First autonomous multi-room exploration with an insect-inspired flapping wing vehicle

Scheper, Kirk; Karasek, Matej; de Wagter, Christophe; Remes, Bart; de Croon, Guido

DOI
10.1109/ICRA.2018.8460702

Publication date
2018

Document Version
Accepted author manuscript

Published in
International Conference on Robotics and Automation

First autonomous multi-room exploration with an insect-inspired flapping wing vehicle

Kirk Y.W. Scheper, Matěj Karásek, Christophe De Wagter,
Bart D.W Remes and Guido C.H.E de Croon*

***Abstract*— One of the emerging tasks for Micro Air Vehicles (MAVs) is autonomous indoor navigation. While commonly employed platforms for such tasks are micro-quadrotors, insect-inspired flapping wing MAVs can offer many advantages, such as being inherently safe due to their low inertia, reciprocating wings bouncing of objects or potentially lower noise levels compared to rotary wings. Here, we present the first flapping wing MAV to perform an autonomous multi-room exploration task. Equipped with an on-board autopilot and a 4 g stereo vision system, the DelFly Explorer succeeded in combining the two most common tasks of an autonomous indoor exploration mission: room exploration and door passage. During the room exploration, the vehicle uses stereo-vision based droplet algorithm to avoid and navigate along the walls and obstacles. Simultaneously, it is running a newly developed monocular color based Snake-gate algorithm to locate doors. A successful detection triggers the heading-based door passage algorithm. In the real-world test, the vehicle could successfully navigate, multiple times in a row, between two rooms separated by a corridor, demonstrating the potential of flapping wing vehicles for autonomous exploration tasks.**

I. INTRODUCTION

Autonomous indoor flight is a major challenge in robotics. Since the Global Positioning System (GPS) is not available in most indoor environments, state estimation and navigation have to rely on other means. To ensure robust behavior in a wide range of environmental conditions this should preferably be done using on-board sensors and processing.

Typically, quadrotors in the weight range of 0.5 - 1.5 kg are employed for autonomous indoor flight. The main reasons for this are that (i) they are well-known, widely available platforms, and (ii) they are able to carry the sensor and processing package used for autonomous flight. Using quadrotors of the given weights in real-world environments will have a significant, undesired impact on these environments. Most importantly, the weight and possible speed of the quadrotors means that they can potentially harm humans that may be present. This will either lead to humans having to be absent or lead to increased safety requirements like having to wear helmets and safety goggles. Moreover, these quadrotors have a significant aerodynamic impact on the environment. For example, they will blow papers off a desk in an office environment, and potentially damage crop in a greenhouse environment. Finally, the sizes of these quadrotors are considerable given the many narrow indoor spaces.

*All authors are with Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands g.c.h.e.decroon@tudelft.nl

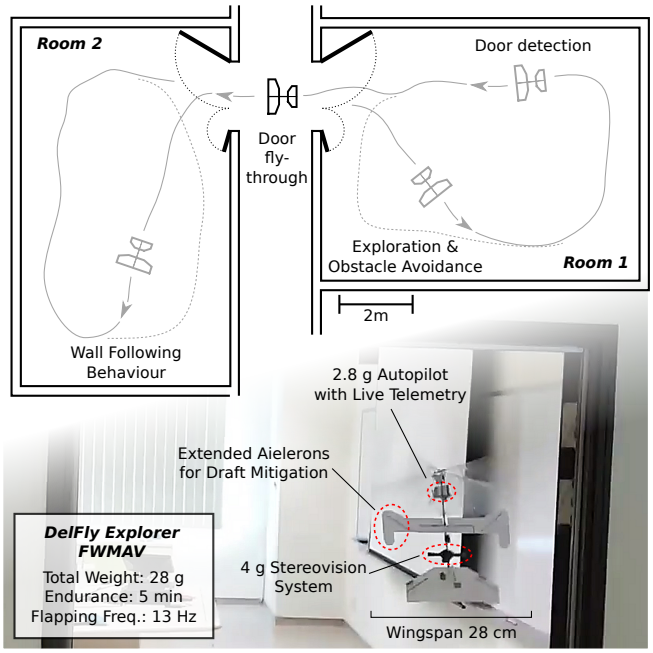

Fig. 1. An autonomous multi-room exploration was achieved, in a real world environment, with the DelFly Explorer flapping wing Micro Air Vehicle carrying a lightweight stereo vision system (bottom). The vehicle could repeatedly navigate between two rooms while exploring them and detecting and passing through the open doors (sketch in the top). The videos from the tests can be found at: https://www.youtube.com/playlist?list=PL_KSX9GOn2P9rFUJE04KQYm1beOP-IVH2

