Optic-flow based slope estimation for autonomous landing

G.C.H.E. de Croon, H.W. Ho, C. De Wagter,
E. van Kampen, B. Remes, and Q.P. Chu*†

Abstract

Micro Air Vehicles need to have a robust landing capability, especially when they operate outside line-of-sight. Autonomous landing requires the identification of a relatively flat landing surface that does not have too large an inclination. In this article, a vision algorithm is introduced that fits a second-order approximation to the optic flow field underlying the optic flow vectors in images from a bottom camera. The flow field provides information on the ventral flow (V_x/h), the time-to-contact ($h/-V_z$), the flatness of the landing surface, and the surface slope. The algorithm is computationally efficient and since it regards the flow field as a whole, it is suitable for use during relatively fast maneuvers. The algorithm is subsequently tested on artificial image sequences, hand-held videos, and on the images made by a Parrot AR drone. In a preliminary robotic experiment, the AR drone uses the vision algorithm to determine when to land in a scenario where it flies off a stairs onto the flat floor.

1 Introduction

Autonomous landing is an important capability for Micro Air Vehicles (MAVs), especially if they have to operate outside line-of-sight. While a barometer

*Micro Air Vehicle lab, Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands.

†AUTHOR DRAFT: final version published in the International Journal of Micro Air Vehicles, http://dx.doi.org/10.1260/1756-8293.5.4.287

provides information on altitude, it does not provide information on the height of the MAV above the terrain, nor on the suitability of the terrain for landing. Active sensors such as a laser scanner [2] or multiple cameras as in stereo vision [1] do offer such information. They instantaneously estimate distances to many points on the landing surface. However, such sensor setups only work at lower heights and are not energy or weight efficient. Since such efficiency is important for MAVs, it would be beneficial to have a robust landing strategy based on a single downward-pointing camera.

With a monocular camera setup, two main approaches to autonomous landing can be employed. The first approach is visual Simultaneous Localization And Mapping (SLAM) [8, 6], in which the 3D locations of all features in sight are determined. Various approaches to visual SLAM have been proposed over the years, which have consistently improved with respect to accuracy and computational effort [9, 23]. Still, SLAM delivers a lot of detailed information on the environment that is not strictly required for the landing task, and hence uses more computational resources than necessary.

The second approach is bio-inspired in the sense that it directly uses the optic flow field for control. The earliest studies of this approach [14, 19] are based on the biological finding that bees keep the ventral flow constant during a grazing landing [4, 5]. The ventral flow is equal to the translational velocity divided by the height (V_x/h), and keeping it constant results in a soft touch down. Since the ventral flow cannot account for the vertical dynamics, recent engineering studies [15, 16, 20] complement the ventral flow with the time-to-contact, i.e., the height divided by vertical velocity ($h/-V_z$). Interestingly, biologists have now confirmed that one of the engineered strategies [15, 16] is also used by honeybees: they keep the time-to-contact constant while landing on flat surfaces [3]. The advantage of a bio-inspired approach is that lightweight neuromorphic sensors with high sensitivity and update frequency can be used directly for controlling the landing [11, 20].

The above-mentioned bio-inspired landing studies focus on autonomous landing on a flat surface. However, many real-world missions for MAVs will involve unknown landing sites that may be cluttered or have a slope. Indeed, bees can choose their landing site and adapt their body attitude to the slope of the surface on which they are landing [10]. For an MAV the detection of the landing surface's slope would also be of interest, either for adapting the MAV's attitude or for evaluating the suitability of the surface for landing.

In this article, a computationally efficient computer vision algorithm is proposed that uses a bottom camera. First, optic flow vectors are deter-

mined and then a second order fit of the entire optic flow field is made. The fit provides information on the ventral flow, time-to-contact, flatness of the landing surface, and surface slope. The algorithm is computationally efficient and robust to noisy optic flow vectors that are likely to occur in a real MAV landing scenario. In addition, it lends itself well for translation to algorithms for novel sensors that mimick insect eyes [22, 13].

The remainder of the article is structured as follows. In Section 2, the vision algorithm is explained. It is tested on image sequences in Section 3. The results of a landing experiment with a Parrot AR drone are discussed in Section 4. Finally, conclusions are drawn in Section 5.

