A Tailless Flapping Wing MAV Performing Monocular Visual Servoing Tasks

D.A. Olejnik *, B.P. Duisterhof †, M. Karásek, K.Y.W. Scheper, T. van Dijk and G.C.H.E. de Croon
MAVLab, Control and Operations, Faculty of Aerospace Engineering, TU Delft

ABSTRACT

In the field of robotics, a major challenge is achieving high levels of autonomy with small vehicles that have limited mass and power budgets. The main motivation for designing such small vehicles is that, compared to their larger counterparts, they have the potential to be safer, and hence be available and work together in large numbers. One of the key components in micro robotics is efficient software design to optimally utilize the computing power available. This paper describes the computer vision and control algorithms used to achieve autonomous flight with the ∼30-gram tailless flapping wing robot, used to participate in the IMAV 2018 indoor micro air vehicle competition. Several tasks are discussed: line following, and circular gate detection and fly-through. The emphasis throughout this paper is on augmenting traditional techniques with the goal to make these methods work with limited computing power while obtaining robust behaviour.

1 INTRODUCTION

Recently, there has been a growing interest in developing autonomous micro aerial vehicles (MAVs) due to their agility and inherent safety. However, limited on-board processing and sensory information still pose a challenge for the real-time robot operations in a complex environment.

A primary role in the attitude and position determination of MAVs is played by accelerometers, gyroscopes, inertial measurement units (IMU) and global positioning system (GPS). Unfortunately, the mentioned sensors tend to be noisy and drift with time. In indoor environments position information can be also obtained from motion tracking systems like Vicon or OptiTrack. Although these systems are highly accurate, they are not feasible for larger spectrum of applications. As an alternative, an on-board camera can be utilized to complement state estimation.

In many applications, stereo vision system is preferred due to depth measurements that do not suffer from scaling ambiguity. However, a single camera is lighter and requires less power than two cameras. Especially, when it comes to MAVs, lightweight and power efficient solutions are the most desirable.

Monocular visual servoing has a great potential that is currently exploited by many researchers [1, 2, 3, 4]. In [5], desired heading of a drone flying indoors is established based on longitudinal lines. In a similar fashion [6] makes use of line correspondence using images of a corridor and window to calculate the position and heading of a MAV. The paper [7] applies visual servoing to power lines inspection, where guidance of drone relies on extracted information from images with linear features. Alternatively, [2] discusses position control for MAVs using circular landmark. Here, camera pose estimation is based on a geometric approach and derived from an epileptic appearance of a circle in a perspective projection.

In this paper, we present a monocular vision based approach for visual servoing tasks such as line following and precise flight through a set of increasingly small hoops. We propose simple algorithms to perform these flight elements that reduce computational effort and enhance robustness. Our main contribution is an implementation of those strategies on a flapping wing MAV with limited on-board processing and sensory information. Finally, we demonstrate autonomous flight capabilities of the MAV in the indoor competition of the International Micro Air Vehicle Conference and Competition (IMAV2018).

2 SYSTEM OVERVIEW

2.1 The DelFly Nimble

The vehicle used in the competition was the DelFly Nimble [8], the latest flapping wing MAV developed within the DelFly project [9]. Compared to its predecessors, which were stabilized and controlled by a tail with conventional control surfaces [10, 11], the Nimble is a tail-less design. The flapping wings are thus not only used to generate sufficient lift and thrust. The vehicle is, similar to insects, controlled by adjustments of the motion of the individual wings.

The tailless concept has many advantages over tailed designs. It allows for fully controlled hovering flight as well as flight in any direction: up/down, left/right or forward/backward. This opens up new possibilities in autonomous flight of FWMAVs. Unlike the tailed autonomous DelFly versions [12, 13], the vehicle can fly sideways e.g. to align itself with the center of a window before flying through it. It can also stop and turn around, when reaching a dead end,

*D.A.Olejnik@tudelft.nl
†b.p.duisterhof@student.tudelft.nl

which was not possible before as the tailed vehicles needed to maintain a minimal forward velocity to stay airborne.

