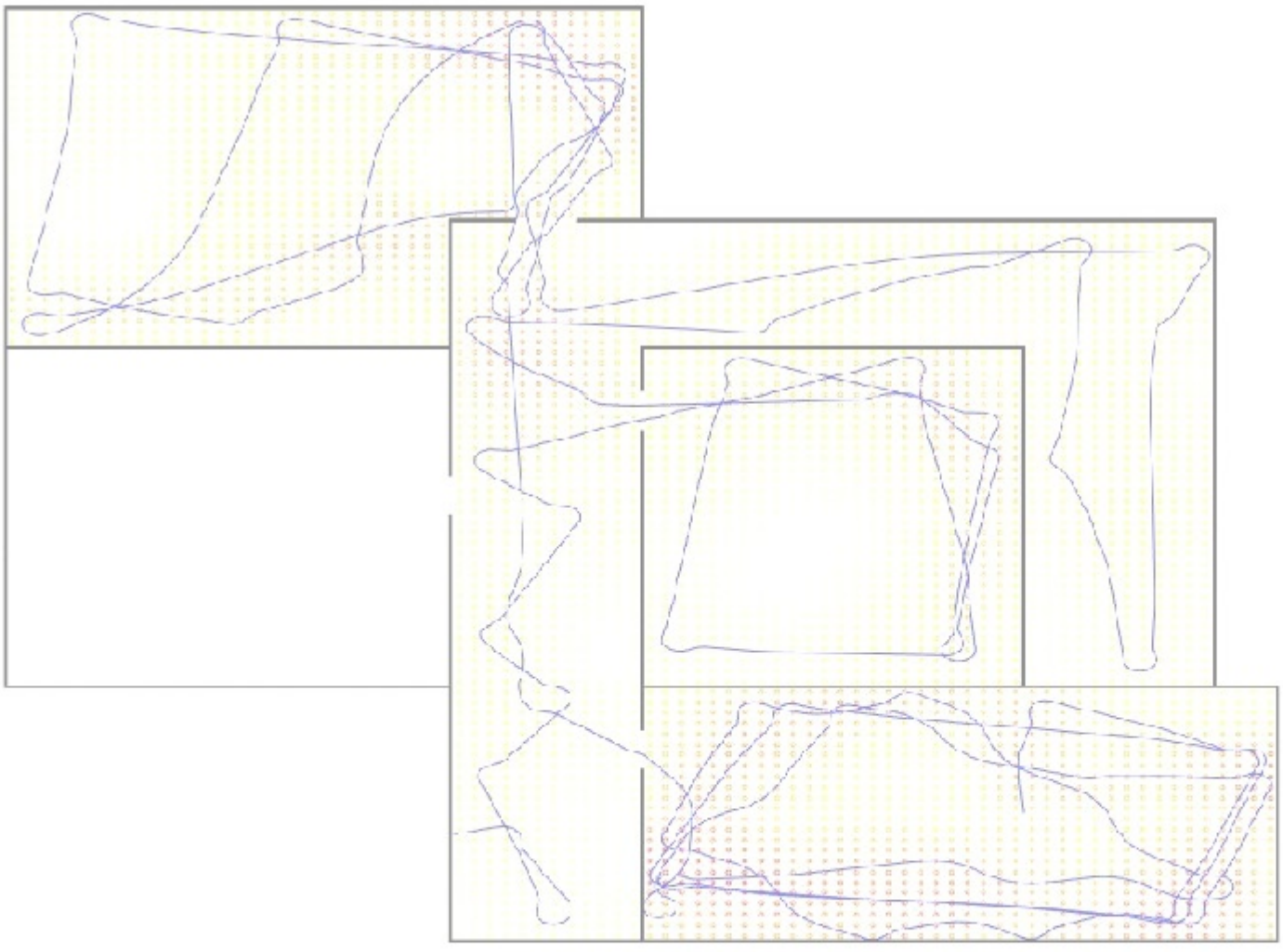
Faculty of
Aerospace Engineering

The Exploring DelFly

How to increase the indoor explored area of the DelFly Explorer by means of computationally efficient routing decisions?

C.R. Fonville

August 1, 2016



TU Delft Delft University of Technology

Challenge the future

The Exploring DelFly

How to increase the indoor explored area of the DelFly Explorer by means of computationally efficient routing decisions?

