Autonomous Flight of a 20-gram Flapping Wing MAV with a 4-gram Onboard Stereo Vision System

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

***Abstract*— Autonomous flight of Flapping Wing Micro Air Vehicles (FWMAVs) is a major challenge in the field of robotics, due to their light weight and the flapping-induced body motions. In this article, we present the first FWMAV with onboard vision processing for autonomous flight in generic environments. In particular, we introduce the DelFly 'Explorer', a 20-gram FWMAV equipped with a 0.98-gram autopilot and a 4.0-gram onboard stereo vision system. We explain the design choices that permit carrying the extended payload, while retaining the DelFly's hover capabilities. In addition, we introduce a novel stereo vision algorithm, LongSeq, designed specifically to cope with the flapping motion and the desire to attain a computational effort tuned to the frame rate. The onboard stereo vision system is illustrated in the context of an obstacle avoidance task in an environment with sparse obstacles.**

I. INTRODUCTION

Autonomous flight of Micro Air Vehicles (MAVs) is a major challenge in the field of robotics. The light weight and small size of MAVs limits the sensors and processing that can be placed onboard, while flying in environments with obstacles requires quick reactions. Impressive results in this area have been obtained with quad rotor MAVs (weighing in the order of 750 g), using sensors ranging from miniaturized laser scanners [1], [14] to RGB-D devices [17], [11] and stereo vision [20], [21].

Flapping Wing Micro Air Vehicles (FWMAVs) form a specific group of MAVs with the advantages of high maneuvrability, quick transition between multiple flight regimes and robustness to impact. Existing FWMAV designs typically have a weight in the order of grams. For example, the extremely small 'Robobee' weighs 0.7 grams (without onboard energy source) [19], and the 'Nano hummingbird' weighs 19 grams [18]. Although most research on FWMAVs focuses on their aerodynamics and design (cf. [22]), several studies have addressed various forms of autonomous flight, with varying sensor / processing configurations.

In the brief overview of related work below, we discern four levels of autonomous flight as studied for FWMAVs: attitude stabilization, visual servoing, height control, and obstacle avoidance. Since obstacle avoidance has not yet been solved, higher level navigation has not yet been studied.

Attitude stabilization is only relevant for tailless FWMAVs, since they are passively unstable. Active attitude stabilization of a tailless FWMAV was first achieved by the Nano hummingbird [18] with onboard sensing and processing. Recently, attitude and 3D-position control has also been achieved with the Robobee by utilizing an external motion tracking system [19]. Visual servoing tasks have been performed in various studies with either offboard sensing and computing [6], [5], [3], [16], onboard sensing and offboard computation [6], [7], [8], [24], or with onboard sensing and computation [2]. In [2] the camera and chip from a 'Wii-mote' were used for detecting and flying toward an infrared light. Height control with external cameras has been achieved by multiple platforms [6], [5], [3], [16]. Vision-based height control in known environments with an onboard camera and offboard processing has been achieved in [6], [7], while height control based on an onboard barometer and processing has been achieved in [9]. Obstacle avoidance has been addressed in [8], [9], [24]. In [8], [9] obstacle avoidance was performed with a single onboard camera, while a laptop determined optic flow and a complementary 'appearance variation cue'. The success of monocular obstacle avoidance remained limited, with a typical flight duration in normal office rooms of around 30 seconds. Recently, stereo vision has been studied for obstacle avoidance with FWMAVs [24], [23], reaching autonomous flights in normal indoor spaces of over 6 minutes. The processing was performed offboard.

For autonomous flight in unknown environments, onboard processing and exteroceptive sensing are essential. Moreover, the exteroceptive sensing needs to provide sufficiently rich information to allow for obstacle avoidance and, later, navigation. None of the above-mentioned studies fulfills these requirements.