2 Vision Algorithm for Slope Estimation

2.1 Optic flow equations for a sloped planar surface

The vision algorithm proposed for the slope estimation is based on the early optic flow work in [17]. In the explanation of the algorithm, a pinhole camera model is employed. The algorithm assumes (1) the camera to point downward, (2) the rotation rates to be known from the MAV's state estimation (e.g., using gyros), and (3) the landing surface in sight to be predominantly planar. Under these assumptions, the optic flow vectors in the image follow the equations:

$$u = (-V_x + xV_z)/Z, \tag{1}$$

$$v = (-V_y + yV_z)/Z, \tag{2}$$

with u and v the (derotated) optic flow in the x- and y-direction of the image, and V_x, V_y, V_z the motion in X, Y, and Z direction, respectively. x and y are image coordinates. Figure 1 shows the relevant coordinate axes. Please note that the Z-axis is aligned with the camera's optical axis and that the coordinate $(X, Y, Z) = (0, 0, 0)$ is located at the intersection of the camera's principal axis with the ground surface. The height of the camera above the ground surface is represented with h. The surface height Z can be modelled as a plane:

$$Z = h + aX + bY, \tag{3}$$

with h the height above the surface at $(x, y) = (0, 0)$. The slope angles of the surface are:

$$\alpha = \text{atan}(a) \tag{4}$$

$$\beta = \text{atan}(b) \tag{5}$$

By a transformation of coordinates, in [17], the surface height is rewritten as:

$$z = (Z - h)/Z = ax + by, \tag{6}$$

, where $x = X/Z$ and $y = Y/Z$. Please remark that the normalized x-coordinate in Eq. 6 is related to the pixel coordinate $\tilde{x}$ by $x = \tilde{x}/\tilde{f}$, where $\tilde{f}$ is the focal length in pixels. In order to keep the equations uncluttered, normalized coordinates will be used for the remainder of the article. However, one should keep in mind that the actual slope angle in degrees can only be retrieved in the case of a calibrated camera (with $\tilde{f}$ known).

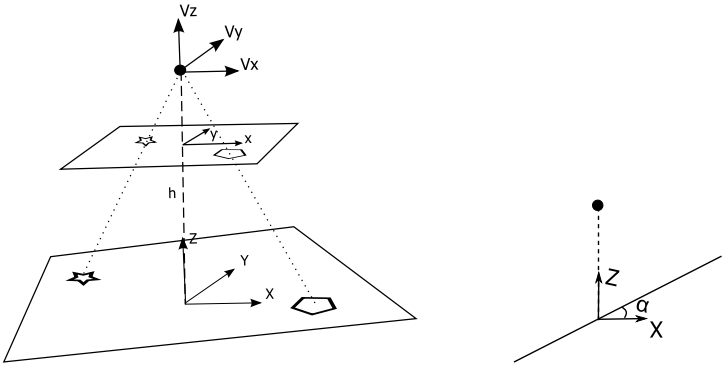

Figure 1: Left: the main coordinate axes involved in the optic flow calculations. Right: illustration of a ground surface with a nonzero slope α.

In addition to the coordinate transformation, the velocities are scaled with respect to the height h, i.e., $\omega_x = V_x/h$, $\omega_y = V_y/h$, and $\omega_z = V_z/h$. This leads to:

$$u = (-\omega_x + x\omega_z)(1 - z) \tag{7}$$

$$v = (-\omega_y + y\omega_z)(1 - z) \tag{8}$$

Replacing z with $ax + by$ (from Eq. 6), leads to:

$$u = -\omega_x + (\omega_x a + \omega_z)x + \omega_x by - a\omega_z x^2 - b\omega_z xy \tag{9}$$

$$v = -\omega_y + \omega_y ax + (\omega_y b + \omega_z)y - b\omega_z y^2 - a\omega_z xy \tag{10}$$

2.2 Optic flow field parameter estimation

In [17], the rotational optic flow terms were left in, and the discussion hence goes toward how to find the Focus-of-Expansion (FoE) and determine the first and second order spatial flow derivatives at that point. In that way, all terms can be identified (translational, rotational, and the slopes of the surface). However, determining the FoE is a difficult task in itself, and errors in its location can have a large influence on the subsequent results. In addition, since MAVs typically have access to gyro measurements, the rotational components do not have to be determined. So instead of finding the FoE, the vision algorithm proposed in this article immediately determines the parameters of the optic flow field:

$$u = \mathbf{p_u}[1, x, y, x^2, xy]^T, \tag{11}$$

$$v = \mathbf{p_v}[1, x, y, y^2, xy]^T, \tag{12}$$

The parameter vectors $\mathbf{p_u}$ and $\mathbf{p_v}$ are estimated separately with a maximal likelihood linear least squares estimate within a robust random sample consensus (RANSAC) estimation procedure [12], with 5 points per fit and 20 iterations. Figure 2 illustrates the result of such a fit.