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

C.R. Fonville

August 1, 2016

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"The Exploring DelFly"** by **C.R. Fonville** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 1, 2016

Readers:

Prof. dr. ir. M. Mulder

dr. G. C. H. E. de Croon

dr. ir. C. J. M. Verhoeven

ir. S. Tijmons

Acronyms

EF	exploration factor
FWMAV	flapping wing micro aerial vehicle
MAV	micro air vehicle
UAV	unmanned aerial vehicle

List of Symbols

Greek Symbols

Δ	Pixel disparity value

Roman Symbols

d	Distance between the left and right lens of the stereo-vision camera
f	Focal length of the lens of the camera
h	Total height of the picture frame in pixels
i	Pixel horizontal position in the picture frame
j	Pixel vertical position in the picture frame
w	Total width of the picture frame in pixels
X‘	Pixel horizontal distance with respect to the camera
Y‘	Pixel vertical distance with respect to the camera
Z‘	Pixel depth distance with respect to the camera

Contents

Chapter 1

Document Introduction

This document consist of a thesis with a paper included. The paper is the primary graduation deliverable and can be found in Part I. The other parts of this document elaborate more on the different aspects presented in the paper. But also on the origin of the research topic. Part II focuses on this origin. A clear literature study (part of the preliminary graduation) is presented and more information on how the research question was chosen and what the different investigated sub-questions were. In Part III the set-up of the simulations and the different simulations modules will be discussed. This part will also compare the newly developed methods with old methods to determine the success-rate of the new algorithms. Part IV will discuss the performance of several modules that were tested in real flight experiments. Finally, Part V will conclude the primary findings of this research and will be accompanied by several recommendations which can improve the algorithm and aid in further research.

Part I

Paper

Increasing Indoor Exploration Capabilities of the DelFly Explorer Flapping Wing Micro Air Vehicle

C.R. Fonville*, S. Tijmons, G.C.H.E. de Croon

Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands

Small robots, such as Micro Aerial Vehicles, form an increasingly popular field of interests in research, industry and the consumer market. The autonomous capabilities of these systems keep evolving and one of the main research goals is to reach full autonomy. However, this is often achieved at the cost of growing hardware demands. In this study a computationally light and efficient way to enhance autonomous on-board exploration capabilities for the DelFly Explorer, a 20-gram flapping wing Micro Aerial Vehicle (FWMAV), is presented. Both theory and new insights were combined to design an exploration algorithm for the on-board stereo-vision system. The algorithm primarily consists of a disparity map based decision tree, different exploration phases and computationally light odometry. Computer simulations proved the effectiveness of the algorithm to enable autonomous exploration capabilities for the FWMAV system. Initial flight tests also show that the proposed algorithm increases its exploration capabilities and form a foundation for future research.

Nomenclature

Δ	Texture disparity [−]
A	Area [m^2]
d	Baseline distance between the stereo camera lenses [mm]
f	Focal length of the camera [mm]
h	Height of the camera image in pixels [−]
i	Horizontal pixel location w.r.t. upper left image corner [−]
j	Vertical pixel location w.r.t. upper left image corner [−]
t	Time [s]
V	Speed [$\frac{m}{s}$]
w	Width of the camera image in pixels [−]
X'	Cartesian coordinate of the texture with respect to the camera lens, horizontal plane sideways [mm]
Y'	Cartesian coordinate of the texture with respect to the camera lens, vertical plane [mm]
Z'	Cartesian coordinate of the texture with respect to the camera lens, horizontal plane forward [mm]

*Graduate Student, Department Control & Operations, Section Control & Simulation, Kluyverweg 1, 2629 HS, Delft, NL

I. Introduction

THE autonomous capabilities of small robots and drones are increasing rapidly. Part of these autonomous capabilities is the ability to explore and/or map the surrounding environment. This research focuses on the exploration of buildings. The capabilities developed can also be applied in tasks such as search and rescue operations, building inspection or entertainment.

