Delft University of Technology

Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones

Mcguire, Kimberly; de Croon, Guido; de Wagter, Christophe; Remes, Bart; Tuyls, K.; Kappen , H.

DOI
10.1109/ICRA.2016.7487496

Publication date
2016

Document Version
Accepted author manuscript

Published in
2016 IEEE International Conference on Robotics and Automation (ICRA)

Citation (APA)
Mcguire, K., de Croon, G., de Wagter, C., Remes, B., Tuyls, K., & Kappen , H. (2016). Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones. In A. Okamura (Ed.), *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3255-3260). IEEE. https://doi.org/10.1109/ICRA.2016.7487496

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Local Histogram Matching for Efficient Optical Flow Computation Applied to Velocity Estimation on Pocket Drones

Kimberly McGuire[1], Guido de Croon[1], Christophe de Wagter[1], Bart Remes[1], Karl Tuyls[2,1], and Hilbert Kappen[3]

***Abstract*— Autonomous flight of pocket drones is challenging due to the severe limitations on on-board energy, sensing, and processing power. However, tiny drones have great potential as their small size allows maneuvering through narrow spaces while their small weight provides significant safety advantages. This paper presents a computationally efficient algorithm for determining optical flow, which can be run on an STM32F4 microprocessor (168 MHz) of a 4 gram stereo-camera. The optical flow algorithm is based on edge histograms. We propose a matching scheme to determine local optical flow. Moreover, the method allows for sub-pixel flow determination based on time horizon adaptation. We demonstrate velocity measurements in flight and use it within a velocity control-loop on a pocket drone.**

I. INTRODUCTION

Pocket drones are Micro Air Vehicles (MAVs) small enough to fit in one's pocket and therefore small enough to maneuver in narrow spaces (Fig. 1). The pocket drones' light weight and limited velocity make them inherently safe for humans. Their agility makes them ideal for search-and-rescue exploration in disaster areas (e.g. in partially collapsed buildings) or other indoor observation tasks. However, autonomous flight of pocket drones is challenging due to the severe limitations in on-board energy, sensing, and processing capabilities.

To deal with these limitations it is important to find efficient algorithms to enable low-level control on these aircraft. Examples of low-level control tasks are stabilization, velocity control and obstacle avoidance. To achieve these tasks, a pocket drone should be able to determine its own velocity, even in GPS-deprived environments. This can be done by measuring the optical flow detected with a bottom mounted camera [1]. Flying insects like honeybees use optical flow as well for these low-level tasks [2]. They serve as inspiration as they have limited processing capacity but can still achieve these tasks with ease.

Determining optical flow from sequences of images can be done in a dense manner with, e.g., Horn-Schunck [3], or with more recent methods like Farnebäck [4]. In robotics, computational efficiency is important and hence sparse optical flow is often determined with the help of a feature detector such as Shi-Tomasi [5] or FAST [6], followed by Lucas-Kanade feature tracking [7]. Still, such a setup does not fit the processing limitations of a pocket drone's hardware, even if one is using small images.

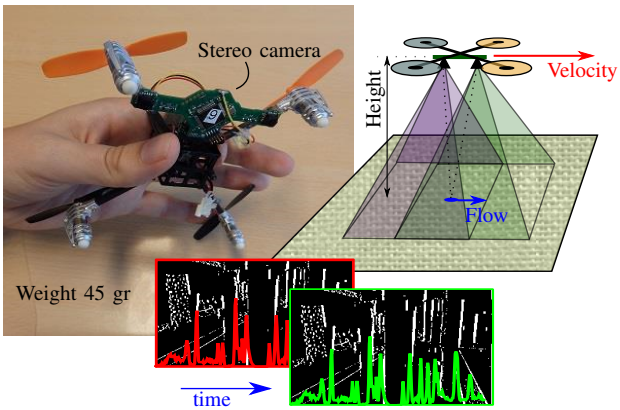


Fig. 1: Pocket drone with velocity estimation using a downward looking stereo vision system. A novel efficient optical flow algorithm runs on-board an STM32F4 processor running at only 168 MHz and with only 192 kB of memory. The so-determining optical flow and height, with the stereo-camera, provide the velocity estimates necessary for the pocket drone's low level control are obtained.