2.3 Extraction of variables relevant for landing

The so-determined parameters can then be used to estimate the following variables of interest for an autonomous landing. The **ventral flow** is set to $(\omega_x, \omega_y) = (p_{u0}, p_{v0})$. The **time-to-contact** and **slopes** can be retrieved from the first order spatial derivatives at the center of the image $(x, y) = (0, 0)$:

$$\left.\frac{\partial u}{\partial x}\right|_{x=0,y=0} = \omega_x a + \omega_z = p_{u1}, \tag{13}$$

$$\left.\frac{\partial u}{\partial y}\right|_{x=0,y=0} = \omega_x b = p_{u2}, \tag{14}$$

$$\left.\frac{\partial v}{\partial x}\right|_{x=0,y=0} = \omega_y a = p_{v1}, \tag{15}$$

$$\left.\frac{\partial v}{\partial y}\right|_{x=0,y=0} = \omega_y b + \omega_z = p_{v2}, \tag{16}$$

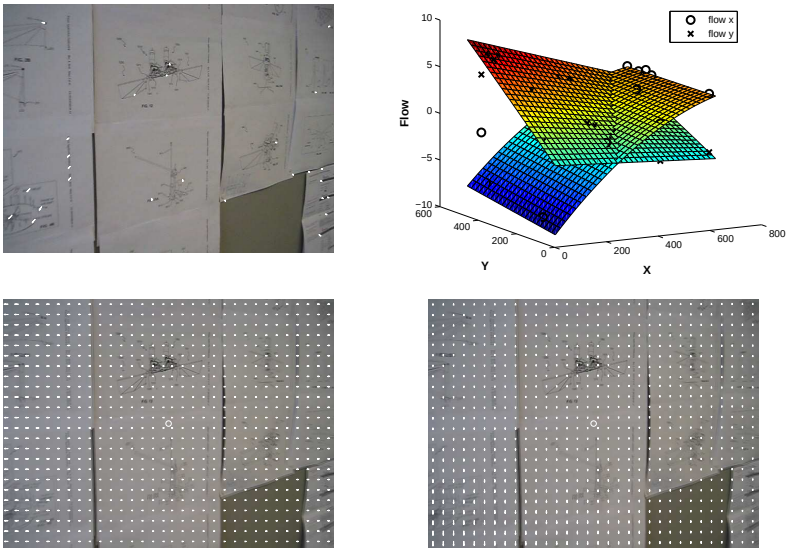

Figure 2: Illustration of the vision process. Top left: the determined optic flow vectors. Top right: quadratic fits for the flow in x-direction (circle markers) and y-direction (cross-markers). Bottom left: estimated flow field in x-direction. Bottom right: estimated flow field in y-direction. The motion of the camera is straight towards the wall, which has an angle of $\sim 45°$ to the camera axis.

The slopes can then be retrieved as $a = p_{v1}/\omega_y$ and $b = p_{u2}/\omega_x$. However, these equations become ill-conditioned and hence sensitive to even the smallest of noise if ω_x or ω_y are small. If there is insufficient motion in X and Y direction, then it may still be possible to estimate the slopes on the basis of the second order derivatives:

$$\frac{\partial u}{\partial x \partial x} = -2a\omega_z = 2p_{u3}, \tag{17}$$

$$\frac{\partial u}{\partial x \partial y} = -b\omega_z = p_{u4}, \tag{18}$$

$$\frac{\partial v}{\partial x \partial y} = -\omega_z a = p_{v4}, \tag{19}$$

$$\frac{\partial v}{\partial y \partial y} = -2b\omega_z = 2p_{v3}, \tag{20}$$

which leads to two formulas for a and two formulas for b. However, all of these formulas depend on ω_z, the relative vertical velocity. When looking at the formulas for the first order spatial derivatives, one will notice that ω_z cannot be determined without knowing a or b - seemingly introducing a chicken-and-egg problem. The solution lies in remembering that we only need these second order derivatives if the ventral flow estimate(s) are small. If, for example, $\omega_x \approx 0$ then:

$$\frac{\partial u}{\partial x} = \omega_x a + \omega_z \approx \omega_z, \tag{21}$$

and, similarly, for $\omega_y \approx 0$:

$$\frac{\partial v}{\partial y} = \omega_y b + \omega_z \approx \omega_z, \tag{22}$$

In summary, when the horizontal flow is not sufficient, the relative velocity ω_z can be determined. In turn, this leads to slope estimates $a = -p_{u3}/\omega_z$, $a = -p_{v4}/\omega_z$, $b = -p_{u4}/\omega_z$, and $b = -p_{v3}/\omega_z$. All these equations become ill-conditioned if ω_z becomes small. This is intuitive, since if there is also little motion in the Z-direction, then no estimates of slope are possible. The remaining question is then how to deal with cases in which an MAV has velocities in multiple directions. For optimal estimation, one should fuse the different a and b estimates on the basis of their certainties and possible prior probability distributions. However, in this preliminary work slopes are determined on the basis of the first-order optic flow derivatives if there is sufficient motion in the X, Y-plane and only on the basis of the second-order optic flow derivatives if there is not.