The majority of research in autonomous exploration tasks is based on the increasing computational and memory capacities of current technology. Simultaneous localization and mapping (SLAM) is one of the most advanced methods currently available.[1,2,3,4,5] However, SLAM is computationally very demanding where the use of a 1.6GHz processor and 1GB RAM used by Shen *et al.* is no exception, see Ref. 6. Decreasing the computational complexity of exploration algorithms, instead of increasing it, is therefore an interesting research branch. A method already widely applied are the BUG-algorithms. BUG- algorithms are a family of exploration strategy algorithms which have very low computational demands.[4,7] However these methods are often purely reactive, have no (topological) map, and need a preset destination or specific target the robot can search for.[8,9]

Extension of the standard BUG-algorithms is done by Ref.4 in the Pursuit-Evasion BUG-algorithm (PE-BUG). This method combines the simple features of BUG-algorithms such as obstacle avoidance and wall-

following in combination with a simple topological map to clear a complex shaped environment from intruders. This combination of wall-following algorithms and topological maps forms a good starting point for this research.

As the drone of interest is light weight and therefore has low computation power, low computational demands are required. The used platform is the DelFly Explorer, a flapping-wing micro air vehicle (FWMAVs) shown in Figure 1.[10,11] The field of FWMAVs is largely uncharted, especially there where autonomous capabilities are incorporated. The DelFly Explorer is a tailed FWMAV. The presence of this tail ensures the passive stability of the DelFly Explorer, in case of a minimal forward velocity. Due to this forward velocity the DelFly Explorer has a speed dependent turn radius, whereas a tailless design such as the Nano Hummingbird,[12] which can rotate in-place, has a higher manoeuvrability.

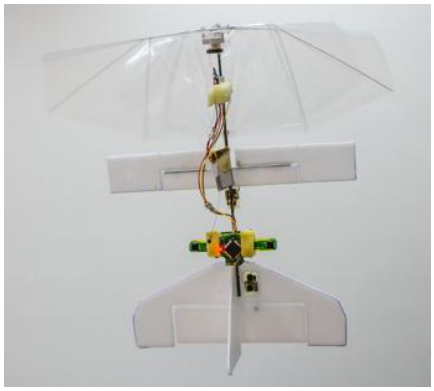


Figure 1: Photo of the DelFly Explorer, its wingspan is 28cm, the weight is 20 gram, the stereo-board is located above the tail.[13]

In this article, an exploration algorithm is presented which will aid in increasing autonomous exploration capabilities. The developed algorithm is specifically optimized and applied to the DelFly Explorer. The main aim in this research concerns the design of a system with modular elements that have added value into reaching the aforementioned goal. This leads to the following research question:

How to increase the indoor explored area of the DelFly Explorer by means of computationally efficient routing decisions?

In Chapter II, the current configuration of the DelFly Explorer will be discussed. This configuration will form the basis on which the research of this article is conducted. The method that will be applied and how that method will be evaluated will be discussed in Chapter III. In Chapter IV the different modules used in the algorithm will be developed and tested in simulation. The combined simulation result will be analyzed at the end of the chapter. In Chapter V different modules will be converted from simulation to the actual platform and their performance will be evaluated in real experiments. Finally, Chapter VI will provide an overall conclusion of the project and recommendations for future and successive research.

II. Current Configuration

A. The DelFly Explorer

The DelFly Explorer is equipped with a ATmega 328P - MLF28 micro-controller (autopilot). And is capable of two-way communication with a ground-station. The battery of the DelFly Explorer provides energy for six minutes of flight time. 3-axis accelerometer, magnetometers, gyrometers, and a barometer are part of the sensors of the DelFly Explorer. The primary sensor used is a stereo-vision camera with two 640x480 pixels sensors on a baseline of 6 cm located just above the tail, as visible in Figure 1. The used resolution is equal to 128x96 pixels. The two datastreams of the cameras are received by a complex programmable logic device (CPLD). The CPLD merges the grey-scale components of the images into a single stream. On a STM32F405 processor (168MHz, 192kB RAM) the video stream-data can be analysed (stereo-board).[14]

With the stereo-vision cameras a disparity map can be constructed.[15] Such a map shows the relative shift of the textures for each of the two frames of the stereo camera. With this relative shift an estimate of the distance to the texture is obtained. This disparity map forms the basis of the primary algorithms used in this research.

B. Droplet

A method to evade obstacles was already developed for the DelFly Explorer by Tijmons *et al.*, see Ref.16. This method consist of a droplet shaped area in front of the DelFly Explorer which is kept clear of obstacles at all time. The droplet shaped area will be addressed as *the droplet*. See Figure 2.