For the competition, we equipped the Delfly Nimble with the VL53L0X range sensor and a custom built mono-camera system, which uses the same hardware as the stereo vision system of the Delfly Explorer [11]. The VL53L0X is a Time-of-Flight (ToF) laser-ranging module providing absolute distance measurement up to 2m. The sensor weighs only 0.54 gram. The custom-made camera module with STM32F405 processor for onboard vision processing weighs 2g and reaches a clock-speed of 168 MHz with 192 kb of RAM [1]. Together both lightweight sensors allowed us to carry out flight tests with height estimation and shape recognition. Finally, an ESP8266 ESP09 WiFi module was installed to provide a bi-directional datalink. This was invaluable during the testing, as it provided live telemetry and allowed online tuning of the various control parameters.

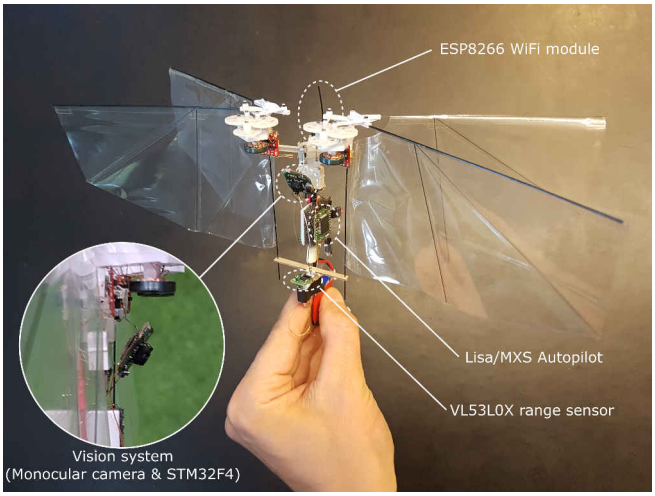


Figure 1: DelFly Nimble tailless flapping wing MAV configured for the IMAV 2019 competition.

The final vehicle configuration, with a total weight of 29.92 g, is in Figure 1. The camera system was mounted via a thin metal strip, which allowed to manually adjust the camera angle according to the task needs. For the line following task, the camera angle was chosen in a trade-off between more information about the future or close to the vehicle. If we point the camera more up, we can look further into the future, but the area close to the DelFly is not visible anymore. For the circle detection task, the camera is placed such that it is looking straight forward. This, in effect, is dependent on the speed at which we fly, as the pitch angle changes with velocity. Because the vehicle was most of the time operated at slow forward flight, the laser range sensor, placed at the bottom of the vehicle, was oriented to point down and slightly forward ($\sim$ 15 degrees) with respect to the vertical body axis.

[1]https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

2.2 Control of a flapping-wing MAV

2.2.1 Attitude stabilization

Tailless flapping wing MAVs are, like multicopters, inherently unstable and require active attitude stabilization. For this, the vehicle was equipped with an open-source STM32F4-based Lisa/MXS autopilot[2] running the open-source Paparazzi UAV autopilot system[3]. The autopilot board was mounted on a soft-mount consisting of PU foam blocks in order to prevent saturation of the on-board 6DOF IMU (MPU 6000) signals. The attitude was stabilized by a standard PD controller with additional low pass filtering, more details can be found in reference [8]. Although no magnetometer was present, the drift of the estimated heading was relatively slow and would typically be just a few degrees over the time needed to complete the competition task. Moreover, in the tasks like line following, the vehicle controls its heading relative to the line direction and an accurate absolute heading was thus not needed.

2.2.2 Height control

Although the autopilot board is equipped with a barometric pressure sensor, this sensor alone proved to be insufficient for precise height control. Even in ideal indoor conditions, the vehicle did oscillate more than +/- 0.5 m from the set point. Thus, for height estimation we have fused the pressure reading with the laser ranger reading using a complementary filter. Nominally, the absolute laser ranger reading was given a high weight. This weight was lowered when very high climb/descend rates were seen by the laser, typically when flying over an obstacle. In such situation, more trust was given to the pressure sensor. Finally, only low-pass filtered pressure based estimate was used when no valid laser measurements were available, e.g. when the floor reflection was insufficient, or when the vehicle got out of the laser sensor range.

The laser ranger worked reliably up to 1.5 m on the multiple floor surfaces where we were testing prior to leaving to the competition in Australia. Unfortunately, as we have noticed at the competition site, the sensor does not work in direct sunlight (high IR light content). And, it also does not work well on tarmac (an unexpected floor material of the indoor competition), which does not reflect enough light back. Thus, we were forced to use wind shades to limit the amount of sunlight and covered the floor with blankets that would reflect the IR light.