In this article, we present the first FWMAV that performs onboard vision processing for autonomous flight in unknown environments. The DelFly 'Explorer' is a 28 cm wing span, 20 gram FWMAV equipped with a 0.98 gram autopilot and a 4.0 gram onboard stereo vision system. The main contributions of this paper are: (1) the light-weight electronics for autopilot and stereo vision system, (2) the design improvements leading to more lift and better handling qualities for making turns with tailed FWMAVs (Section II), and (3) a novel real-time and memory-efficient stereo vision algorithm, named LongSeq, which is robust to the FWMAV's flapping motion (Section III). The onboard stereo vision system is illustrated in the context of an obstacle avoidance task in an environment with sparse obstacles (Section IV). We draw conclusions in Section V.

*All authors are with the Micro Air Vehicle laboratory of the Faculty of Aerospace, Delft University, 2629 HS Delft, The Netherlands c.dewagter@tudelft.nl, microuav@gmail.com.

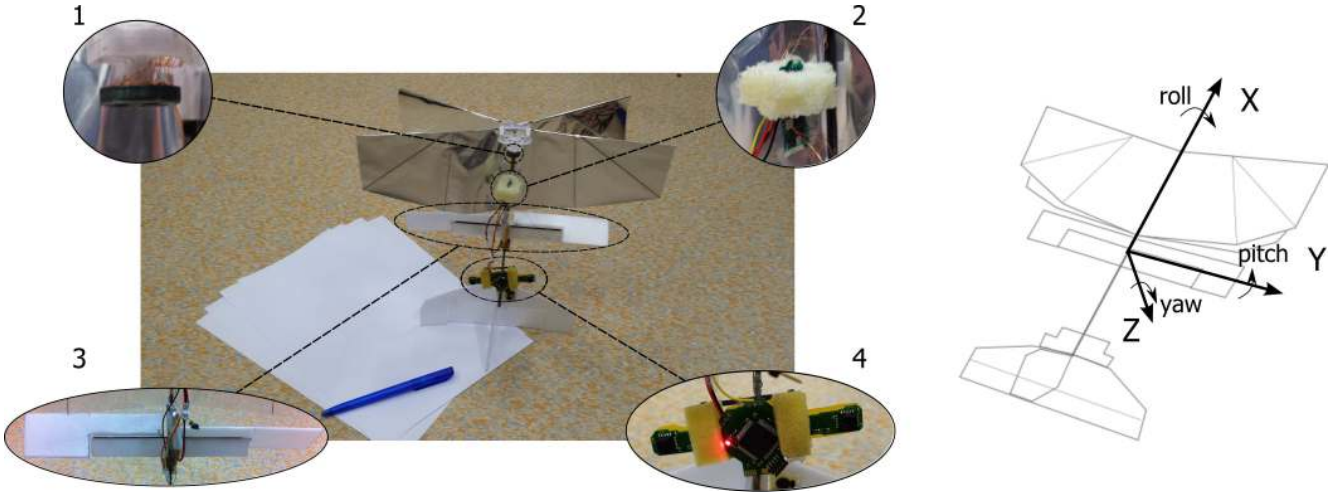

Fig. 1. **Left:** Picture of the *DelFly Explorer*. The four insets show the main changes with respect to the DelFly II: (1) the number of windings in the brushless motors has been reduced to cope with the Explorer's higher weight, (2) an autopilot with a complete IMU, barometer, and an ATmega328P - MLF28 microcrontroller, (3) the DelFly explorer uses ailerons behind the wings instead of a rudder on the tail, and (4) the onboard stereo vision system with STM32F405 processor for onboard vision processing. **Right:** Sketch of DelFly Explorer in flight with the body-axes definition.

II. THE DELFLY 'EXPLORER'

In the course of the DelFly project, many DelFly versions have been created. The project started in 2005 with the *DelFly I*, which weighed 21.00 grams and had a 50 cm wing span. The *DelFly II*, was demonstrated in 2007. It had a 28 cm wing span, and weighed 16.07 grams. The smallest DelFly version, the *DelFly Micro*, was presented in 2008. It weighed 3.07 grams and had a 10 cm wing span. It is important to note that the unique property of the DelFly versions is not their size or weight, but that they can perform free-flight with onboard energy source and camera. The camera allows research on the use of FWMAVs as observation platforms or as autonomous robots.