The strategy is to keep the area of the droplet, in which the DelFly Explorer can still safely make a full turn, clear of any obstacles. The area of the droplet is within the field of view from the position of the DelFly Explorer, visualized in Figure 2. This is why the shape is droplet-shaped and is determined by the turn radius (r), field of view (α), and safety margin (m). These constraints also determine the total length (l) of the droplet, which is currently set to 3 meters. The droplet is designed for making clockwise turns only, therefore the camera is also aimed under

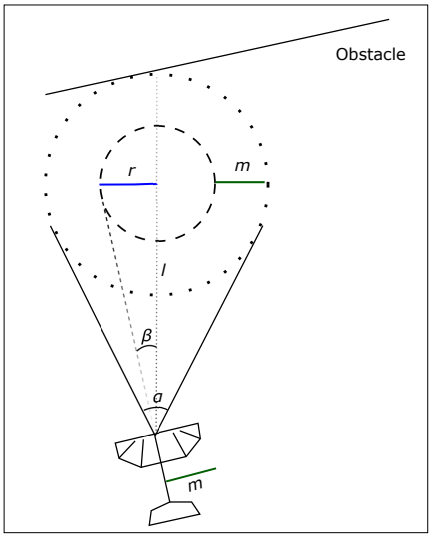


Figure 2: Graphical droplet representation with stereo-camera under angle β.[17]

an angle (β).

III. Method

A. Exploration Factor

For determining the success of the exploration of the DelFly Explorer and compare obtained results, a figure of merit is required, *the exploration factor*. A number of factors are of importance in this formula. First, and most important, is the total area that is visited. Secondly, one of the sub-goals of this project is to be able to access multiple rooms. Therefore the ratio of rooms over the total rooms that could have been explored will be incorporated in the exploration factor. Lastly the speed and flying time of the DelFly Explorer will be incorporated. Multiplication of those parameters will result in the total distance travelled, which again with the total explored area can give an insight in the efficiency of the taken route. The exploration factor is then expressed as:

$$EF = \frac{nRooms}{nRoom_{total}} \cdot \frac{A_{explored}}{A_{total}} \cdot \frac{A_{explored}}{V_{average} \cdot t} \qquad (1)$$

This equation directly shows some limitations of the figure of merit. For example, if two runs are compared where one of the runs has a much larger total area, that run would have a lower exploration factor due to the fact that it has explored a lower percentage of the available area. The other way around however, if it wouldn't be a ratio, in the bigger room there would be more exploration possible after a certain time while in the smaller room exploration is finished already. Therefore, it is recommended when comparing results that equal or similar total areas should be considered. Similar argumentation can be used for the number of rooms. Furthermore, the total simulation time (t), if it differs, compensates in the last term the decreased explored area from the second factor. However, this compensation is not adequate for the number of rooms visited. Introducing a second recommendation to keep simulation time constant over the different compared experiments.

B. Algorithm layout

As discussed in the introduction the total simulation will be build up out of different modules, each of these modules will represent an *exploration phase*. Exploration phase 1 will identify the general contour layout of the room by means of wall-following. Exploration phase 2 will, when the contour of the room is known, explore the remainder. Exploration phase 3 consist of an algorithm which will identify door candidates. This exploration phase will provide the location of the best candidate to the algorithm of exploration phase 4, which will head for the door. These last two phases interchange quickly for better convergence. Finally some overall modules will handle obstacles, room detection, and a topological map. A finite state machine representation of this structure is provided in Figure 3.

IV. Simulation

SmartUAV* is an in-house C++ software package that can be adapted to simulate several unmanned aerial vehicles (UAVs). SmartUAV, developed by the MavLab, contains a library of these different UAVs and their dynamics, layout, in-, and outputs. For the DelFly Explorer this data is already available within the SmartUAV environment. Moreover, SmartUAV builds a graphic environment based on the layouts provided.

There are three different environment layouts used for the simulations, each layout represents a couple of challenges and possible real world situations. Furthermore the textures that are used in the environment represent typical wall, corridor, floor and roof profiles. The three environments are visualized in Figure 4.

In simulation several assumptions were made: constant flying velocity, perfectly known dynamics, no disturbances, static environment, and constant flying altitude.