In order to maintain a leveled flight we are using a PI control to minimize the error between the desired and measured height. Furthermore, because the demand for throttle level increases as the battery discharges, the altitude controller included also a battery-level dependent feed-forward control

[2]https://wiki.paparazziuav.org/wiki/Lisa/MXS_v1.0
[3]http://wiki.paparazziuav.org/

based on measurements in Figure 2.

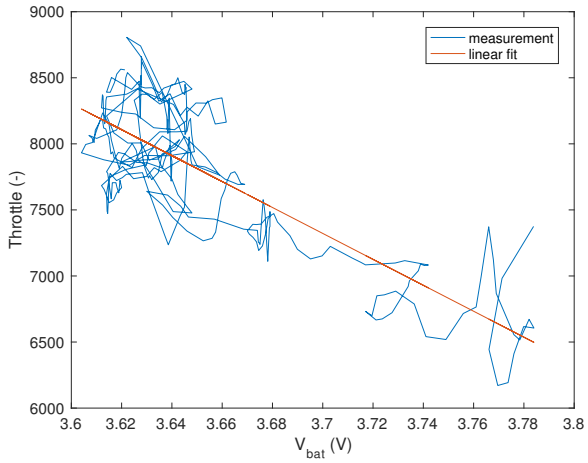


Figure 2: Throttle level at near hover against battery voltage; experiment (blue) and linear fit (red).

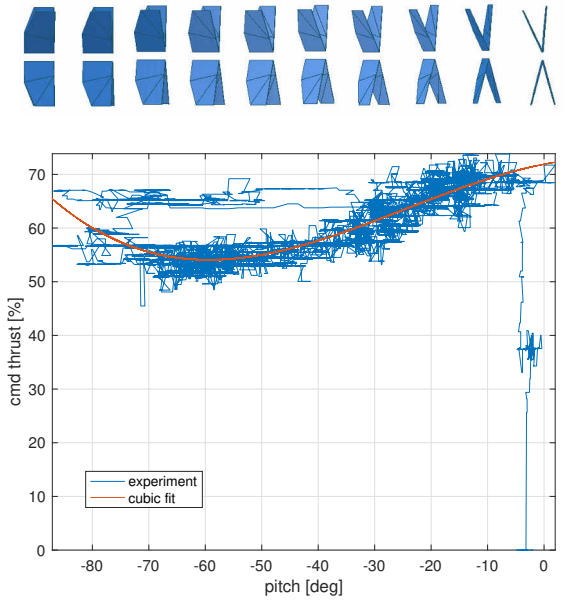


Figure 3: Throttle level against body pitch when flying in the wind tunnel with increasing wind speed and maintaining approximately levelled flight; experiment (blue) and cubic fit (red); the robot's body posture is shown from top view.

Near hover, tail-less flapping wing MAVs are comparable to quadrotors. The vehicle stays airborne thanks to the thrust generated due to flapping motion. Higher thrust, and thus climbing, can be achieved by higher flapping frequency. Flying forward and sideways is achieved by titling the entire MAV, and thus by titling the thrust vector, forward and sideways, respectively.

In forward flight, flapping wing MAVs generate additional lift due to a change in body posture and the oncoming airflow, where the wing surface acts as an airfoil. Thus, higher pitch angles require less throttle in order to achieve the same lift force. These flapping-wing-specific coupling effects were characterized while flying in a wind tunnel [14] (Figure 3) and were accounted for in the feed-forward controller in the form of a cubic fit.

2.3 Competition Tasks

The competitions were focused around topics like aircraft efficiency and innovative designs, light and small MAVs, autonomy and image processing. The indoor mission accommodated a flight through windows, hoops, and following a predetermined flight path. The detailed map of the indoor competition is shown in Figure 4.

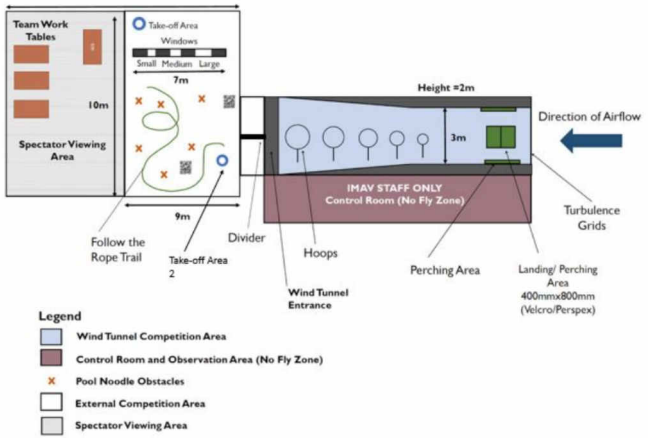


Figure 4: Indoor competition map.[4]