The time-to-contact τ is determined on the basis of the divergence:

$$D = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{23}$$

, with $\tau = 2/D$. Hence, τ includes the ventral flow and the slopes (see Eqq. 13 and 16). This makes sense, because in the case of pure horizontal motion, the MAV can still intersect the landing surface if it has a slope. However, if one is only interested in the component in the Z-direction, $\tau = 1/\omega_z$ could be used. In the current work the first definition is employed. Please note that in contrast to the slope angle, the time-to-contact can also be determined without knowledge of $\tilde{f}$. Moreover, the time-to-contact is in

principle expressed in frames. Knowledge of the time interval between image frames is necessary to transform it to seconds.

Finally, the **flatness** of the slope is related to how good the optic flow vectors fit with the above-described quadratic model. The RANSAC procedure returns a number of inliers and an error, which can both serve as measures for flatness.

Figure 3: Example images. From left to right: artificial image, images from manual videos (wall, flat scene, structured scene), image from the drone's bottom camera.

2.4 Implementation

A main decision for the implementation of the vision algorithm described above is the choice of optic flow algorithm. We employ a sparse optic flow setup with the method of Shi and Tomasi [21] to find good features for tracking and Lucas-Kanade's pyramid method [18] for optic flow determination. We have made both an implementation in MATLAB, for off-line experiments, and in C++ for online experiments on an MAV. The MATLAB code of the vision algorithm is available at `http://mavlab.lr.tudelft.nl/`, so that the reader can test it on his / her own image sequences. Please note that the open source code involves our own implementation of Lucas-Kanade's optic flow tracking, while we used the implementation of [7] for the experiments. This code gives slightly better results. The code from [7] is openly available, but copyrighted, so we chose not to include it in our code package. However, readers can download it and incorporate it easily into the code.

3 Experiments on Image Sequences

In this section, a preliminary test of the vision algorithm is performed by applying it to various image sequences. The algorithm is tested on three

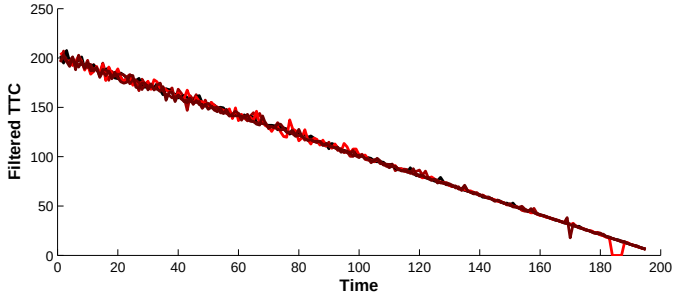

Figure 4: Results of time-to-contact estimation on five image zooms. The ground-truth time-to-contact goes from 200 to 5.

types of image sequences: (1) image zooms, (2) manually made videos, and (3) images from the Parrot AR drone. Figure 3 shows five example images.

The test on artificial image sequences is motivated by the fact that in this setup the ground-truth values are exactly known. The virtual camera used in these experiments has a field-of-view of 50° and generates relatively small 256×256 pixel images. To test the determination of time-to-contact, five image zooms were performed on a flat roof texture. The ground-truth time-to-contact decreases at a constant rate from 200 to 5. The results of the time-to-contact estimation are shown in Figure 4. The estimated time-to-contact is very accurate, even up to 200 frames from contact.

In order to test the slope estimation, a virtual camera moves with respect to the roof texture image, which is placed at various slope angles in the X-direction: $\alpha \in \{0, 15, 30, 45, 60\}°$. Three types of movements are studied: horizontal movement in the X-direction, horizontal movement in the Y-direction, and vertical movement towards the artificial landing surface (Z-direction).