*https://svn.lr.tudelft.nl/trac/ADIO-CS/SMARTUAV/

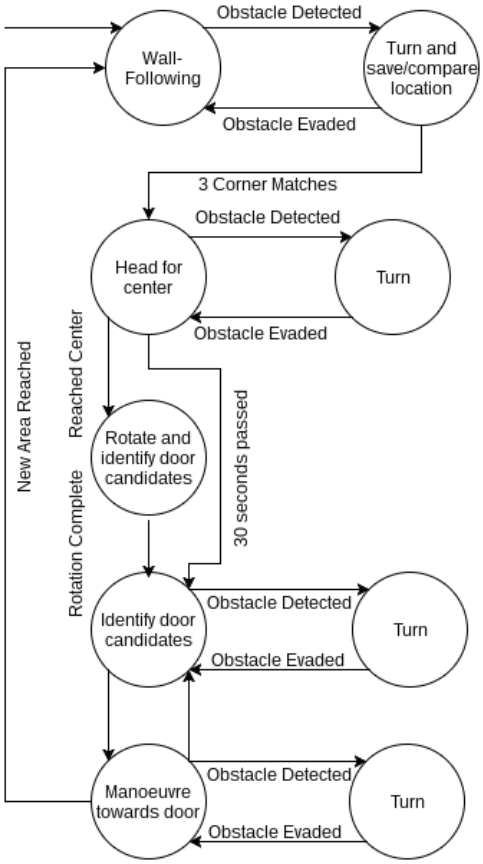


Figure 3: Finite state machine representation of the exploration algorithm.

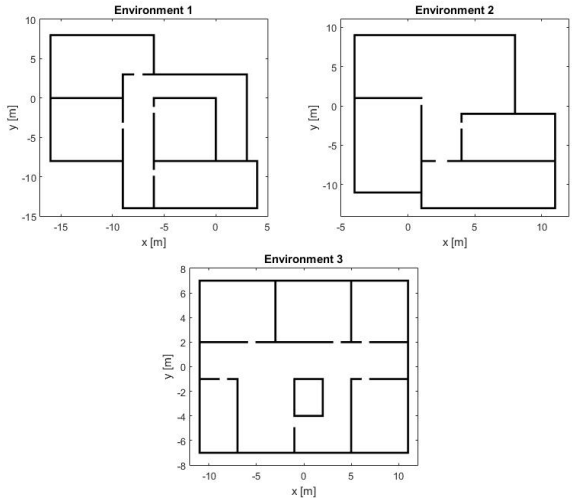


Figure 4: Layout of the three different environments used in simulations.

A. Stereo-vision Altitude Estimate

A rather stand-alone subject is the flying altitude of the DelFly Explorer. The main control loops in this simulation focus on the horizontal plane. In a large part of the simulations the flying altitude of the DelFly Explorer was fixed. However in real world scenario's this is not feasible. In order to prevent any collisions of the DelFly Explorer with ceilings or floors a constant altitude would be preferable, this also eases the process of matching visual cues. In Chapter II it was mentioned that one of the sensors on-board is a barometer. To determine the exact altitude, a pressure table is necessary which matches a pressure to an altitude. However the corresponding air-pressure for a certain altitude can even differ per room in the environment. Therefore a calibration method has to be designed to calibrate the barometer for every room.

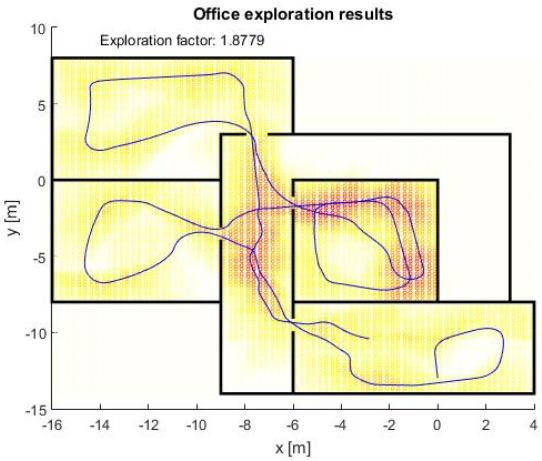
In Chapter II the working principle behind obtaining a disparity map was explained in short. From the disparity map a 3D-coordinate with respect to the left-camera for each detected pixel can be calculated via:

$$Z' = \frac{d \cdot f}{\Delta} \qquad X' = (i - \frac{w}{2}) \cdot \frac{Z'}{f} \qquad Y' = (j - \frac{h}{2}) \cdot \frac{Z'}{f} \tag{2}$$