In this paper we will focus on two tasks: line following and precise flight through a set of increasingly small hoops. To complete the first task the MAV had to follow the rope all the way to the end and navigate around obstacles. The rope used during the challenge had high contrast against the floor. The obstacles were represented as green poles fixed in dark blue buckets. The second task was performed in the wind tunnel test section. Five hoops of different sizes, starting with largest, and getting smaller were placed in equal distances. Points were awarded for each hoop flown through. The mission could have been carried out in various wind conditions. Due to limited gust rejection capabilities of our platform the decision was made to fly without wind. The final setup of the indoor competition is shown in Figure 5.

3 LINE FOLLOWING

The goal of this task is to follow a line in an unknown environment. The line can have any shape or curvature which makes accurate line following essential. A test setup was created in TU Delft's Cyberzoo, depicted in Figure 6.

[4] imav2018.org

Figure 5: A screenshot from a video stream of the IMAV's Indoor Competition which shows the DelFly Nimble performing the task of precise flight through a set of increasingly small hoops.

Figure 6: Line Following test setup.

3.1 Perception

This was a best case scenario, as the 'line' is of uniform colour and much thicker than what is expected in the competition.

3.1.1 Segmentation

The fact that the colour of the line is known can be used to our advantage. The least computationally expensive method to determine line position is to do straightforward per-pixel segmentation. However, segmentation of the entire image would be too heavy for the on-board processor and would result in low frame rates [15]. Sub-sampling is used to reduce the computational effort, resulting in a set of pixels that lie on the line. This set of pixels is the starting point for the algorithms considered.

3.1.2 Centroid

As the DelFly Nimble had never been flown autonomously before, the team started off by implementing a straightforward strategy. One of the most straightforward ways to perform the task utilizes the centroid of the line. The coordinates of this centroid in the image frame can then be used as an input for the control system.

Various subsampling methods have been considered, being 1) selecting an upper or lower fraction of the image, 2) selecting evenly spaced or randomly selected rows along the image and 3) using randomly selected pixels over the entire image. All lead to similar results, but the third strategy seems to be the best approach, as every pixel gets the same chance to land in the subset. That is, in theory, the resulting set of points is most representative of the real world.

Even though this algorithm is relatively simple, it already led to promising results. The DelFly Nimble was able to follow the circle fully autonomously. The main shortcoming here was that even though the DelFly would follow the circle, tight corners would cause the system to stop tracking the line. The reason for this is that once a large portion of a turn enters the camera's field of view (FOV), the position of the centroid becomes a less ideal control input. Figure 7 shows a curve in the line as an example, where the vehicle is tempted to steer into the curve due to the c.g. position. This will result in this corner to be 'cut', that is, the vehicle will stop flying directly over the line. Considering the obstacles surrounding the line, accurately following of the line will be essential.

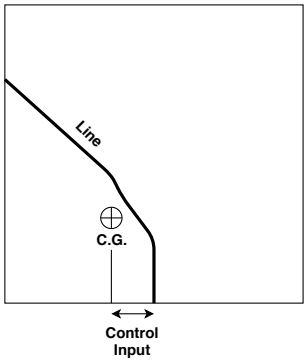
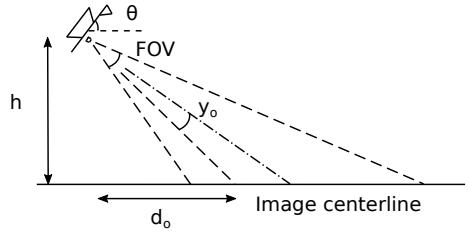


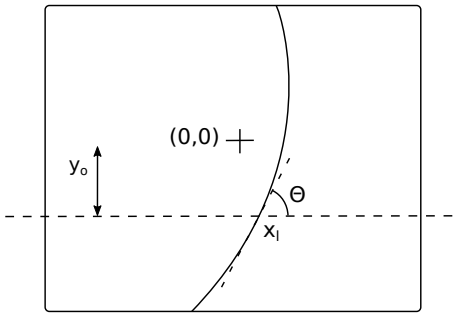
Figure 7: C.G. of the line would trigger the control system to start rotating left, which would cause the corner to be 'cut'. More precise following of the line is required.