For many autonomous indoor flight tasks, it is highly desirable to use much smaller, lighter, and hence safer drones. To this end, smaller quadrotors can be used [1], [2]. However, flapping wing robots may on the long term be much more promising for indoor flight. A major reason for this is that they use different, unsteady aerodynamic principles for achieving flight [3], [4]. This way of producing lift and thrust scales well as we get smaller and Reynolds numbers are lower. Flapping wings allow for a broad flight envelope, from hover to fast forward flight, where for higher speeds the wings provide additional lift (similar to fixed wing drones) and hence increased energy efficiency. In addition, flapping wing robots naturally handle soft collisions quite well. The flapping wings have zero velocity at the end of each stroke, which means little to no risk for the objects or persons a flapping wing may collide with. Finally, flapping wing robots have a more natural appearance and sound than quadrotors, making them easier to accept for humans sharing

the environment.

With the benefits of flapping winged robots also come some challenges. First, their design is difficult and actually still a field of research in itself [5], [6], [7], [8], [9], [10]. Second, most small flapping wing robots are so light, that common approaches to autonomous flight cannot be employed. Either the sensors are too heavy, such as when using miniature laser scanners [11], [12], or the amount of processing required is too high [13], [14]. As a consequence, most research on flapping wing robots has shown quite limited autonomous flight capabilities. Research on flapping wing robots that steer with their wings is mainly still focusing on achieving attitude and altitude control [5], [15], [16]. The research on flapping wing robots that steer with an airplane-like tail has gone somewhat further in terms of autonomous flight capabilities. For instance, there has been work on target seeking [17], [18], height control [19], [20], window fly-through [21], [22] and obstacle avoidance [19]. Perhaps the most advanced autonomous flapping wing robot is the *DelFly Explorer* [23], [24], which has a 4 g on-board stereo system with which it can avoid obstacles in generic environments. However, it just avoids obstacles and can only explore an environment by chance. For truly flying around in an indoor environment, the robot will have to not only avoid obstacles, but also fly through narrow indoor passages such as doors.

In this article, we present the first experiments in which a flapping wing robot performs a multi-room exploration task. Achieving this task required contributions in terms of both hardware and software, with an emphasis on efficient vision and control algorithms. Please note that the focus here is on reliably flying around in a room and passing through doors, and that high-level navigation (deciding which way to go when passing a door for instance) is out of the scope of this article. Our work is closest to [21], in which a ground robot observes a flapping wing robot and a window and performs all sensing and processing - sending commands to the flapping wing robot. In comparison, we do all tasks on-board of the flapping wing robot. We run both stereo vision and door detection in real-time on-board of an STM32F04 with a 168 MHz processor and 192 kB of memory. The approach followed in this article is very different from our previous work in [22]. One of the major differences is that we now present more efficient, robust vision algorithms for door detection in order to perform tests in a real-world environment. Finally, compared to [24], we are able to pass through quite narrow door passages, providing a new essential building block for full-fledged indoor navigation.

The remainder of the article is organized as follows. Sect. II describes the augmented DelFly Explorer flight platform and the stereo camera system followed by the task description in Sect. III. The room exploration method and the door detection algorithms are described in Sect. IV. Next, the control techniques used to perform high performance navigation to fly through the door is detailed in Sect. V. Finally, Sect. VI shows the results of various test campaigns as well as the final flights performed in a real world indoor test environment.

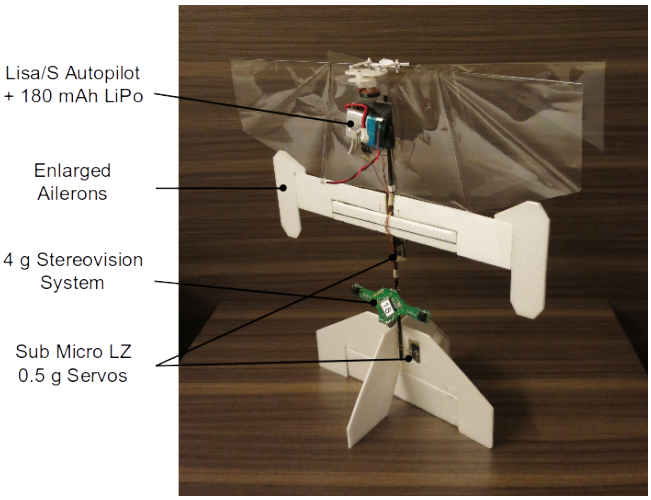

Fig. 2. DelFly Explorer with extended ailerons used to increase control effectiveness of theses yaw actuators.