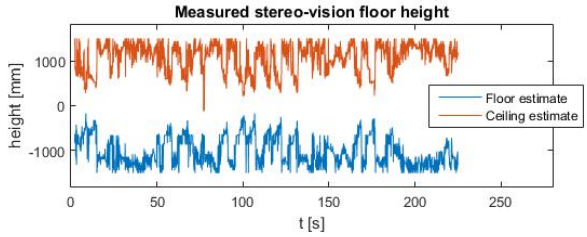
Where Z' is the depth coordinate of the texture and X', Y' are the horizontal and vertical coordinate respectively. (Δ) denotes the pixel disparity, b the lateral distance between the lenses, and f the focal length. Finally, h and w are the image height and width respectively and (i, j) is the pixel position with respect to the image centre.

The lowest value obtained for the vertical output of the different pixels corresponds to the height of the floor with respect to the DelFly Explorer (when the floor is in visual range). This value can then be used to calibrate the barometer. During simulation the above described method was tested, with a constant flying altitude of 1,5m. The raw height measurements based on the disparity map are shown in Figure 5b. The route flown through environment 1 for this specific data-set is shown in Figure 5a.

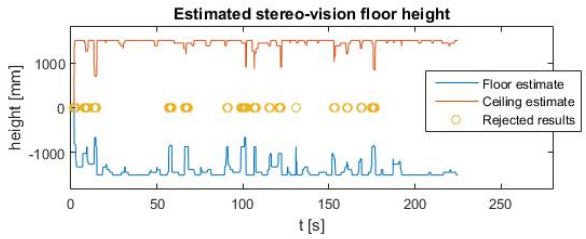
However using this varying altitude output to constantly calibrate the barometer is due to all the noise not very useful and will results in erratic flight behaviour. Therefore an addition to this algorithm would be essential. The calibration method could be improved by taking the lowest dominant value within the last measurement period to calibrate the barometer. The result of this addition can be seen in Figure 5c. The result is a significantly less erratic output. One will still observe some peaks, these measurements will be rejected When the height difference be-



(a) Route flown during the stereo-height simulation.



(b) The measured floor/ceiling altitude by the stereo-height calculations algorithm during simulation.



(c) The estimated floor/ceiling altitude estimated over a period by the height calculations algorithm during simulation. Rejected measurements due to a detected wall are also shown.

Figure 5: Stereo-vision altitude estimate simulation results.

tween the floor and ceiling is less than 1.5 m. These are denoted with a circle in the figure. Analysis of the route showed that these rejected measurements indeed correspond to moments when the DelFly Explorer was approaching an obstacle, and thus the floor and ceiling were not within view.

B. Droplet Adjustments

Rooms and corridors are explored clockwise. Most corridors have a relative narrow layout, due to this fact it is possible the droplet is triggered by the corridor wall on the right side while the DelFly Explorer is following the left wall. A clockwise turn would then result in flying back in the direction the DelFly Explorer just came from. Therefore the possibility should exists to make a counter-clockwise turn. Changes to the droplet were made accordingly, aiming the camera along the body axis. Based on the disparity map and exploration phase the turning direction of the droplet is decided. When the turning direction is known the DelFly Explorer makes a course correction to steer in for the droplet turn.