3.1.3 Line fit

The centroid-based approach was demonstrated to be a first functional algorithm, while a more accurate algorithm was desired. Until now we have been looking at the centroid only, while more information of the line would be beneficial. Ideally, we would want to know the position and orientation of the line directly beneath the vehicle and translate that to yaw and roll commands. In that way, the vehicle is flying



(a) Side view of DelFly in flight showing the projected image on the ground.



(b) Sample resultant second order line fit in the image plane

Figure 8: Graphical depiction of error terms used for line following control.

above the line at all times and not cutting corners.

Using the points (pixels) generated before, a second order line fit can be done. As the FOVs, attitude and altitude of the camera are known, it is possible to extrapolate the line fit outside of the image.

After having implemented this strategy, it turned out that it was impractical to perform extrapolation. The fitted line would quickly diverge from the line in the real world and an unusable detection was the result. Because of this reason, it was chosen to compute the position and orientation of the line at some point in the lower half of the image. In other words, for a chosen value along the vertical y-axis of the image, we compute the position and orientation of the line. Depending on speed and camera orientation, this point can be moved along the vertical y-axis of the image.

One final improvement is that, for every pixel that is found on the line, the algorithm will search in all 4 directions to find more pixels that meet the colour filter. In this way, more pixels will be found at low cost. This is the strategy that was finally used in the IMAV 2018 indoor competition.

3.2 Control

Figure 8 shows a sample resultant second order line fit as described in the previous section. This figure depicts a DelFly flying at height h and body pitch angle θ. With this, we can define the parameter y_0 as the angular vertical offset from the center of the image plane which, when projected onto the ground will coincide with a target distance d_0 away from the vehicle. Note that a potential camera offset can be added to this computation.

$$y_o = \theta - tan^{-1}\left(\frac{h}{d_o}\right) \tag{1}$$

From this we can compute our lateral offset x_l from the line at some point ahead of the vehicle which we can track.

$$x_l = ay_o^2 + by_o + c \tag{2}$$

This angular attitude error can be projected onto the ground plane to determine the metric lateral offset from the line. This lateral error can then be minimized with a simple PID controller coupled to the vehicle roll angle ϕ_{sp}.

Additionally, we extract the gradient of the line fit (Θ) at this target offset to determine our alignment error.

$$\Theta = \begin{cases} 2ay_o + b, & \text{if } 2ay_o + b \leq 2 \\ 2, & \text{otherwise} \end{cases} \tag{3}$$

With this we can then set our desired heading as

$$\psi_{sp} = \psi + \tilde{\Theta} \tag{4}$$

4 FLIGHT THROUGH AND DETECTION OF CIRCULAR GATES

Algorithms described in this section allow the MAV to perform precise flight through a set of increasingly small hoops. A test setup was created in TU Delft's Cyberzoo, depicted in Figure 9.

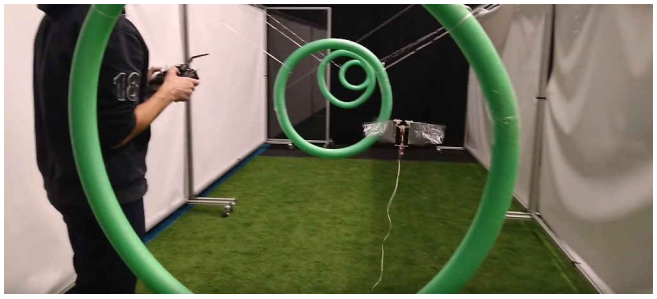


Figure 9: Flight through hoops - test setup.

4.1 Perception

To detect the gates, we find circles in the image using a probabilistic Hough transform based on the bisector between pixel pairs. The Hough transform provides robustness against segmentation errors (for instance by uneven lighting) and against errors in the shape of the gates, while the probabilistic sampling keeps the method computationally lightweight.

Pixels are randomly sampled from the image; YUV color thresholding is then used to test whether these pixels belong

to the gate or the background. Only pixels belonging to the gate are considered for further processing. The thresholds were tuned conservatively, as only a small number of inliers is required to locate the gate in the image while false positives are likely to degrade the result.