Figure 5 shows the results of the artificial movement sequences. The top left plot shows the slopes estimated during a constant movement in the Y-direction (with $\omega_y \sim 3$). Overall, the estimated slopes are rather close to the ground-truth values, with errors in the order of a few degrees. There are a few outlier cases with errors of up to 15 degrees. The top right plot shows the more difficult case of a constant movement toward the surface, in the Z-direction. Although the estimates rely on second-order derivatives, the slopes are still estimated rather accurately. During the last 10 time steps of the approach, the implemented optic flow algorithm has problems

determining the correct optic flow vectors. This leads to a degradation of the slope estimation. The bottom left plot shows the slopes estimated during a constant movement in the X-direction. Although the results are close to the ground-truth values at the start of the approach, after 40 frames a few of the estimates start to deviate from the ground-truth in the order of tens of degrees. The cause of this phenomenon lies in the fact that the movement brings the virtual camera further from the landing surface, reducing the magnitude of the ventral flow ω_x. The bottom right plot shows ω_x over time during the movement. A smaller ventral flow leads to less accurate slope estimates.

For further testing, three sets of videos have been made of a textured wall. In the first set the camera moves in the Y-direction (first up and then down), in the second set in the X-direction (first left and then right), and in

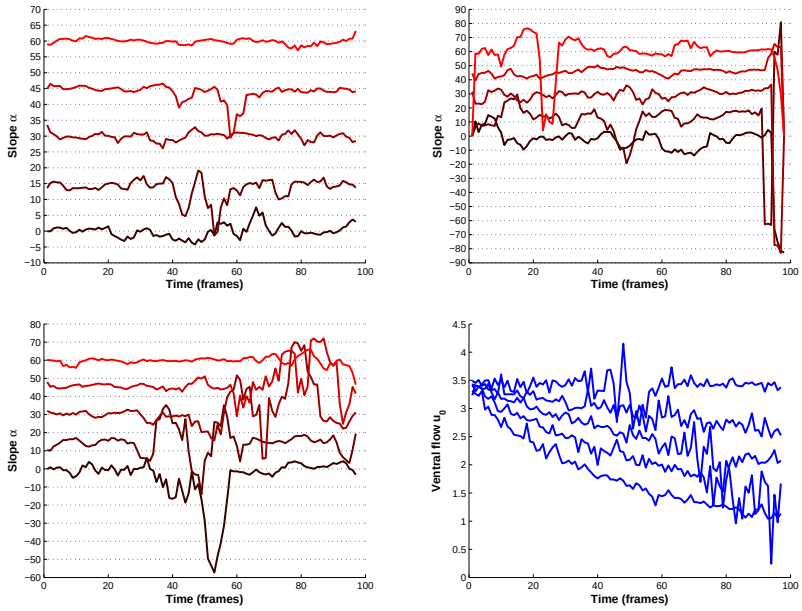

Figure 5: Slope estimation results on artificial movement sequences. The ground-truth values for the slopes α are $\{0, 15, 30, 45, 60\}^\circ$. Top left: slopes estimated during movement in the Y-direction. Top right: slopes estimated during movement in the Z-direction. Bottom left: slopes estimated during movement in the X-direction. Bottom right: ventral flow ω_x during movement in the X-direction.

the third set the camera moves in the Z-direction (in this case toward the wall). Put in a landing context, the first two sets cover horizontal motion, while the third set covers vertical motion toward the landing surface. Each set contains 5 image sequences in which the angle of the wall roughly iterates over the angles $\{0, 15, 30, 45, 60\}$ degrees. Please remark that the camera is moved in-hand, so additional motions and even rotations are present in the images. Since there is no clear ground-truth and the camera is uncalibrated, this experiment mainly serves the goal of verifying that the slope a increases with an increasing angle to the wall (the slope values are not transformed to angles in this case). Figure 3 shows an example image from the sequence at ~ 45 degrees.

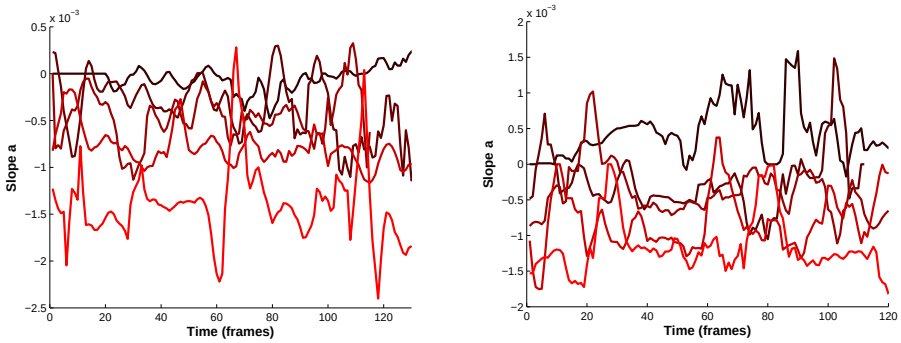

Figure 6: Results of slope estimation on video sequences. Left: motion in Y-direction. Right: motion in X-direction. The plots show the estimated slope a (y-axis) over time (x-axis). The brightness of the lines are determined by the angle with the wall, with the darkest color corresponding to 0 degrees and the brightest color corresponding to 60 degrees.