Despite the weight of an onboard camera and transmitter, the DelFly II has a large flight envelope: it can fly forward at 7 m/s, hover, and even fly backward at -1 m/s. Although the DelFly II was presented to the public in 2007, its design and aerodynamics have been the subject of extensive study, leading to considerable improvements in the handling properties, possible lift, and flight duration. As a consequence of these properties and the miniaturization of electronics, more sensors have been added to it over time. While at first the onboard images were processed offboard both for height control and obstacle avoidance [7], [8], in a more recent study height control was performed by on onboard microcontroller and barometer [9]. However, visual obstacle detection was still performed offboard.

The newest DelFly can carry sufficient payload to carry a 0.98-gram autopilot and a 4.0 gram stereo vision system (cameras and processor), the details of which are given below. Although the payload makes the DelFly heavier, it also allows the autonomous exploration of unknown spaces. Since this sets this DelFly apart from all previous versions, we give it a new name: the *DelFly Explorer*. It has a wing span of 28 cm and a weight of 20 grams.

The DelFly Explorer is shown in Figure 1, with insets showing its four main innovative components. The first inset shows the brushless motor. The number of windings around the colis has been reduced from 37 to 32. This way the ratio of rpm (and hence lift) versus input voltage is increased at the cost of a lower torque. As a result the lift generated at 3.5V is still sufficient to keep the heavier DelFly Explorer in the air. This is in contrast to the old case where it would descend when the voltage dropped below 3.9V. The flight time of the DelFly Explorer is typically around 10 minutes.

The second inset shows a side-view of the autopilot, including an ATmega328P - MLF28 microcontroller, 3-axis accelerometers, gyros, magnetometers, and a barometer. Furthermore, it features two-way telemetry and rpm-monitoring. The autopilot is not necessary to achieve stable flight, as the tail of the DelFly passively stabilizes it during flight. However, the autopilot can serve other purposes, such as performing height control, disturbance rejection or more precise attitude control.

The third inset shows a set of ailerons placed just behind the wings. These ailerons are necessary for making smooth turns, which is essential to autonomous flight. The DelFly II featured a rudder for making turns. Deflection of the rudder first caused the DelFly II to yaw (around the Z-axis - see the right part of Figure 1 for the axes definition), which in turn also resulted in a heading change. However, the yaw rotations during turns rendered computer vision processing during turns problematic. The ailerons of the DelFly Explorer make the DelFly roll (around the X-axis), and since it flies close to up-right, this directly influences the heading without creating any rotations of the camera images.

Finally, the fourth inset shows the stereo vision system in more detail. It has two digital cameras with a baseline of 6.0 centimeter and an STM32F405 processor. Importantly, the flapping motion of FWMAVs introduces deformations in the camera images [4], [8]. Therefore, it is not possible to use subsequently recorded left and right images for

stereo matching [24]. The cameras of the stereo system are synchronized and provide $YUYV$ image streams, and in the current implementation a CPLD merges the streams from both cameras by alternately taking the Y component of the stream from both cameras. This results in a single image stream with the order $Y_l Y_r Y_l Y_r$. The resulting stream contains simultaneously sampled pixels at full camera resolution but without color.

III. STEREO VISION ALGORITHM

For the stereo vision system carried onboard the DelFly, a new stereo vision algorithm was developed that is presented in this section. For autonomous obstacle avoidance of the DelFly, it is required to have real-time processing of the stereo images in combination with sufficient depth quality. Since the stereo system is heavily restricted in terms of processing speed (168 MHz) and memory availability (max. 192 kB RAM), it is important to find the right point on the trade-off between speed and quality.

Among the huge amount of stereo vision algorithms that can be found in literature there are two groups that are not regarded to be suitable for our application. These are the algorithms that perform global optimization, and the algorithms that are based on local matching. The first group is too demanding in terms of power and memory requirements, while the second group provides insufficient quality when dealing with image regions that contain little texture. In between these groups there is another group of algorithms that perform semi-global optimization. Examples of these algorithms are 1-D Dynamic Programming [12] and Semi-Global Matching [15]. These algorithms perform optimization along certain individual directions. The drawback of such an algorithm is that an error somewhere along this optimization line has an effect on the rest of the optimization line. These effects are limited in [15] by optimizing over multiple directions. However, this increases the required amount of processing and memory again.