II. DELFLY EXPLORER

The vehicle used for the test described here is a modified, slightly heavier version of the DelFly Explorer ornithopter [23]. The MAV has two wings arranged in a cross configuration and flapping in counter phase, which, apart from a lift enhancement due to the clap-and-peel effect, helps to minimize the body rocking as the drag and inertial forces of the individual wings cancel out. Further more, it is equipped with a tail for inherent stability, which helps to further damp out any disturbances. As a result, motion blur in the on-board cameras is significantly limited, which makes this flapping wing platform well suited for vision-based autonomy.

The vehicle is controlled via an elevator located on the tail and a pair of ailerons placed just behind the trailing edge of the wings. For the current tests, the aileron surface, interacting with the induced airflow of the flapping wings, was increased near the tips for even higher turning control effectiveness, which was needed when flying in rooms with strong air-conditioning. This augmentation was critical to rejecting external perturbations and maintain accurate heading control facilitating the high level navigation performance. The body pitch and yaw were controlled in closed loop using a PID controller, in order to resist disturbances and maintain the trim speed at around 0.7 m/s. This speed gave the vehicle sufficient resilience against typical drafts experienced indoors while providing enough time to react to and steer away from obstacles detected by its cameras. High derivative gains were used in the yaw loop to maintain straight flight and a feed-forward term was added for a quicker steering response.

The 28 cm wings flapping at around 14 Hz provide enough thrust to carry a 2.8 g Lisa/S autopilot board with 9DOF IMU and live telemetry [25] and a 4 g stereo-vision system [23], bringing the total take off weight to 27 g. The flight time of the whole system, powered by a single cell 180 mAh LiPo battery, is around 6 minutes. The stereo camera

is equipped with a STM32F405 ARM microprocessor which has an operating frequency of 168 MHz and 192 kB of RAM along with two TCM8230MD CMOS cameras separated by 6 cm. These cameras have a 1/6 inch optical format with a lens which results in a field of view of 57.4° in horizontal and 44.5° in the vertical plane. This sensor can output 10 bit color images with a resolution of 640 × 480 pixels at a maximum rate of 30 Hz. Due to memory and processing restrictions of this small platform, we reduce the size of the output to 8 bit, 128 × 96 pixel images at 26 Hz. When in stereo mode, these the images are grayscale only.

We use a computationally efficient version of the well known Sum of Absolute Differences (SAD) window matching stereo vision algorithm described in described in detail in [24]. This algorithm produces a sparse map of distances of features in the environment. This map has been effectively used to identify obstacles in the environment facilitating autonomous flight of MAVs.

III. TEST SCENARIO

The long term goal of our project is to develop a lightweight MAV for autonomous exploration of unknown buildings. While such a mission is very complex and will involve many diverse tasks, there are two tasks the MAV will have to perform repeatedly: exploration of spaces that may include obstacles, such as rooms or halls, and flight through openings, such as windows or doors. We have come up with a test scenario that combines these two most important (low level) tasks, while for now limiting the amount of other tasks such as (high level) navigation.

At the beginning the MAV is located in room 1. It needs to find the door leading to room 2, enter room 2 through the door, explore room 2, find the door leading back to room 1 and return to room 1 through the door. This mission can be repeated multiple times to test the robustness of the autonomous system.

Because of the layout of the building, where we carried out the real world tests (see Fig. 1), an additional corridor was located in between room 1 and 2, with the door of the room 1 facing the door of the room 2. As mentioned earlier, (high level) navigation was not in the scope of the current experiments. Thus, the MAV was supposed only to cross the corridor and enter the other room. Nevertheless, the corridor was a source of disturbances in the form of wind drafts (it lead to other parts of the building with different temperature), which the MAV had to face. Please note that although for this task the thrust and therefore the altitude of the DelFly was controlled by a human pilot via a hand held Radio Controller, the attitude and lateral control of the DelFly was performed on-board the vehicle with no input from an external source.

IV. EXPLORATION AND DOOR DETECTION ALGORITHMS

A. Room exploration - droplet algorithm

To explore an area we must first ensure that the DelFly Explorer can safely operate in its environment without colliding into any obstacles. The vehicle must operate, to the best of its knowledge, as safely as possible. To facilitate this we use the *Droplet* avoidance strategy [24].

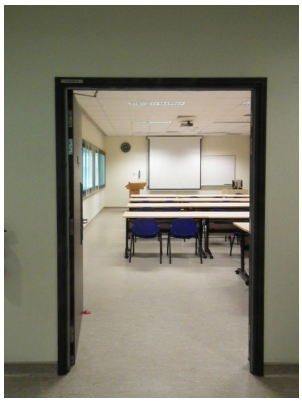

Fig. 3. Full color image of a typical door showing large contrast of color with the surrounding walls and a continuous color defining the three sides of the door.