Optical flow based stabilization and velocity control is done with larger MAVs with a diameter of 50 cm and up [8][9]. As these aircraft can carry small commercial computers, they can calculate optical flow with more computationally heavy algorithms. A MAV's size is highly correlated on what it can carry and a pocket drone, which fits in the palm of your hand, cannot transport these types resources and therefore has to rely on off-board computing.

A few researchers have achieved optical flow based control fully on-board a tiny MAV. Dunkley et al. have flown a 25 gram helicopter with visual-inertial SLAM for stabilization, for which they use an external laptop to calculate its position by visual odometry [10]. Briod et al. produced on-board processing results, however they use multiple optical flow sensors which can only detect one direction of movement [11]. If more sensing capabilities are needed, the use of single-purpose sensors is not ideal. A combination of computer vision and a camera will result in a single, versatile, sensor, able to detect multiple variables and therefore saves weight on a tiny MAV. By limiting the weight it needs to carry, will increase its flight time significantly.

Closest to our work is the study by Moore et al., in which multiple optical flow cameras are used for obstacle avoidance

*This work was not supported by any organization
[1]Delft University of Technology, The Netherlands k.n.mcguire@tudelft.nl
[2] University of Liverpool, United Kingdom
[3] Radboud University of Nijmegen, The Netherlands

[12]. Their vision algorithms heavily compress the images, apply a Sobel filter and do Sum of Absolute Difference (SAD) block matching on a low-power 32-bit Atmel micro controller (AT32UC3B1256).

This paper introduces a novel optical flow algorithm, computationally efficient enough to be run on-board a pocket drone. It is inspired by the optical flow method of Lee et al. [13], where image gradients are summed for each image column and row to obtain a horizontal and vertical edge histogram. The histograms are matched over time to estimate a global divergence and translational flow. In [13] the algorithm is executed off-board with a set of images, however it shows great potential. In this paper, we extend the method to calculate local optical flow as well. This can be fitted to a linear model to determine both translational flow and divergence. The later will be unused in the rest of this paper as we are focused on horizontal stabilization and velocity control. However, it will become useful for autonomous obstacle avoidance and landing tasks. Moreover, we introduce an adaptive time horizon rule to detect sub-pixel flow in case of slow movements. Equipped with a downward facing stereo camera, the pocket drone can determine its own speed and do basic stabilization and velocity control.

The remainder of this paper is structured as follows. In Section II, the algorithm is explained with off-board results. Section III will contain velocity control results of two MAVs, an AR.Drone 2.0 and a pocket drone, with both using the same 4 gr stereo-camera containing the optical flow algorithm on-board. Section III will conclude these results and give remarks for future research possibilities.

II. OPTICAL FLOW WITH EDGE FEATURE HISTOGRAMS

This section explains the algorithm for the optical flow detection using edge-feature histograms. The gradient of the image is compressed into these histograms for the horizontal and vertical direction. This reduces the 2D image search problem to 1D signal matching, increasing its computational efficiency. Therefore, this algorithm is efficient enough to be run on-board a 4 gram stereo-camera module, which can used by an MAV to determine its own velocity.

A. Edge Features Histograms

The generated edge feature histograms are created by first calculating the gradient of the image on the vertical and horizontal axis using a Sobel filter (Fig. 2(a)). From these gradient intensity images, the histogram can be computed for each of the image's dimensions by summing up the intensities. The result is an edge feature histogram of the image gradients in the horizontal and vertical directions.

From two sequential frames, these edge histograms can be calculated and matched locally with the Sum of Absolute Differences (SAD). In Fig. 2(b), this is done for a window size of 18 pixels and a maximum search distance of 10 pixels in both ways. The displacement can be fitted to a linear model with least-square line fitting. This model has two parameters: a constant term for translational flow and a slope