A. LongSeq

For these reasons a new algorithm is proposed that performs optimization along one image line at a time, where badly matched pixels do not have a degrading effect on the matching quality. For reasons to become clear in the explanation, we call it the *LongSeq* algorithm. The first step in the algorithm is to compute the matching costs $C(x, d)$ of the pixels in one image line by calculating the absolute difference in intensity for a disparity range d_{range} starting from a minimal disparity d_{min}.

$$C(x, d) = |I_l(x) - I_r(x - d)| \tag{1}$$

Then the minimum matching cost $C_{min}(x)$ for each pixel is computed:

$$C_{min}(x) = \min_d C(x, d) \tag{2}$$

Based on these cost measures (matching cost and minimum matching cost), a binary image B is computed for all pixels and disparities of the image line using two thresholds: τ_{cost} and τ_{min}:

$$B(x, d) = \begin{cases} 1 & \text{if } C(x, d) > \tau_{cost} \text{ and } C_{min}(x) < \tau_{min} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The cost threshold τ_{cost} is used to define if a pixel match is good or bad. A matching cost above the threshold indicates a bad match. The minimum cost threshold τ_{min} is used to check if there is at least one disparity value for which the pixel has a good match. $B(x, d)$ will only be nonzero when pixel x has a some good matching candidate, but if that is not the case for the disparity value considered. Pixels that have no good matching candidates are simply ignored. As a result, image B indicates which pixels have better candidates at other disparities. All other pixels are ignored at this stage since they have either no good matching candidate, or they match well at the considered disparity value.

The next step is to find sequences of neighboring pixels in an image line that do not have better matching candidates at other disparities (i.e. $B(x, d) = 0$). The length of this sequence will be used as a measure for matching quality and it is therefore stored in image B. This is done by replacing all zero values by the length of the sequences they belong to. For example, let us consider eight neighboring pixels (50 to 57) in a line for one disparity value, e.g., 7. From Equation 3 the following fictitious values were obtained:

$$B([50\ 57], 7) = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 1]$$

This series of values contains two sequences of zeros; one with length 3 and one with length 2. The zeros in B then are accordingly replaced by these numbers.

$$B([50\ 57], 7) = [1\ 3\ 3\ 3\ 1\ 2\ 2\ 1]$$

An initial disparity map D_{init}^{left} is then computed by selecting from B for all x the disparity value with the highest number (longest sequence):

$$D_{init}^{left}(x) = \max_d B(x, d) \tag{4}$$

The matching cost as described in Equation 1 is defined for matching the left image with the right image. The process is repeated for matching the right image with the left image to obtain D_{init}^{left} and D_{init}^{right}. These disparity maps can now be combined to optimize the result. This is done by mapping the left disparity image to the right disparity image:

$$D_{map}^{left->right}(x - D_{init}^{left}(x)) \leftarrow -D_{init}^{left}(x) \tag{5}$$

The optimal disparity is then found by taking the minimum of the two disparity maps:

$$D_{opt}(x) = \min(D_{map}^{left->right}(x), D_{init}^{right}) \tag{6}$$

This optimization step is required to handle disparity discontinuities. The algorithm is named LongSeq, because

it favors long sequences with constant disparity in an image line. In situations where there is little to no texture, this will slightly bias the result to high-disparity estimates. In the context of obstacle avoidance, this is very sensible: low-texture images often occur close to obstacles and in any case present a danger, since they do not provide information on distances to obstacles ahead.

This method assumes that the images contain only fronto-parallel planes. Furthermore it specifically tries to match image planes with low variation in texture. By sliding these planes over each other, there will be one disparity where the overlap between the planes from the left and right image will reach its maximum. This effect is measured by the length of the sequences, and for this reason the maximum length is selected as the best match.