This method is particularly useful to ensure safe flight in cluttered environments on non-holonomic vehicles equipped with an obstacle detection system with a small field of view. In brief, the Droplet tries to keep an area in front of the vehicle free of obstacles, if an obstacle is identified in this area, the obstacle avoidance is then performed in this obstacle free zone. Unlike strictly reactive obstacle avoidance techniques, the Droplet guarantees a method to perform safe obstacle avoidance with a limited field of view. Please see [24] for more information on the Droplet definition and implementation.

B. Snake-gate detection algorithm

Doors and windows in indoor environments can be characterized in many ways. As can be seen in Fig. 3, one of the many features that most of these openings have in common is that their frame is typically visually composed of a single continuous color which is different than the surrounding structure. The *Snake-gate* detection algorithm is a cascaded gate classifier designed to utilize this very feature. The operation of the Snake-gate detector is graphically shown in Fig. 4 and works as follows.

First, a point is randomly sampled from the image and compared to a user defined target color. If the pixel falls within the predefined color range the gate search begins. The algorithm tests the color of adjacent pixels, in the horizontal or vertical directions. If successful, the algorithm *snakes* along the line of colored pixels that match the target till it finds the extremities of the shape. The local neighbor pixel search is performed in a fixed order (shown in Fig. 4. This ensures that gate sides under an angle or sides with a missing pixel can still be identified.

Once the snaking has found the extremities of the gate side, the detector will only positively identify a side of the gate if it snakes some minimum length. In this paper, this was set to 10 pixels or 0.0782 rad. If the side does not meet this requirement the search is cut short.

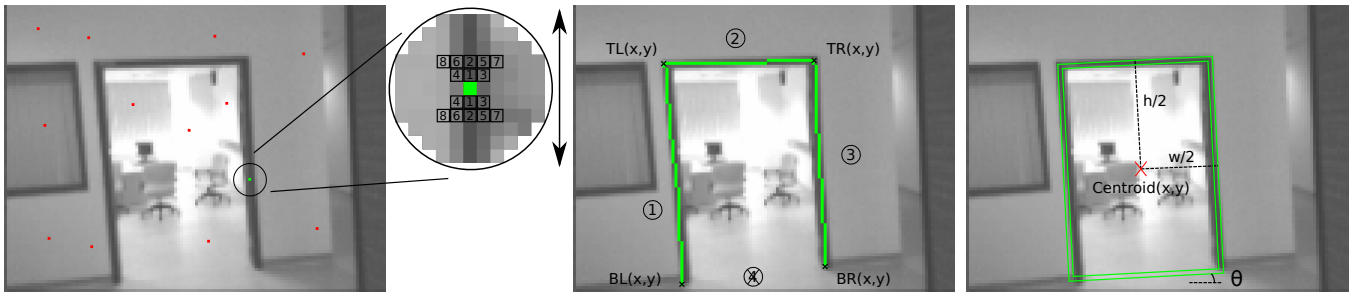

Fig. 4. Graphical depiction of the snake-gate detector applied to a grayscale image captured by the stereo camera. Left image: A pixel is randomly sampled from the image (highlighted in red and green), if it passes the color check the snake-gate search begins. Highlight: The snake-gate detector checks the color of local pixels (identified with highlighted boxes), first checking vertical pixels then further out diagonally (the order is shown with the numbering of the pixels). When a pixel color passes the test, the detector *snakes* to that pixel and process is repeated until the extreme of the line is found. The snake then continues by searching horizontally, left and right. Center image: Shows the result of the first cascade of the snake-gate detector with the coordinates Top-Left (TL), Top-Right (TR), Bottom-Left (BL), Bottom-Right (BR) and the number of sides of the door positively identified. Right image: The identified parameters are then used to compute the centroid, half width, half height and angle of rotation (θ) of the identified gate. This identified gate is then checked against a template to measure its suitability to the goal gate. To identify doors, the identified gate is tested against a rectangle. This is drawn over the image and the percentage of pixels that match the required color range that fall inside of the border of the rectangular shape represents the suitability or fitness of an identified gate.

If the algorithm initially snakes vertically, then the second stage of the search will snake horizontally from the two extremes of the first identified side. Likewise, if the initial search was horizontal, then the second will snake vertically. Again, these two sides are only positively identified if they meet the minimum length requirement. If neither side is positively identified, the search is again cut short. Finally, the algorithm tries to snake the final side of the gate trying to close the shape.

Using this algorithm, at least two sides of the gate must be positively identified to produce a result. This first filter returns the corners of the identified gate and the number of sides identified which is then feed to the second filter where the shape of the identified gate is tested. This is done to allow the detector to discriminate between windows (which are generally square or have a wide aspect ratio), doors (which are generally rectangular with a tall aspect ratio) and other objects.