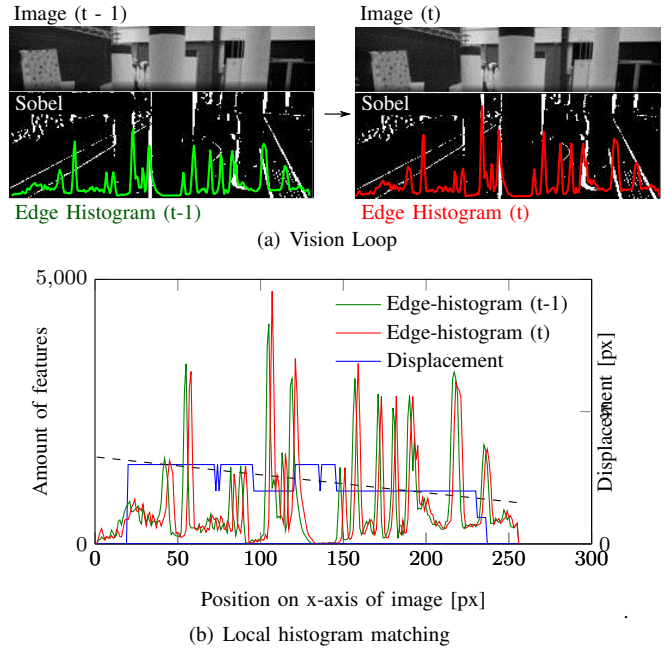


Fig. 2: (a) The vision loop with for creating the edge feature histograms and (b) the matching of the two histograms (previous frame (green) and current frame (red)) with SAD. This results in a pixel displacement (blue) which can be fitted to a linear model (dashed black line) for a global estimation.

for divergence. Translational flow stands for the translational motion between the sequential images, which is measured if the camera is moved sideways. The slope/divergence is detected when a camera moves to and from a scene. In case of the displacement shown in Fig. 2(b) both types of flows are observed, however only translation flow will be considered in the remainder of this paper.

B. Time Horizon Adaptation for Sub-Pixel Flow

The previous section explained the matching of the edge feature histograms which gives translational flow. Due to a image sensor's resolution, existing variations within pixel boundaries can not be measured, so only integer flows can be considered. However, this will cause complication if the camera is moving slowly or is well above the ground. If these types of movements result in sub-pixel flow, this cannot be observed with the current state of the edge flow algorithm. This sub-pixel flow is important for to ensure velocity control on an MAV.

To ensure the detection of sub-pixel flow, another factor is added to the algorithm. Instead of the immediate previous frame, the current frame is also compared with a certain time horizon n before that. The longer the time horizon, the more resolution the sub-pixel flow detection will have. However, for higher velocities it will become necessary to compare the current edge histogram to the closest time horizon as possible. Therefore, this time horizon comparison must be adaptive.

Which time horizon to use for the edge histogram match-

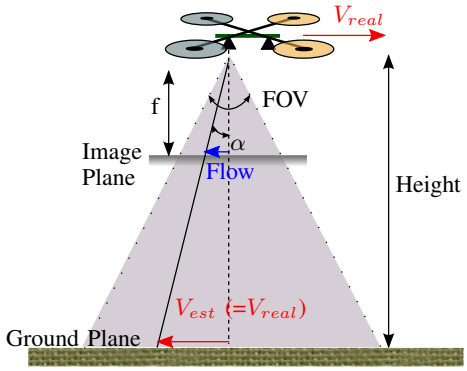


Fig. 3: Velocity estimation by measuring optical flow with one camera and height with both cameras of the stereo-camera.

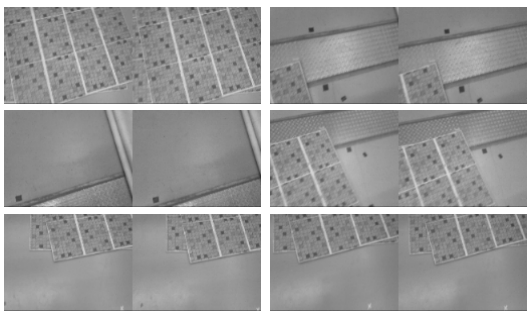


Fig. 4: Several screen shots of the set of images used for off-line estimation of the velocity. Here the diversity in amount of texture can be seen.