The proposed method shows some similarities with plane sweeping algorithms [13] in that it tries to match an image plane for a certain orientation. However, in the proposed method only fronto-parallel planes are considered for computational reasons. Moreover, in contrast to [13], LongSeq searches the largest line section meeting this assumption.

B. Subsampling

In the interest of computational efficiency, typical stereo vision steps such as undistortion and image rectification are skipped. Without these steps, LongSeq takes around 90 ms of processing on the STM32F405 on a full image of 128×96 pixels. Hence, it runs at ~ 11 Hz. For many applications of the stereo vision system 11 Hz can be sufficient. However, for some applications, such as obstacle avoidance or flying through a window, a higher processing frequency may be desired. The same goes if one wants to perform additional vision tasks besides stereo vision.

If the interest is not in dense 3D scene reconstruction, but some type of aggregate disparity values are used (as in [24], [23]), then *sub-sampling* can be applied. Sub-sampling typically leads to a considerable gain in computational efficiency at a low cost in accuracy [10]. Since LongSeq is line-based, a natural way of sub-sampling is to process fewer lines. The use of sub-sampling with the stereo vision algorithm will be tested in an application of sparse obstacle avoidance.

IV. APPLICATION TO SPARSE OBSTACLE AVOIDANCE

We apply the DelFly Explorer to a sparse obstacle avoidance task. In the context of this task, we show the results of the stereo vision processing, also when combined with sub-sampling. Avoidance of sparse obstacles is rather straightforward, as is the employed control strategy. The main goal here is to show that the stereo vision system works in real-time and can cope with the FWMAV's flapping motion.

Specifically, the task of the FWMAV derives from the indoor competition of the IMAV 2013[1], which took place on September 19, 2013. The FWMAV had to take off autonomously and fly through a sparse obstacle field, keeping its heading. The obstacles are tall, brightly colored poles. Below, we first discuss the stereo vision results (Subsection IV-A), then explain the control algorithms involved in the experiment (Subsection IV-B), and finally show the results of the experiment (Subsection IV-C).

[1] http://www.imav2013.org/

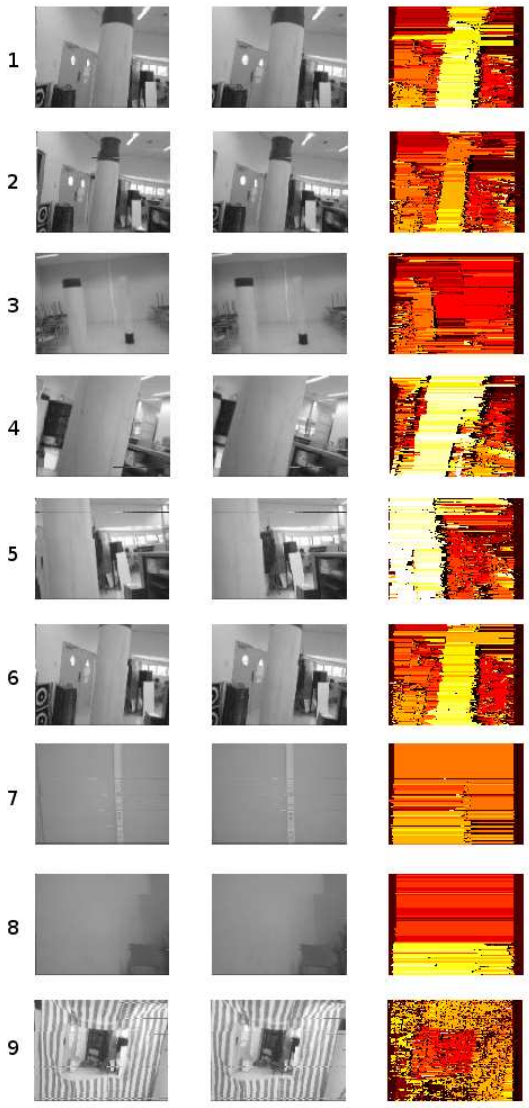

Fig. 2. Nine examples of the stereo vision processing. The columns show from left to right: the left image, the right image, and the disparity image produced by the proposed stereo vision algorithm LongSeq. The disparity images are color coded from low-disparity (dark) to high-disparity (bright).