Figure 6 shows the results of this experiment for motion in the Y-direction (left plot) and X-direction (right plot). The plots show the estimated slope a (y-axis) over time (x-axis). The brightness of the lines are determined by the angle with the wall, with the darkest color corresponding to 0 degrees and the brightest color corresponding to 60 degrees. The lines are somewhat noisy, especially when the motion gets small, e.g., at reversal points of the movements. However, in both plots, a clear ordering can be seen from dark to bright slopes, as desired.

Figure 7 contains the results for motion in the Z-direction, illustrated in the same manner. In this case, the results are much less clear. The lower

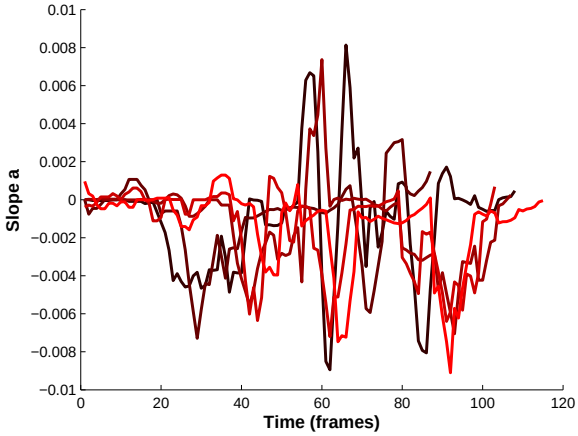

Figure 7: Results of slope estimation on video sequences with the camera moving in Z-direction. The plots show the estimated slope a (y-axis) over time (x-axis). The brightness of the lines are determined by the angle with the wall, with the darkest color corresponding to 0 degrees and the brightest color corresponding to 60 degrees.

degree slopes switch between positive and negative values, while the higher degree slopes are constantly negative. The worse results may be explained by the rotations in the video not accounted for by optic flow derotation. In addition, the second-order spatial derivatives of the optic flow field are more difficult to determine than the first order ones.

Subsequently, the detection of flatness is tested with the help of two sets of each three videos. The first set contains flat surfaces, while the second set contains considerable 3D structure. The videos are made by moving toward the ground surface, i.e., in the Z-direction. The results are shown in Figure 8, which shows the mean absolute errors of the quadratic fits. The black lines illustrate the results for the flat surfaces, while the red lines illustrate the results for the scenes with 3D structure. The videos of flat surfaces have clearly a lower error than the videos of structured scenes. However, placing a threshold is not straightforward, as the error also seems to depend on the time-to-contact. In particular, lower time-to-contacts entail larger optic flow, which leads to larger errors.

Finally, the slope estimation algorithm is run on the 160×120 pixel images of the Parrot AR drone's bottom camera, while it moves down a stairs. The

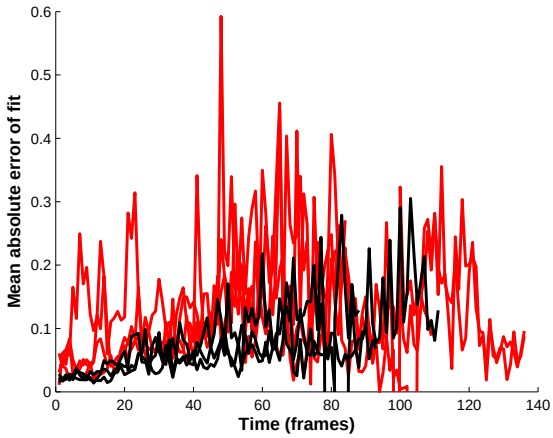

Figure 8: Results for estimation of landing surface flatness. Mean absolute error of the quadratic fits (y-axis) over time (x-axis). The black lines illustrate the results for the flat surfaces, while the red lines illustrate the results for the scenes with 3D structure.