We use the bisector between pixel pairs to find the midpoint of the gate. For all pairs of pixels lying on a circular gate, their bisector should intersect the gate's midpoint. This is used as follows: each time a new gate pixel is found, it is paired with all previously found pixels. For each new pair of pixels, the bisector is constructed and all accumulator bins along this line are incremented (Figure 10). This procedure is repeated until a fixed number of pixels is sampled; we sample 20 pixels, leading to a total of 190 bisectors. Once enough pixel pairs have been evaluated, the bin with the highest inlier count is selected as the gate's midpoint (x_g, y_g).

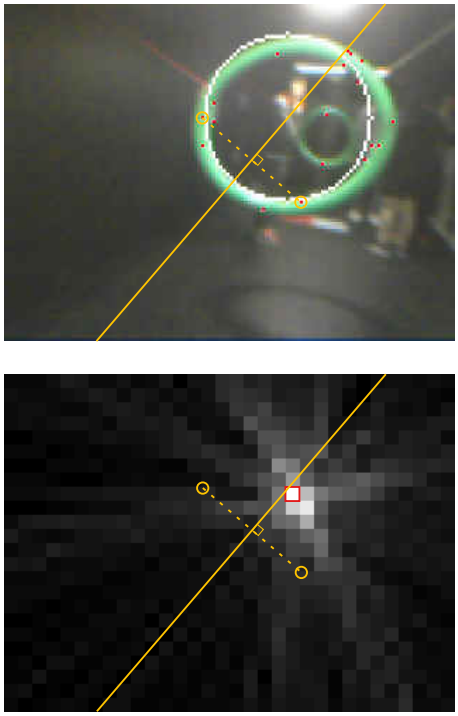


Figure 10: Circular gate detection using the probabilistic Hough transform. *Top:* input image with the sampled pixels highlighted in red and the detected gate shown in white. An example bisector between two sampled pixels is shown in yellow. *Bottom:* the accumulator belonging to this image. Brighter pixels indicate a higher likelihood of the gate's midpoint lying at that position; the bin highlighted in red has the highest inlier count and is selected as the gate's midpoint. The example bisector is also overlaid on the accumulator and is shown to intersect the gate's midpoint.

Estimation of the gate's radius was deliberately left out of the Hough transform to reduce the size of the accumulator. The accumulator was also made four times smaller than the input image to further reduce memory consumption and processing time. Instead of estimating the radius during the Hough transform, we perform this in a later stage where the median distance between the estimated midpoint and a small number (11) of inlier pixels is used to measure the gate's apparent radius or aperture β_g.

4.2 Control

We run a very simple iterative algorithm to localize our position along the tunnel (p) using the predefined location ($\mathbf{D_g}$) and width ($\mathbf{w_g}$) of the gates. When we start the flight attempt, we reinitialize the localization algorithm at an assumed start distance from the first gate. We use two estimates to update our estimated position, the first based on odometry and the second on the perceived location of the gates.

The odometry estimate is obtained using the estimated vehicle air speed generated by a linear transform (c) from the vehicle pitch angle $v_{est} = c\theta$. c was identified as -0.049 by performing a simple line fit through the measured pitch and speed using a motion tracking system as ground truth. Due to the relatively large profile drag of the DelFly at small pitch angles, this linear transform is generally quite accurate. If we assume no external drafts, we can equate the vehicle air speed and its ground speed.

The identified angular aperture of the gate obtained from the perception algorithm described earlier, is used to generate a relative position estimate to the gates. There is, however, an ambiguity as to which gate was identified. To address this, we compute the estimated metric distance to each gate given the size of each gate and the identified angular size. Given our current estimate of the position computed using our odometry estimate ($\tilde{p}(t) = p(t-1) + c\delta t$) and the gate positions from our map, we extract the gate index which result in the smallest estimation error.

$$i = argmin(\mathbf{D} - \mathbf{w_g} tan\frac{\beta_g}{2} - \tilde{p}) \tag{5}$$

Now that we know which gate we are looking at, we use the estimation error to update our position estimate with a filter using a discount factor $\alpha = 0.25$.

$$p(t) = \tilde{p}(t) + (\mathbf{D}(i) - \mathbf{w_g}(i) tan\frac{\beta_g}{2} - \tilde{p}(t)) * \alpha \tag{6}$$

We then set our desired lateral and vertical position for the control system which drive the vehicle roll and thrust respectively. The body pitch is kept constant to have constant forward airspeed.