A. Stereo vision results

The stereo vision system onboard the DelFly Explorer does not yet have any wireless connection for sending images during flight. Therefore, we show results of the stereo system in-hand, with the images sent via a serial connection. Figure 2 shows nine examples of stereo vision images and their corresponding disparity maps. The left column shows the left images, the center column the right images, and the right column shows the disparity maps, in the interval $[0, 10]$ (bad pixels are also set to 0). Please remark that even though

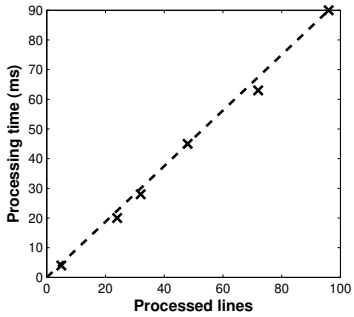

Fig. 3. Processed image lines vs. processing time onboard the DelFly.

the camera is held in hand, the images already have motion deformations and blur. The top six examples show results for detecting poles as is the interest of our application, while the bottom three examples show results in different situations.

The line-wise matching strategy of the proposed algorithm can be clearly seen in the images by the striping effects. By observing the detected poles it can be noted that texture poor areas tend to have the same disparity as the poles. This effect might be reduced in some cases by using more complex algorithms that perform optimization in more directions. For the task of avoiding poles this effect is not a real issue, since the pole will be avoided anyway. In general the background will be assigned the same disparity as the pole and not the other way around. Exceptions occur in situations where the contrast between the pole and the background is very low. The first six examples in Figure 2 show that the presence of the poles is clearly indicated.

The effect of the pixel-based matching cost is illustrated by example 9 of Figure 2. The dense variation in contrast in combination with the low resolution images results in many small sequences and a large variation of disparity values. This effect might be reduced by using windows for calculating the matching cost but this increases computational load as well as memory requirements.

The results from example 7 and 8 in Figure 2 are far from perfect, but the results are useful for our application. Example 7 shows that the wall is fairly close to our camera, even though the structure in the middle is the only feature that provides sufficient texture. In the case of example 8, the algorithm is able to indicate that the bottom part of the images contain obstacles at at smaller range compared to the rest of the image.

The control algorithm explained in the next subsection bases its decisions on the number of pixels with a disparity higher than 5 in the left and the right part of the image. This implies that the detailed disparity maps are aggregated into only two values. Hence, it makes sense to apply subsampling for achieving higher processing frequencies. Figure 3 shows the number of processed image lines vs. the processing times as measured on the STM32F405 (crosses). As to be expected, this relation is roughly linear (dashed line). In order to process at frame rate, one can sample 32 image lines (one third of the image).

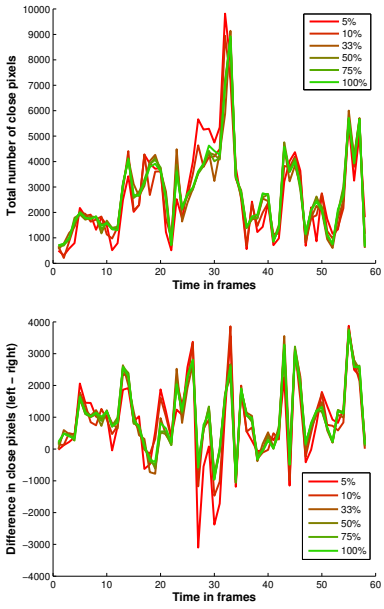

Fig. 4. Effect of subsampling on the aggregate values used by the obstacle avoidance control algorithm. Top: total number of pixels with a disparity higher than 5 (close pixels). Bottom: difference between the number of close pixels in the left and right part of the image. The results are shown for various subsampling ratios, ranging from 5% (red) to 100% (green).