This filter works by first constraining the identified gate to the desired shape. The shapes used to date are: Square, Rectangle and Polygon. Circles can also be identified however the method is slightly different. For the Square and Rectangle are defined with the center of the identified gate and the longest identified side(s). The rotation (θ) of the shape is also identified and used in the next filter. The Polygon simply draws straight lines between the identified corners.

We then can check the suitability, i.e. *fitness* (f), of this detection by determining the percentage of pixels of the projected shape that belongs to the chosen color range. If the gate has a fitness larger than the previously estimated gate, it is appended to a list of possible gates. These gates are characterized by the relative bearing and azimuth (w.r.t. the center of the image), width, height and rotation in radians. The first positively identified gate must have a fitness larger than a given minimum fitness f_{min}, which was set to 15% in this paper.

Once complete, a new pixel is randomly sampled and the process is repeated until $N_{samples}$ have been sampled or $N_{maxgates}$ gates have been positively identified. In this paper, $N_{samples}$ is 1000 and $N_{maxgates}$ is 25. To further increase the computational efficiency of this detector, any subsequent pixels randomly sampled inside of an already identified gate are rejected to reduce redundant searching.

The random sampling of the image helps to limit the total computational load of the gate search while effectively trying to search the entire image. Additionally, the cascade of filters to throw away unfit gates, sequentially increasing in complexity also helps to remove poor gate candidates before expending unnecessary computation. This method is not constrained to full color images but can also be applied to grayscale images, which is the case when we combine the Snake-gate detection with autonomous obstacle avoidance using the stereo images.

C. Door detection

To specifically identify doors, we use the Snake-gate detector with the Rectangular shape filter. To further discriminate for door detection, we add three additional filter steps.

Firstly, doors are almost always longer than they are wide, if the identified shape from the Snake-gate detector does not have this feature, it is discarded.

The second is based on the fact that although a door can be characterized by a four point polygon, it only has 3 sides. If the identified shape has 4 sides it is discarded. We further specify that the gates must have three sides to be positively identified as a door. If the identified horizontal side is at the bottom of the shape, the gate is also discarded.

Finally, we design a filter to reject the identification of closed doors or filled plates. We test the color vertically and horizontally along the center of the door. If there is a continuous solid color, we discard the candidate as likely not an open door.

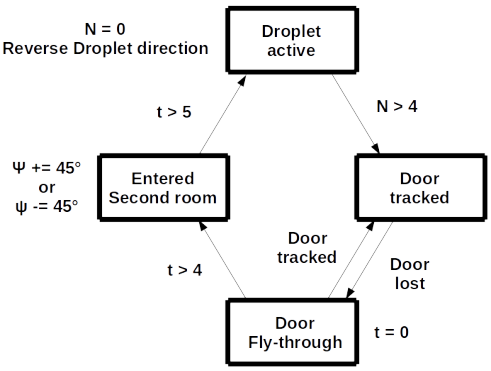

Fig. 5. State machine showing the logic used to effectively traverse the two rooms of the Temasek Laboratories for the displayed demo.

V. DOOR FLY-THROUGH FLIGHT CONTROL AND NAVIGATION TECHNIQUES

Once we have an estimate of the location of the door we need to be able to fly towards and eventually through it. To do this we employed an augmented heading controller. The heading control loop for this task has two main inputs: a heading set-point set to the center of an identified door or a rate command set from the Droplet algorithm. When the Snake-gate detector positively estimates a door location, the relative bearing is passed through a lowpass filter to help limit the effect of outliers and false positives. This filtered signal is then used as the heading set-point of the lateral control system. If no gate is detected, no update is performed so the last set-point is used allowing us to maintain a heading lock to the last identified gate location.

Handling of the rate command is more involved. First, when the Droplet commands a rate, this command is integrated to update the current heading set-point. Due in part to non-linearities in the control effectiveness and to drafts, the DelFly can sometimes not track this commanded rate and lags behind. To ensure the commanded heading does not run away, the integrator is limited to a maximum heading error. Additionally, when the commanded rate is reset to zero after being set, the current heading set-point is also reset to the current heading to null the heading rate integrator and ensure the DelFly will fly in the intended direction. This augmentation resulted in the Droplet effectively acting as a wall following navigation algorithm in a simple room.

Due to the limited field of view of the cameras we use, at some point before we have actually traversed the door, we can no longer see the door. At this point it would be dangerous for the vehicle to perform an avoidance maneuver and therefore once the vehicle is certain that the door fly-through has been initiated the Droplet is disengaged. Fig. 5 shows the state machine used to control the behavioral logic.