ing, is determined by the translational flow calculated in the previous time step p_{t-1}:

$$n = \min\left(\frac{1}{|p_{t-1}|}, N\right) \tag{1}$$

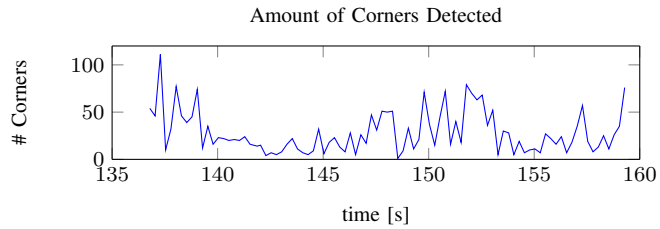
where n is the number of the previous stored edge histogram that the current frame is compared to. The second term, N, stands for the maximum number of edge histograms allowed to be stored in the memory. It needs to be limited due to the strict memory requirements and in our experiments is set to 10. Once the current histogram and time horizon histogram are compared, the resulting flow must be divided by n to obtain the flow per frame.

C. Velocity Estimation on Set of Images

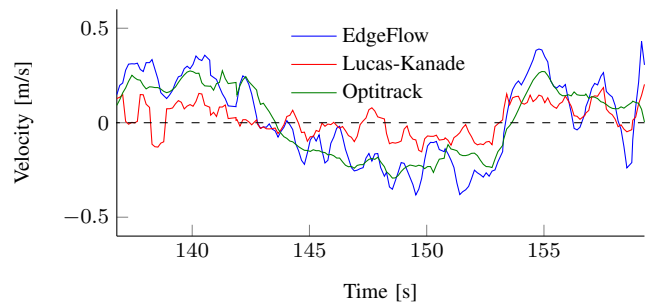
The previous sections explained the calculation of the translational flow, for convenience now dubbed as *EdgeFlow*. As seen in Fig. 3, the velocity estimation V_{est} can be calculated with the height of the drone and the angle from the center axis of the camera:

$$V_{est} = h * tan(p_t * FOV/w)/\Delta t \tag{2}$$

where p_t is the flow vector, h is the height of the drone relative to the ground, and w stands for the pixels size of the image (in x or y direction). FOV stands for the field-of-view of the image sensor. A MAV can monitor its height by means of a sonar, barometer or GPS. In our case we do it differently, as we match the left and right edge histogram from the stereo-camera with global SAD matching. This implies that only one sensor is used for both velocity and height estimation.



(a) Amount of corners per time frame of the data set sample.



(b) Smoothed velocity estimates of EdgeFlow and Lucas-Kanade.

	Lucas-Kanade		EdgeFlow	
	hor.	ver.	hor.	ver.
MSE	0.0696	0.0576	0.0476	0.0320
NMXM	0.3030	0.3041	0.6716	0.6604
Comp. Time	0.1337 s		0.0234 s	

(c) Comparison values for EdgeFlow and Lucas-Kanade.

Fig. 5: Off-line results of the optical flow measurements: (a) the measure of feature-richness of the image data-set by Shi-Tomasi corner detection and (b) a comparison of Lucas-Kanade and EdgeFlow with horizontal velocity estimation. In (c), the MSE and NMXM values are shown for the entire data set of 440 images, compared to the OptiTrack's measured velocities.

For off-board velocity estimation, a dataset of stereo-camera images is produced and synchronized with ground truth velocity data. The ground truth is measured by a motion tracking system with reflective markers (OptiTrack, 24 infrared-cameras). This dataset excites both the horizontal and vertical flow directions, which is equivalent to the x- and y-axis of the image plane, and contains areas of varying amounts of textures (Fig. 4). As an indication of the texture-richness of the surface, the number of features, as detected by the Shi-Tomasi corner detection, is plotted in Fig. 5(a).

For estimating the velocity, the scripts run in Matlab R2014b on a Dell Latitude E7450 with an Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz processor. In Fig. 5(b), the results of a single pyramid-layer implementation of the Lucas-Kanade algorithm with Shi-Tomasi corner detection can be seen (from [7]). The mean of the detected horizontal

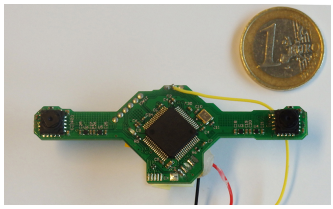


Fig. 6: 4 gram stereo-camera with a STM32F4 microprocessor with only 168 MHz speed and 192 kB of memory. The two cameras are located 6 cm apart from each other.

velocity vectors is shown per time frame and plotted against the measured velocity by the OptiTrack system, as well as the velocity measured by EdgeFlow. For Lucas-Kanade, the altitude data of the OptiTrack is used. For EdgeFlow, the height is determined by the stereo images alone by histogram matching.