Figure 4 shows the effects of sub-sampling on the estimated number of pixels with a disparity larger than 5 (top) and on the difference between the left and the right image (bottom). As can be seen in the figure, all sampling ratios follow the trend of the case of full sampling (100%) - albeit with a variation that increases with a decreasing sampling ratio. Surprisingly, this is even valid for a low sampling ratio of 5% (4 image lines out of 96 in our implementation).

B. *Flight Control Algorithms*

In this subsection, we discuss the control algorithms used for take-off, height control, and obstacle avoidance. Take-off is performed with open-loop control. Before the control sequence starts, the barometer measurement at that moment is taken as a reference for a height of 0 m. The sequence starts by setting the flap frequency above the trim setting, which results in a steep climb. After that the flap frequency is reduced to a trim value (for trimmed horizontal flight) after which closed loop control is performed on the height using the barometer feedback.

Obstacle avoidance is performed on the basis of the LongSeq's stereo vision processing. First, the number of pixels with disparity larger than 5 are determined in the left and right part of the image. If the total number of such pixels is lower than the empirically set threshold of 300, the DelFly will continue to fly straight. Else it will turn toward the side that has fewer such pixels, with a fixed aileron deflection. Sub-sampling is applied with 32 image lines, so that the processing matches the frame rate of the digital cameras.

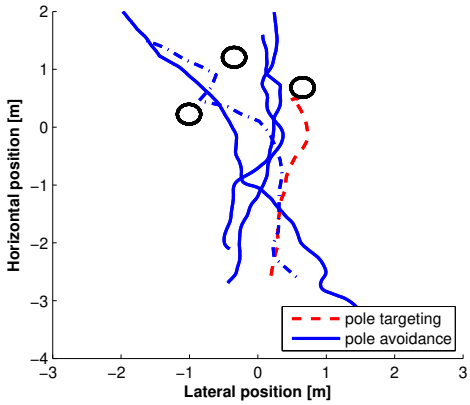

Fig. 5. Flight trajectories of the DelFly Explorer in an area with three round vertical poles (black circles), shown from above. The obstacle avoidance trajectories are shown in blue. Three tracks show successful trials (solid blue). One track shows a trial where the Delfly hit two poles with its wings (dash-dotted blue). Finally, one trajectory is shown where the control input was inversed in order to fly toward poles (red dashed).

C. Experiment

The DelFly Explorer can successfully take off and fly through an obstacle field. Figure 5 shows the flight trajectories from a test with four trials (blue lines). In three out of four trials, the DelFly passes through the field without touching any obstacle. The trial with the dash-dotted line shows a track where the DelFly passed through the obstacle field, but touched two poles with its wings, resulting in sharp turns. In order to show that the vision actively determines the DelFly's heading, we include a trial with a gain that inverts the avoidance reactions (red dashed trajectory). As a result, the DelFly targets the pole instead of avoiding it.

D. More difficult environments

In order to test the DelFly Explorer in more difficult environments, the avoidance algorithm proposed in [24] was also implemented. This resulted in autonomous obstacle avoidance with flight times up to nine minutes in different environments. Videos of these tests and from the tests described in this paper can be found online[2]. Note that the method from [24] is able to handle other important situations such as flying toward a straight wall or a corner.

V. CONCLUSIONS

We have demonstrated the first light-weight flapping wing MAV flying autonomously with onboard stereo vision processing. Having the stereo vision processing onboard has been made possible by: (1) the light-weight electronics for autopilot and stereo vision system, (2) the design improvements regarding the motor and ailerons, and (3) the development of a robust, computation and memory efficient, line-based stereo vision algorithm, named LongSeq. In particular, the quality of the disparity maps created by the stereo vision algorithm shows that it copes well with low visual texture (typical for indoor environments) and image deteriorations such as blur. The computational efficiency is enhanced with the help of sub-sampling, at a negligible cost in accuracy. The functioning of the system in the presence of flapping motion has been illustrated with an application to a sparse obstacle avoidance task, including autonomous take-off and height control.

[2] http://www.delfly.nl

REFERENCES

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

[2] S.S. Baek, F. Garcia Bermudez, and R. Fearing. Flight control for target seeking by 13 gram ornithopter. In *IEEE/RSJ Int Conf on Intelligent Robots and Systems.*, 2011.