In short, the default case is the droplet exploration algorithm is active. To switch to the door tracking mode, we must have three consecutive positive readings from the door detection algorithm. This is done to remove any spurious false positive that may occur during flight. Once we switch to a the door detection, the droplet is temporarily deactivated and we keep tracking the door till we are too close to it to have it in view. At this point we continue flying towards the last heading set-point for 5 seconds, the time it takes to traverse the doors and enter the adjacent room. At this point we offset the heading by 45° (opposite to the current Droplet avoidance direction). This is done to ensure the safe area of the droplet is maintained. This heading is maintained for 1 second after which the droplet is re-engaged.

Although this state machine was specifically designed to facilitate door fly-through for the room layout at the Temasek laboratories, parts of the approach can be generalized in the future. The fixed time used to assume we have traversed the threshold of the door can be replaced using simple odometry. This should make the fly-through behavior more robust to drafts. Additionally, the decision to change the direction of the droplet can be autonomously made, for example, when a particular turn direction causes the DelFly to circle with no positive door detections the DelFly can autonomously reverse its turn direction.

VI. FLIGHT TESTING

A. Door fly-through training

Before testing our approach in a full real world environment, we first performed simplified tests in our flight arena. During the initial tests, we used a red square gate to test and tune the approach algorithms. The red gate was chosen so that the detection was reliable, which enabled us to fine tune the approach algorithms with minimal disturbances from the vision part, such as false positive detections. In these tests, the gate was placed inside our flight arena and the pilot flew the DelFly manually in front of the gate, so that the gate was within the camera view. Once the gate was detected, gate approach algorithm took over and steered the DelFly until it passed through the gate. After that, the pilot took over and again steered the DelFly back in front of the gate. This allowed us to effectively split the exploration and door fly-through parts of the task, testing only one element.

Fig. 6 shows two of the scenarios tested: a simple gate and a double gate with side panels to simulate walls. The bottom of the same figure displays the MAV trajectories as recorded with a motion tracking system (24 $\times$ OptiTrack Flex13, tracking volume 9 $\times$ 9 $\times$ 5 m), where the gate sides are represented by the red asterisk symbols. The trajectories show that most of the flights were successful and that the gate could be successfully approached (and passed) even when starting (substantially) off the gate axis.

In the first tests, the aileron deflection was proportional to the detected gate position measured with respect to the center of the image. This approach worked well as long as the detection was reliable. However, if the window was lost, e.g. due to motion blur caused by an external disturbance, the success rate would decrease. Thus, we decided instead to calculate the gate heading, as explained in the previous section, and fly towards this heading even when the gate was lost for several frames. This approach further improved the robustness of the gate approach. In addition, a basic complementary filter on the detected gate heading was added

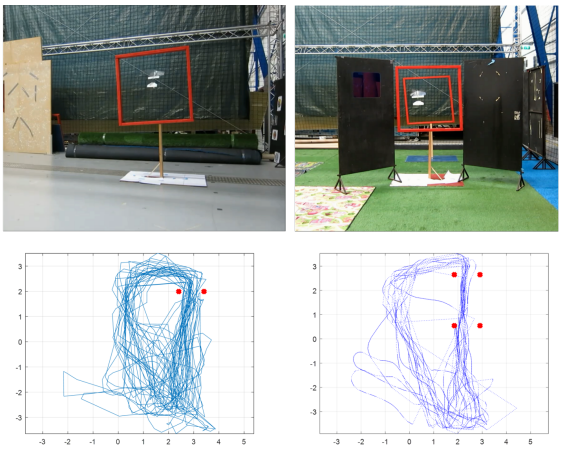

Fig. 6. Two scenarios used in the initial tests: a single gate (left) and a double gate with side panels (right). The plots at the bottom of the figure display the flight trajectories recorded by a motion tracking system (the units meters). The gate side positions are represented by the red asterisk symbols.

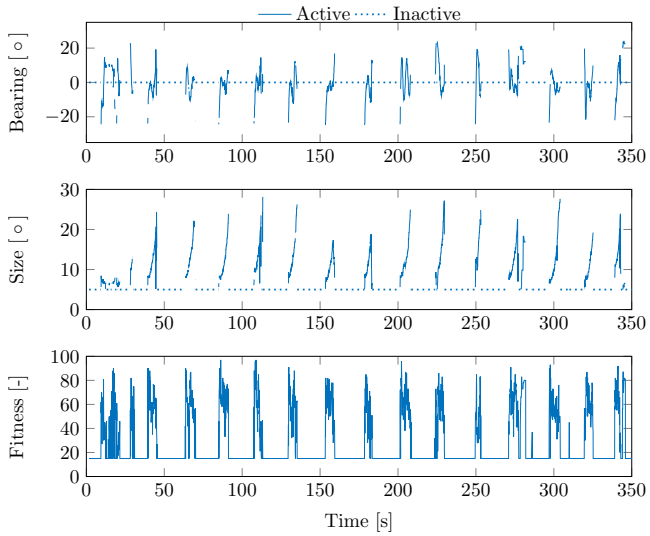

Fig. 7. Telemetry from the single gate tests in Fig. 6 left. The plots show (from top to bottom) the relative gate position, the gate size and gate fitness. In total 15 passages were attempted, 13 of which were successful.

to reduce the effect of the detection algorithm noise and smoothen the approach.