In Fig. 5(c), comparison values are shown of the EdgeFlow and Lucas-Kanade algorithm of the entire data set. The mean squared error (MSE) is lower for EdgeFlow than for Lucas-Kanade, where a lower value stands for a higher similarity between the compared velocity estimation and the OptiTrack data. The normalized maximum cross-correlation magnitude (NMXM) is used as a quality measure as well. Here a higher value, between a range of 0 and 1, stands for a better shape correlation with the ground truth. The plot of Fig. 5(b) and the values in Fig. 5(c) shows a better tracking of the velocity by EdgeFlow when compared. We think that the main reason for this is that it utilizes information present in lines, which are ignored in the corner detection stage of Lucas-Kanade. In terms of computational speed, the EdgeFlow algorithm has an average processing of 0.0234 sec for both velocity and height estimation, over 5 times faster than Lucas-Kanade. Although this algorithm is run off-board on a laptop computer, it is an indication of the computational efficiency of the algorithm. This is valuable as EdgeFlow needs to run embedded on the 4 gr stereo-board, which is done in the upcoming sections of this paper.

III. VELOCITY ESTIMATION AND CONTROL

The last subsection showed results with a data set of stereo images and OptiTrack data. In this section, the velocity estimated by EdgeFlow is run on-board the stereo-camera. Two platforms, an AR.Drone 2.0 and a pocket drone, will utilize the downward facing camera for velocity estimation and control. Fig. 7(a) gives a screen-shot of the video of the experiments[1], where it can be seen that the pocket drone is flying over a feature-rich mat.

A. Hardware and Software Specifics

The AR.Drone 2.0[2] is a commercial drone with a weight of 380 grams and about 0.5 meter (with propellers considered)

[1] YouTube playlist: https://www.youtube.com/playlist?list=PL_KSX9GOn2P9TPb5nmFg-yH-UKC9eXbEE

[2] http://wiki.paparazziuav.org/wiki/AR_Drone_2

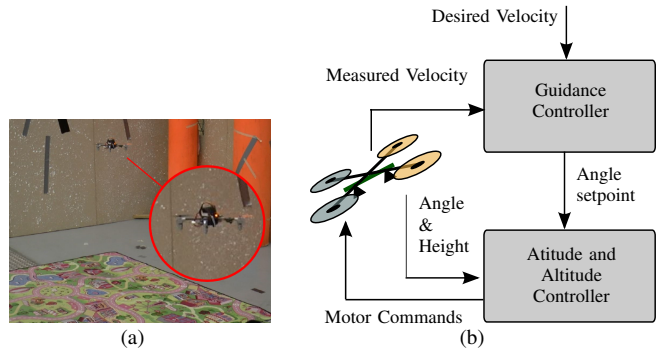
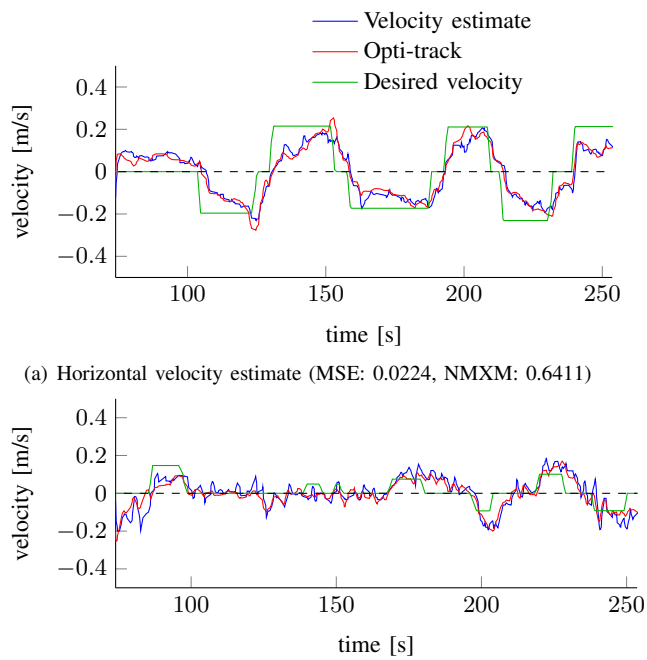


Fig. 7: (a) A screen-shot of the video of the flight and (b) the control scheme of the velocity control.