[3] S.S. Baek and R.S. Fearing. Flight forces and altitude regulation of 12 gram i-bird. In *IEEE RAS and EMBS Int Conf on Biomedical Robotics and Biomechatronics (BioRob)*, pages 454–460, 2010.

[4] F. Garcia Bermudez and R. Fearing. Optical flow on a flapping wing robot. In *IROS 2009*, pages 5027–5032, 2009.

[5] Cheng-Lin Chen and Fu-Yuen Hsiao. Attitude acquisition using stereo-vision methodology. In *IASTED Conference*, 2009.

[6] G.C.H.E. de Croon, K.M.E. de Clerq, R. Ruijsink, B. Remes, and C. de Wagter. Design, aerodynamics, and vision-based control of the delfly. *International Journal on Micro Air Vehicles*, 1(2):71 – 97, 2009.

[7] G.C.H.E. de Croon, C. de Wagter, B.D.W. Remes, and R. Ruijsink. Random sampling for indoor flight. In *International Micro Air Vehicle conference, Braunschweig, Germany (2010)*, 2010.

[8] G.C.H.E. de Croon, E. de Weerdt, C. de Wagter, B.D.W. Remes, and R. Ruijsink. The appearance variation cue for obstacle avoidance. *IEEE Transactions on Robotics*, 28(2):529–534, 2012.

[9] G.C.H.E. de Croon, M.A. Groen, C. De Wagter, B.D.W. Remes, R. Ruijsink, and B.W. van Oudheusden. Design, aerodynamics, and autonomy of the delfly. *Bioinspiration and Biomimetics*, 7(2), 2012.

[10] G.C.H.E. de Croon, C. De Wagter, B.D.W. Remes, and R. Ruijsink. Sub-sampling: real-time vision for micro air vehicles. *Robotics and Autonomous Systems*, 60(2):167–181.

[11] M.F. Fallon, H. Johannsson, and J.J. Leonard. Efficient scene simulation for robust monte carlo localization using an rgb-d camera. pages 1663–1670, 2012.

[12] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering, and A. Schmitt. Real-time stereo by using dynamic programming. In *Computer Vision and Pattern Recognition Workshop(CVPRW)*, pages 29–29, 2004.

[13] D. Gallup, J.M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[14] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *(ICRA 2009), Kobe, Japan*, 2009.

[15] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814, 2005.

[16] F.Y. Hsiao, H.K. Hsu, C.L. Chen, L.J. Yang, and J. F. Shen. Using stereo vision to acquire the flight information of flapping-wing mavs. *Journal of Applied Science and Engineering*, 15(3):213–226, 2012.

[17] A.S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *ISRR*, 2011.

[18] M. Keennon, K. Klingebiel, H. Won, and A. Andriukov. Development of the nano hummingbird: A tailless flapping wing micro air vehicle. In *50th AIAA Aerospace Science Meeting*, pages 6–12, 2012.

[19] K.Y. Ma, P. Chirarattananon, S.B. Fuller, and R.J. Wood. Controlled flight of a biologically inspired, insect-scale robot. *Science*, 340(6132):603–607, 2013.

[20] K. Schmid, M. Suppa, and D. Burschka. Towards autonomous mav exploration in cluttered indoor and outdoor environments. In *RSS*, 2013.

[21] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *RSS*, 2013.

[22] W. Shyy, H. Aono, C. Kang, and H. Liu. *An Introduction to Flapping Wing Aerodynamics*. Cambridge University Press, 2013.

[23] S. Tijmons, G.C.H.E. de Croon, B.D.W. Remes, C. De Wagter, and R. Ruijsink. Obstacle avoidance by stereo vision on flapping wing mavs. *Submitted.*

[24] S. Tijmons, G.C.H.E. de Croon, B.D.W. Remes, C. De Wagter, R. Ruijsink, E-J. Van Kampen, and Q. Chu. Stereo vision based obstacle avoidance on flapping wing mavs. In *EuroGNC*, 2013.