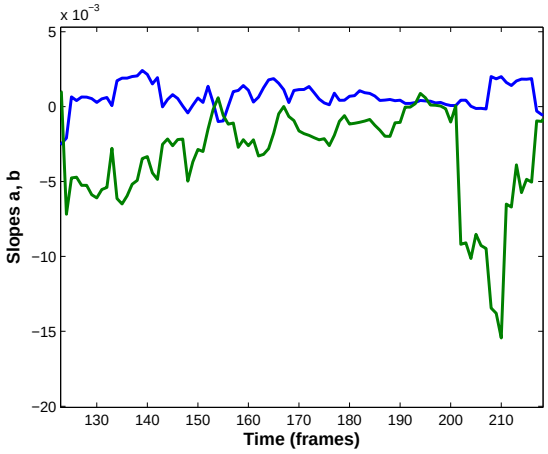

Figure 9: Results for estimation of slope with the Parrot AR drone's bottom camera. The plot shows a (blue) and b (green) over time (x-axis).

rotations of the drone are not accounted for by derotation. Figure 9 shows the estimated slope a in the x-direction (blue) and b in y-direction (green) during a part of the trajectory over the stairs. The y-direction is in the

direction of the stairs. Indeed, while a is reasonably small and switches from positive to negative and back, b is larger and mostly negative, as it should be. Please note that the slopes are not expressed in angles.

In summary, when the camera is calibrated and rotation is accounted for, the vision algorithm is able to retrieve the angle of a landing surface rather accurately. In the case of an uncalibrated camera and in the presence of small rotations, the slope estimates vary clearly with the angle of the wall in the case of movements in the X, Y-plane. The slope angle estimation then depends only on the first order spatial derivatives of the optic flow. A higher ventral flow leads to better slope estimates. The results on videos with motion in the Z-direction give worse results. Finally, the vision algorithm provides a cue for the detection of 3D structure below the MAV.

4 Landing Experiment

The above-described vision algorithm is implemented in TU Delft's *SmartUAV* ground control station, and tested in real-time on a Parrot AR drone. The goal of the experiment is to show that the vision algorithm can work in real-time, real-world conditions.

In order to show that the vision algorithm can be used in a generic way, a staircase was chosen as the test environment. Stairs are challenging, since it is not a flat inclined surface. The setup for the experiment is presented in Figure 10. The experiment started with a forward motion of the drone (in the Y-direction) generated by a small pitch angle. This forward motion was controlled by proportional controller using the feedback of the velocity estimated from the AR drone optic flow measurement. The purpose of moving the drone in forward direction was to generate ventral flow to estimate the slope in real-time. The drone thus travels down the stairway and to a flat ground in the end. The height of the drone was under control of the Parrot firmware, implying that it remained at a fixed height above the steps of the stairs.

A simple landing strategy was used in this preliminary experiment, i.e. when the value of the estimated slope was close to zero $b \in [-0.0005, 0.0005]$, a landing command was given to land the drone immediately. The results of the estimated slope b from the onboard images captured during the flight is shown in Figure 11. It is clearly seen that the value of the slope was always negative when the drone was flying above the stairs. At the instant when

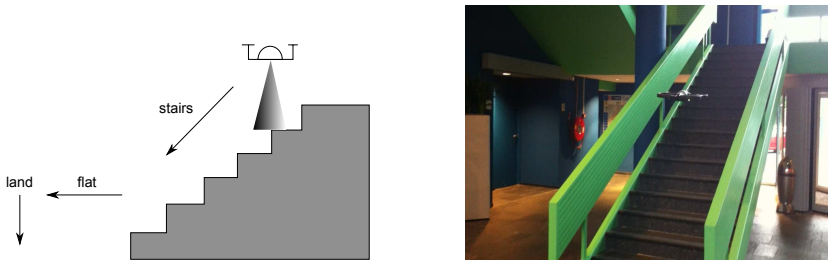

Figure 10: Landing experiment setup

it reached the flat surface, the value of the slope changed its sign and the autoland was activated to bring the MAV to the ground.

Please note that the drone moves forward by pitching forward. The experimental results show that this considerably influences the estimated slope, which was not accounted for in the code. This led to slightly less negative slope values above the stairs and slightly more positive slope values above the flat floor[1]. If a camera calibration is available, the angle of the surface can be estimated in degrees. The estimated attitude of the MAV can then be subtracted from this angle.

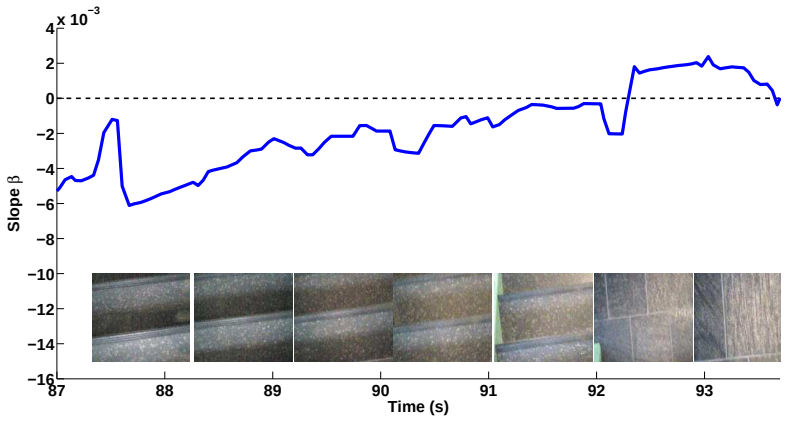

Figure 11: Real-time slope estimation results with the Parrot AR drone's bottom camera. The plot shows the estimated slope b and images taken from the onboard camera over time (x-axis)