(a) Horizontal velocity estimate (MSE: 0.0224, NMXM: 0.6411)

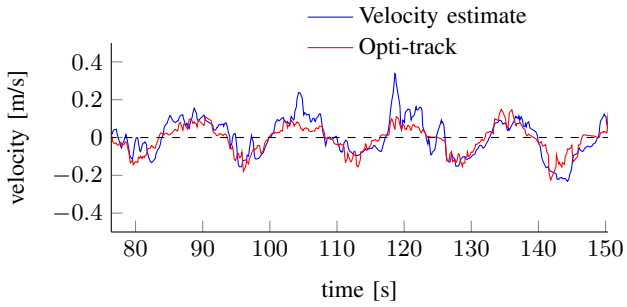
(b) Vertical velocity estimate (MSE: 0.0112, NMXM: 0.6068)

Fig. 8: The velocity estimate of the AR.Drone 2.0 and stereo-board assembly during a velocity control task with ground-truth as measured by OptiTrack. MSE and NMXM values are calculated for the entire flight.

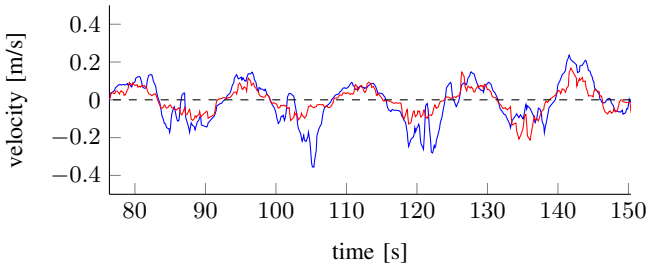
in diameter. The pocket drone[3] is 10 cm in diameter and has a total weight of 40 grams (including battery). It contains a Lisa S autopilot [14], which is mounted on a LadyBird quadcopter frame. The drone's movement is tracked by a motion tracking system, OptiTrack, which tracks passive reflective markers with its 24 infrared cameras. The registered motion will be used as ground truth to the experiments.

The stereo-camera, introduced in [15], is attached to the bottom of both drones, facing downward to the ground plane (Fig. 6). It has two small cameras with two 1/6 inch image sensors, which are 6 cm apart. They have a horizontal FOV of 57.4^o and vertical FOV of 44.5^o. The processor type

[3] http://wiki.paparazziuav.org/wiki/Lisa/S/Tutorial/Nano_Quadcopter

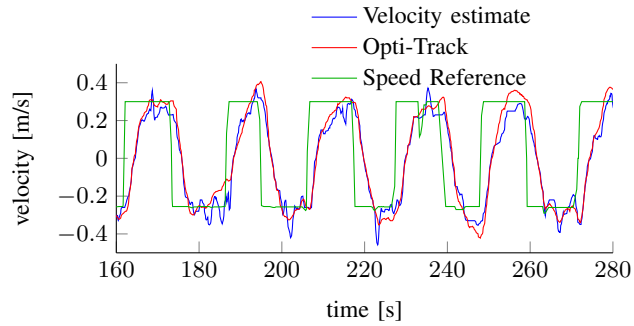


(a) Horizontal velocity estimate (MSE: 0.0064 m, NMXM: 0.5739)

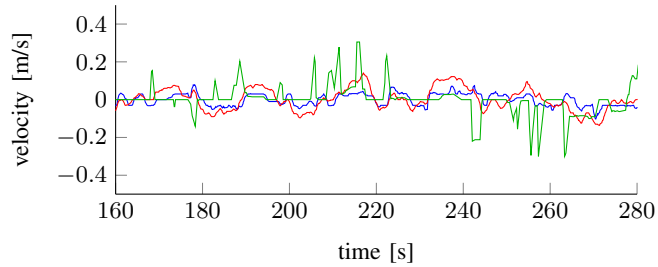


(b) Vertical velocity estimate (MSE: 0.0072 m, NMXM: 0.6066)

Fig. 9: Velocity estimates calculated by the pocket drone and stereo-board assembly, during an OptiTrack guided flight. MSE and NMXM values are calculated for the entire flight.