$$Lat_{sp} = \mathbf{w_g}(i) tan\frac{\beta_g}{2} x_g \tag{7}$$

The height set-point is computed as:

$$h_{sp} = h + \mathbf{w_g}(i) tan\frac{\beta_g}{2} y_g \tag{8}$$

5 CONCLUSION

In this paper we have presented augmented versions of traditional computer vision and control algorithms. With these, we participated in the 2018 International Micro Air Vehicle (IMAV) competition with the DelFly Nimble. First of all, robust flight of the platform was assured, by implementing attitude stabilization and battery level dependent height control. Then, we demonstrated how a second order line fit was performed using a sub-sampling method, which was then used to control roll and yaw. Finally, flight through and detection of circular gates in a tunnel was discussed. A probabilistic Hough transform in conjunction with a position estimate was used to control the vehicle.

With these algorithms we could now, for the first time, run several non-trivial perception tasks on the DelFly Nimble. In a further iteration of the algorithms, increased robustness should be the main focus, to allow for application in more challenging environments. For example, the control algorithms can be altered to work better in outdoor environments and a higher dynamic range in the camera would be desirable.

Even though novel algorithms could further improve mission capabilities, we believe that more computing power is necessary for increasingly autonomous flight with such limited power and weight budgets. Custom SoC (System on Chip) design with accelerators specific to the application have the potential to enhance the autonomous capabilities of flapping wing MAVs.

REFERENCES

[1] Syaril Azrad, Farid Kendoul, and Kenzo Nonami. Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. *Journal of System Design and Dynamics*, 4(2):255–268, 2010.

[2] Daniel Eberli, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Vision based position control for mavs using one single circular landmark. *Journal of Intelligent & Robotic Systems*, 61(1-4):495–512, 2011.

[3] Abel Gawel, Mina Kamel, Tonci Novkovic, Jakob Widauer, Dominik Schindler, Benjamin Pfyffer von Altishofen, Roland Siegwart, and Juan Nieto. Aerial picking and delivery of magnetic objects with mavs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5746–5752. IEEE, 2017.

[4] Justin Thomas, Giuseppe Loianno, Kostas Daniilidis, and Vijay Kumar. Visual servoing of quadrotors for perching by hanging from cylindrical objects. *IEEE robotics and automation letters*, 1(1):57–64, 2015.

[5] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *2011 IEEE International Conference on Robotics and Automation*, pages 5776–5783. IEEE, 2011.

[6] Hanoch Efraim, Amir Shapiro, Moshe Zohar, and Gera Weiss. Position-based visual servoing of a micro-aerial vehicle operating indoor. *Journal of Dynamic Systems, Measurement, and Control*, 141(1):011003, 2019.

[7] Oualid Araar and Nabil Aouf. Visual servoing of a quadrotor uav for autonomous power lines inspection. In *22nd Mediterranean Conference on Control and Automation*, pages 1418–1424. IEEE, 2014.

[8] Matěj Karásek, Florian T Muijres, Christophe De Wagter, Bart DW Remes, and Guido CHE de Croon. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.

[9] GCHE De Croon, M Perçin, BDW Remes, R Ruijsink, and C De Wagter. *The DelFly*. Springer, 2016.

[10] GCHE De Croon, KME De Clercq, Remes Ruijsink, Bart Remes, and Christophe De Wagter. Design, aerodynamics, and vision-based control of the delfly. *International Journal of Micro Air Vehicles*, 1(2):71–97, 2009.

[11] Christophe De Wagter, Sjoerd Tijmons, Bart DW Remes, and Guido CHE de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4982–4987. IEEE, 2014.

[12] Sjoerd Tijmons, Guido CHE de Croon, Bart DW Remes, Christophe De Wagter, and Max Mulder. Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav. *IEEE Transactions on Robotics*, 33(4):858–874, 2017.

[13] Kirk YW Scheper, Matěj Karásek, Christophe De Wagter, Bart DW Remes, and Guido CHE De Croon. First autonomous multi-room exploration with an insect-inspired flapping wing vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[14] Karl Martin Kajak, Matěj Karásek, Qi Ping Chu, and GCHE de Croon. A minimal longitudinal dynamic model of a tailless flapping wing robot for control design. *Bioinspiration & biomimetics*, 2019.

[15] Guido C. H. E. de Croon, C. De Wagter, Bart D. W. Remes, and Rick Ruijsink. Sub-sampling: Real-time vision for micro air vehicles. *Robotics and Autonomous Systems*, 60:167–181, 2012.