Fig. 7 shows an example of telemetry data received during 15 subsequent gate-flight-throughs, 13 of which were successful and the remaining two ended up by a crash into the side of the gate. The crashes were caused by a vehicle sideways drift in the last phase of the approach, when the gate is already out of the camera field of view.

B. Complete mission

After performing isolated tests of the door pass-through, we performed a full-system test in an unaltered, meeting room environment in a typical office building. The environment was quite challenging, as there were many places with little visual texture (making stereo matching in those places

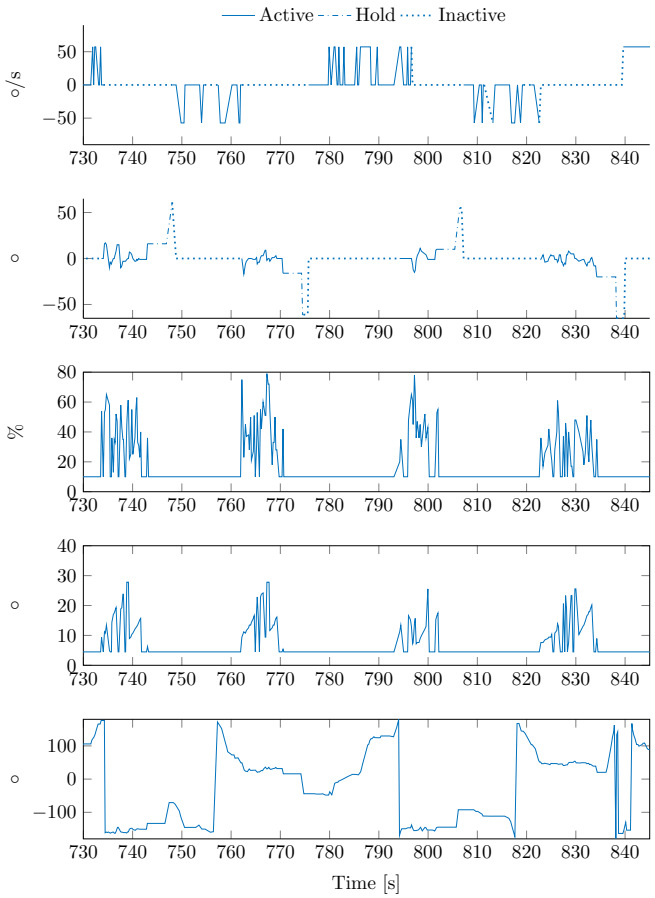

Fig. 8. Telemetry from an autonomous flight tests of the DelFly Explorer performing multi-room exploration as depicted in Fig. 1. The DelFly starts by exploring Room 1, finding and flying through the two sets of doors into Room 2 and returning. This flight was autonomously repeated twice in a row. From top to bottom: Turn command from droplet collision avoidance algorithm; Relative heading to the identified door from the Snake-gate detector; Certainty of door as detected from the Snake-gate detector; Size of the identified door in pixels from the Snake-gate Detector; and the heading set-point of the flight control system.

difficult) and windows with dark borders (possible distractors for the door detection). Even in this visually challenging environment, the DelFly Explorer was able to reliably pass multiple times from one room to the other and back.

Fig. 8 shows the telemetry of one of the tests where we were able to complete 3 room traversals with the 4th attempt ending in a false positive detection that ended the flight. This type of flight performance was quite repeatable with the vehicle repeatedly flying through both rooms multiple times. Some sample images of the DelFly Explorer performing the flight can be seen in Fig. 9.

VII. CONCLUSION

We have demonstrated for the first time a flapping wing robot performing a multi-room exploration task by only using on-board sensing and processing. This achievement relied on hardware adaptations (enlarging the ailerons) and on new, computationally efficient algorithms for door detection and control. With these algorithms, the flapping wing robot is

Fig. 9. Sequence of images showing the flight of the DelFly Explorer performing autonomous exploration and door fly-through.

able to fly around rooms and reliably pass through doors. This is an important building block for future indoor autonomous flight.

The presented algorithms still have their limitations. The door detection currently relies on the door border being darker than the surrounding walls. Although this is often the case, they may also be lighter though as well. Future work should definitely investigate more generic solutions, looking for instance at contrast in general (to also capture door borders lighter than their surroundings) and at the disparity measurements in the found door opening. Moreover, the current algorithms have the flapping wing robot fly from one room to the other and back indefinitely. Higher level navigation, potentially using efficient methods such as topological SLAM, is hence a very important topic for future research as well.