(a) Horizontal velocity estimate (MSE: 0.0041 m, NMXM: 0.9631)



(b) Vertical velocity estimate (MSE: 0.0025 m, NMXM: 0.7494)

Fig. 10: Velocity estimates calculated by the pocket drone and stereo-board assembly, now using estimated velocity in the control. MSE and NMXM values are calculated for the entire flight which lasted for 370 seconds, where several external speed references were given for guidance.

is a STM32F4 with a speed of 168 MHz and 192 kB of memory. The processed stereo-camera images are grayscale and have 128×96 pixels. The maximum frame rate of the stereo-camera is 29 Hz, which is brought down to 25 Hz by the computation of EdgeFlow, with its average processing time of 0.0126 seconds. This is together with the height estimation using the same principle, all implemented on-board the stereo-camera.

The auto-pilot framework used for both MAV is Paparazzi[4]. The AR.Drone 2.0's Wi-Fi and the pocket drone's Bluetooth module is used for communication with the Paparazzi ground, station to receive telemetry and send flight commands. Fig. 7(b) shows the standard control scheme for the velocity control as implemented in paparazzi, which will receive a desired velocity references from the ground station for the guidance controller. This layer will send angle setpoints to the attitude controller. The MAV's height should be kept constant by the altitude controller and measurements from the sonar (AR.drone) and barometer (pocket drone). Note that for these experiments, the height measured by the stereo-camera is only used for determining the velocity on-board and not for the control of the MAV's altitude.

B. On-Board Velocity Control of a AR.Drone 2.0

In this section, an AR.Drone 2.0 is used for velocity control with EdgeFlow, using the stereo-board instead of its standard bottom camera. Its difference with the desired velocity serves as the error signal for the guidance controller. During the flight, several velocity references were sent to the AR.Drone, making it fly into specific direction. In Fig. 8, the stereo-camera's estimated velocity is plotted against its velocity measured by the OptiTrack for both horizontal and vertical direction of the image plane. This is equivalent to respectively sideways and forward direction in the AR.Drone's body fixed coordinate system.

The AR.Drone was is able to determine its velocity with EdgeFlow computed on-board the stereo-camera, as the MSE and NMXM quality measures indicate a close correlation with the ground truth. This results in the AR.Drone's ability to correctly respond to the speed references given to the guidance controller

.

C. On-board Velocity Estimation of a Pocket Drone

In the last subsection, we presented velocity control of an AR.Drone 2.0 to show the potential of using the stereo-camera for efficient velocity control. However, this needs to be shown on the pocket drone as well, which is smaller and hence has faster dynamics. Here the pocket drone is flown based on OptiTrack position measurement to present its on-board velocity estimation without using it in the control loop. During this flight, the velocity estimate calculated by the stereo-board is logged and plotted against its ground truth (Fig. 9).

The estimated velocity by the pocket drone is noisier than with the AR.Drone, which can be due of multiple reasons, from which the first is that the stereo-board is subjected to

[4]http://wiki.paparazziuav.org/

more vibrations on the pocket drone than the AR.Drone. This is because the camera is much closer to the rotors of the MAV and mounted directly on the frame. Another thing would be the control of the pocket drone, since it responds much faster as the AR.Drone. Additional filtering and de-rotation are essential to achieve the full on-board velocity control.

De-rotation is compensating for the camera rotations, where EdgeFlow will detect a flow not equivalent to translational velocity. Since the pocket drone has faster dynamics than the AR.Drone, the stereo-camera is subjected to faster rotations. De-rotation must be applied in order for the pocket drone to use optical flow for controls. In the experiments of the next subsection, the stereo-camera will receive rate measurement from the gyroscope. Hre it can estimate the resulting pixel shift in between frames due to rotation. The starting position of the histogram window search in the other image is offset with that pixel shift (an addition to section II A).