[1]A video of the experiment can be found here: `http://www.youtube.com/watch?v=TXIPb1NvUJY`

5 Conclusion

The main conclusion is that the proposed vision algorithm is able to extract ventral flow, time-to-contact, flatness, and slope of the landing surface beneath an MAV. A preliminary landing experiment shows that the algorithm is computationally efficient enough to run in real-time and can discriminate between the stairs and a flat surface. The experiments show that it is important to take into account the pitch angle of the MAV while determining the slope of the surface.

Future work will focus on a further investigation of how optic flow can be used to best estimate the variables of interest. For example, the current maximal likelihood estimation has its limitations, especially if there is a lot of noise or if the assumption is violated that rotation is accounted for. A maximal a posteriori estimate probably would better cope with such situations. In addition, combining slope estimates in a Bayesian fashion could considerably improve the estimates and allow for the exploitation of translation in multiple directions. In order to create such a Bayesian estimation scheme, we need to gain more insight into matters such as the relation between the magnitude of the ventral flow and slope estimation accuracy. Finally, in this article we have not studied the control-part of landing in detail. In future work, a more elaborate landing procedure utilizing the estimated variables will be devised and tested in a more complex environment.

References

[1] M.W. Achtelik, A.G. Bachrach, R. He, S. Prentice, and N. Roy. Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *Unmanned Systems Technology XI, SPIE*, 2009.

[2] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *EMAV, the Netherlands*, 2009.

[3] E. Baird, N. Boeddeker, M.R. Ibbotson, and M.V. Srinivasan. A universal strategy for visually guided landing. *PNAS: Biological Sciences - Neuroscience*, In press.

[4] E. Baird, M.V. Srinivasan, S. Zhang, and A. Cowling. Visual control of flight speed in honeybees. *Journal of Experimental Biology*, 208:3895–3905, 2005.

[5] E. Baird, M.V. Srinivasan, S. Zhang, R. Lamont, and A. Cowling. Visual control of flight speed and height in honeybee. In *Simulation of Adaptive Behavior*, pages 40–51, 2006.

[6] 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.

[7] Jean-Yves Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. description of the algorithm, 2000.

[8] 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 GNC'08*, 2008.

[9] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[10] C. Evangelista, P. Kraft, M. Dacke, J. Reinhard, and M.V. Srinivasan. The moment before touchdown: Landing manoeuvres of the honeybee apis mellifera. *Journal of Experimental Biology*, 213:262–270, 2010.

[11] F. Expert, S. Viollet, and F. Ruffier. Outdoor field performances of insect-based visual motion sensors. *Journal of Field Robotics*, 28(4):529–541, 2011.

[12] 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*, 24:381–395, 1981.

[13] D. Floreano, R. Pericet-Camara, S. Viollet, F. Ruffier, A. Brueckner, R. Leitel, W. Buss, M. Menouni, F. Expert, R. Juston, M.K. Dobrzynski, G. LEplattenier, H.A. Mallot, and N. Franceschini. Miniature curved artificial compound eyes. *Proceedings of the National Academy of Sciences*, 2013.

[14] N. Franceschini, S. Viollet, F. Ruffier, and J. Serres. Neuromimetic robots inspired by insect vision. *Advances in Science and Technology*, 58:127–136, 2008.

[15] B. Hérissé, 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*, pages 1600–1605. IEEE, 2010.

[16] 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*, 35(4):1362–1367, 2012.

[17] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of Royal Society, London B*, 208:385–397, 1980.

[18] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging understanding workshop*, pages 121–130, 1981.

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

[20] G. Sabiron, P. Chavent, T. Raharijaona, P. Fabiani, and F. Ruffier.

[21] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.

[22] Y.M. Song, Y. Xie, V. Malyarchuk, J. Xiao, I. Jung, K.-J. Choi, Z. Liu, H. Park, C. Lu, R.-H. Kim, R. Li, K.B. Crozier, Y. Huang, and J.A. Rogers. Digital cameras with designs inspired by the arthropod eye. *Nature*, 497:95–99.

[23] S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-slambased navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.