REFERENCES

[1] A. Briod, J.-C. Zufferey, and D. Floreano, "A method for ego-motion estimation in micro-hovering platforms flying in very cluttered environments," *Autonomous Robots*, vol. 40, no. 5, pp. 789–803, 2016.

[2] K. N. McGuire, G. C. H. E. de Croon, C. De Wagter, K. Tuyls, and H. J. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone," *Robotics and Automation Letters*, vol. 2, pp. 1070–1076, 2017.

[3] C. Ellington, "The aerodynamics of hovering insect flight. I. The quasi-steady analysis," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, pp. 1–15, 1984.

[4] M. H. Dickinson, F.-O. Lehmann, and S. P. Sane, "Wing rotation and the aerodynamic basis of insect flight," *Science*, vol. 284, no. 5422, pp. 1954–1960, 1999.

[5] 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 Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. AIAA Nashville, TN, 2012, pp. 1–24.

[6] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled Flight of a Biologically Inspired, Insect-Scale Robot," *Science*, vol. 340, pp. 603–607, 2013.

[7] M. Karásek, A. Hua, Y. Nan, M. Lalami, and A. Preumont, "Pitch and Roll Control Mechanism for a Hovering Flapping Wing MAV," *International Journal of Micro Air Vehicles*, vol. 6, no. 4, pp. 253–264, feb 2014.

[8] G. C. H. E. De Croon, M. Perçin, B. D. W. Remes, R. Ruijsink, and C. De Wagter, *The DelFly: design, aerodynamics, and artificial intelligence of a flapping wing robot*. Springer, 2015.

[9] H. V. Phan, T. Kang, and H. C. Park, "Design and stable flight of a 21g insect-like tailless flapping wing micro air vehicle with angular rates feedback control," *Bioinspiration & Biomimetics*, vol. 12, no. 3, Apr 2017.

[10] A. Roshanbin, H. Altartouri, M. Karásek, and A. Preumont, "COLIBRI: A hovering flapping twin-wing robot," *International Journal of Micro Air Vehicles*, vol. 9, no. 4, pp. 270–282, mar 2017.

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

[12] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *2009 IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2878–2883.

[13] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[14] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[15] S. B. Fuller, A. Sands, A. Haggerty, M. Karpelson, and R. J. Wood, "Estimating attitude and wind velocity using biomimetic sensors on a microrobotic bee," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1374–1380.

[16] E. F. Helbling, S. B. Fuller, and R. J. Wood, "Altitude estimation and control of an insect-scale robot with an onboard proximity sensor," in *Robotics Research*. Springer, 2018, pp. 57–69.

[17] S. S. Baek, F. L. G. Bermudez, and R. S. Fearing, "Flight control for target seeking by 13 gram ornithopter," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2674–2681.

[18] S. Ryu, U. Kwon, and H. J. Kim, "Autonomous flight and vision-based target tracking for a flapping-wing mav," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 5645–5650.

[19] G. C. H. E. de Croon, K. M. E. de Clercq, R. Ruijsink, B. D. W. Remes, and C. de Wagter, "Design, aerodynamics, and vision-based control of the delfly," *International Journal of Micro Air Vehicles*, vol. 1, no. 2, pp. 71–98, 2009.

[20] S. S. Baek and R. S. Fearing, "Flight forces and altitude regulation of 12 gram i-bird," in *2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2010, pp. 454–460.

[21] R. C. Julian, C. J. Rose, H. Hu, and R. S. Fearing, "Cooperative control and modeling for narrow passage traversal with an ornithopter mav and lightweight ground station," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 103–110.

[22] K. Y. W. Scheper, S. Tijmons, C. C. de Visser, and G. C. H. E. de Croon, "Behavior trees for evolutionary robotics," *Artificial life*, vol. 22, no. 1, pp. 23–48, 2016.

[23] C. de Wagter, S. Tijmons, B. D. W. Remes, and G. C. H. E. 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)*, Hong Kong, jun 2014, pp. 4982–4987.

[24] S. Tijmons, G. C. H. E. de Croon, B. D. W. Remes, and M. Mulder, "Obstacle Avoidance Strategy using Onboard Stereo Vision on a Flapping Wing MAV," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 858–874, 2017.

[25] B. D. W. Remes, P. Esden-Tempski, F. van Tienen, E. J. J. Smeur, C. De Wagter, and G. C. H. E. de Croon, "Lisa-S 2.8g autopilot for GPS-based flight of MAVs," in *2014 International Micro Air Vehicle Conference and Competition (IMAV 2014)*. Delft University of Technology, aug 2014, pp. 280–285.