D. On-board Velocity Control of a Pocket drone

Now the velocity estimate is used in the guidance control of the pocket drone and the OptiTrack measurements is only used for validation. The pocket drone's flight, during a guidance control task with externally given speed references, lasted for 370 seconds. Mostly horizontal (sideways) speed references where given, however occasional horizontal speed references in the vertical direction were necessary to keep the pocket drone flying over the designated testing area. A portion of the velocity estimates during that same flight are displayed in Fig. 10. From the MSE and NMXM quality values for the horizontal speed, it can be determined that the EdgeFlow's estimated velocity correlates well with the ground truth. The pocket drone obeys the speed references given to the guidance controller.

Noticeable in Fig. 10(b) is that the NMXM for vertical direction is lower than for the horizontal. As most of the speed references send to the guidance controller were for the horizontal direction, the correlation in shape is a lot more eminent, hence resulting in a higher NMXM value. Overall, it can be concluded that pocket drone can use the 4 gr stereo-board for its own velocity controlled guidance.

IV. CONCLUSION

In this paper we introduced a computationally efficient optical flow algorithm, which can run on a 4 gram stereo-camera with limited processing capabilities. The algorithm EdgeFlow uses a compressed representation of an image frame to match it with a previous time step. The adaptive time horizon enabled it to also detect sub-pixel flow, from which slower velocity could be estimated.

The stereo-camera is light enough to be carried by a 40 gram pocket drone. Together with the height and the optical flow calculated on-board, it can estimate its own velocity. The pocket drone uses that information within a guidance control loop, which enables it to compensate for drift and respond to external speed references. Our next focus is to use the same principle for a forward facing camera.

REFERENCES

[1] P.-J. Bristeau, F. Callou, D. Vissiere, N. Petit, *et al.*, "The navigation and control technology inside the ar. drone micro uav," in *18th IFAC world congress*, vol. 18, no. 1, 2011, pp. 1477–1484.

[2] M. V. Srinivasan, "Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics," *Physiological Reviews*, vol. 91, no. 2, pp. 413–460, 2011.

[3] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical symposium east*. International Society for Optics and Photonics, 1981, pp. 319–331.

[4] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*. Springer, 2003, pp. 363–370.

[5] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[6] E. Rosten and T. Drummond, "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.

[7] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, pp. 1–10, 2001.

[8] H. Romero, S. Salazar, and R. Lozano, "Real-time stabilization of an eight-rotor uav using optical flow," *Robotics, IEEE Transactions on*, vol. 25, no. 4, pp. 809–817, 2009.

[9] V. Grabe, H. H. Bülthoff, D. Scaramuzza, and P. R. Giordano, "Nonlinear ego-motion estimation from optical flow for online control of a quadrotor uav," *The International Journal of Robotics Research*, p. 0278364915578646, 2015.

[10] O. Dunkley, J. Engel, J. Sturm, and D. Cremers, "Visual-inertial navigation for a camera-equipped 25g nano-quadrotor," *IROS2014 Aerial Open Source Robotics Workshop*, 2014.

[11] A. Briod, J.-C. Zufferey, and D. Floreano, "Optic-flow based control of a 46g quadrotor," in *IROS 2013, Vision-based Closed-Loop Control and Navigation of Micro Helicopters in GPS-denied Environments Workshop*, no. EPFL-CONF-189879, 2013.

[12] R. J. Moore, K. Dantu, G. L. Barrows, and R. Nagpal, "Autonomous mav guidance with a lightweight omnidirectional vision sensor," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3856–3861.

[13] D.-J. Lee, R. W. Beard, P. C. Merrell, and P. Zhan, "See and avoidance behaviors for autonomous navigation," in *Optics East*. International Society for Optics and Photonics, 2004, pp. 23–34.

[14] B. Remes, P. Esden-Tempski, F. Van Tienen, E. Smeur, C. De Wagter, and G. de Croon, "Lisa-s 2.8 g autopilot for gps-based flight of mavs," in *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014*. Delft University of Technology, 2014.

[15] C. de Wagter, S. Tijmons, B. Remes, and G. de Croon, "Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4982–4987.