. R. Ibbotson, and M. V. Srinivasan, "A universal strategy for visually guided landing," *Proceedings of the National Academy of Sciences*, vol. 110, no. 46, pp. 18 686–18 691, 2013.

[17] R. Brockers, M. Hummenberger, S. Weiss, and L. Matthies, "Towards autonomous navigation of miniature uav," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 645–651.

[18] V. Desaraju, N. Michael, M. Humenberger, roland Brockers, S. Weiss, and L. Matthies, "Vision-based landing site evaluation and trajectory generation toward rooftop landing," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[19] V. Grabe, H. H. Bülthoff, and P. R. Giordano, "Robust optical-flow based self-motion estimation for a quadrotor uav," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2153–2159.

[20] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[21] U. A. Muller, L. D. Jackel, Y. LeCun, and B. Flepp, "Real-time adaptive off-road vehicle navigation and terrain classification," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2013, pp. 87 410A–87 410A.

[22] A. Lookingbill, J. Rogers, D. Lieb, J. Curry, and S. Thrun, "Reverse optical flow for self-supervised adaptive autonomous robot navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 287–302, 2007.

[23] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 208, no. 1173, pp. 385–397, 1980.

[24] G. C. H. E. de Croon, H. W. Ho, C. De Wagter, E. Van Kampen, B. Remes, and Q. Chu, "Optic-flow based slope estimation for autonomous landing," *International Journal of Micro Air Vehicles*, vol. 5, no. 4, pp. 287–298, 2013.

[25] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[26] ——, "Fusing points and lines for high performance tracking," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1508–1515.

[27] J. Bouquet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs, http://www. intel. com/research/mrl/research/opencv*, 2000.

[28] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[29] M. Varma and A. Zisserman, "Texture classification: Are filter banks necessary?" in *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on*, vol. 2. IEEE, 2003, pp. II–691.

[30] G. C. H. E. de Croon, E. De Weerdt, C. De Wagter, B. Remes, and R. Ruijsink, "The appearance variation cue for obstacle avoidance," *Robotics, IEEE Transactions on*, vol. 28, no. 2, pp. 529–534, 2012.

[31] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[32] G. C. H. E. De Croon, K. De Clercq, R. Ruijsink, B. Remes, and C. De Wagter, "Design, aerodynamics, and vision-based control of the delfly," *International Journal of Micro Air Vehicles*, vol. 1, no. 2, pp. 71–97, 2009.

[33] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. De Ridder, D. Tax, and S. Verzakov, "A matlab toolbox for pattern recognition," *PRTools version*, vol. 3, 2000.

[34] A. Cesetti, E. Frontoni, A. Mancini, and P. Zingaretti, "Autonomous safe landing of a vision guided helicopter," in *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*. IEEE, 2010, pp. 125–130.