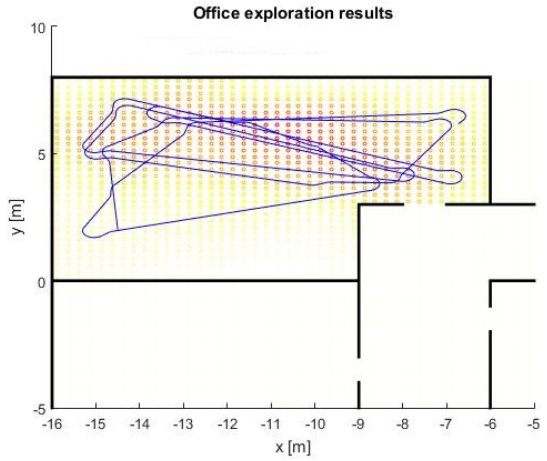
C. Wall-Following

The wall-following algorithm is developed such that rooms and corridors can easily be explored and to aid in the determination of the roomsize. The wall-following algorithm is programmed to explore rooms and corridors clockwise. Therefore the algorithm tries to keep the wall on the left side of the DelFly Explorer. The disparity map, which was explained in Chapter I, and the droplet are used as the basis for the algorithm to do this. The algorithm evaluates the disparities of the left half of the disparity map with respect to the droplet settings. If a minimum amount of textures have a too low disparity value, meaning textures are far away, the DelFly adjusts its heading to the left. When disparities are too high, textures are too close, the heading is adjusted to the right. However, when the difference between the average disparities on the left and right half plane are below a certain threshold and the detected textures are almost nearby, a left turn is prevented. This turn is prevented such that the DelFly Explorer does not crash near corners.

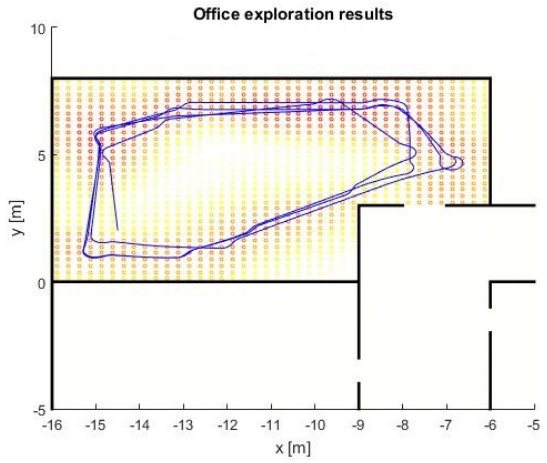
The output of the wall-following algorithm will be an actuator input, either for correctly following the wall, or because the droplet was activated, or to adjust for the corridor direction. Methods not based on the disparity map, but for example on contour-lines were not robust enough.

The difference of the wall-following method with respect to solely the droplet is visible in Figures 6a - 6b. In these figures the black lines represent the

walls, the blue line the path of the DelFly Explorer. The background intensity resembles how long an area has been insight. Note: the droplet is also part of the wall-following algorithm to ensure a collision free flight. The performance is determined by the average distance from the wall. For the wall-following algorithm this is 2.03m on average and for the droplet it is 2.69m, obtained in environment 1. From this it can be concluded that the wall-following algorithm works successfully.



(a) Droplet-only behaviour.



(b) Wall-following behaviour.

Figure 6: Simulated flight behaviour of the DelFly Explorer for different settings with a 150 seconds simulation time.

D. Room detection

Room detection is the method in which the size and shape of the room is detected and is determined when the room has been explored. The wall-following provides a good basis for room detection. As seen in Figures 6a - 6b the wall-following algorithm is better able to reach the outskirts of the room than the droplet-only method. These outer points can be used to define a polygon which approximates the room layout. The corner points of this polygon are based on the locations where the droplet was activated and the heading change was above a preset threshold. These points tend to correspond indeed with the major corners of the room. Simultaneously each newly created point is compared to the already stored points based on the heading before and after the turn, and their estimated location. Only when a turn fulfills the next two requirements it is regarded as such.

The first requirement is evaluated based on the magnetometer, each time a turn is made the magnetometer reading (heading) is stored and linked to this turn. A future turn can be matched to this turn based on their magnetic fingerprint. In a square room this method already rules out 75% of the locations in the database. For more diverse shapes this value can become $\geq 75\%$.

For the second requirement odometry is used. In simulation the speed is fixed, flight time is known, and the heading changes are recorded by the magnetometer. With this information the position of the DelFly Explorer can be estimated. When a new turn is detected within a 3 meter radius of a previous point, the turns are matched. So according to the odometer the DelFly Explorer has been here before.

Note that the assumptions of no magnetometer disturbances and constant speed may have a big influence on this part of the algorithm when it would be tested in real situations.

When three successful turn matches are made the final room layout is defined by the polygon and assigned a room number. During the successive exploration phases an algorithm is constantly checking whether the DelFly Explorer left the polygon (room) and thus has successfully entered an new area of the environment. When this is the case the exploration phase moves back to 1 (wall-following), the old polygon is saved for later reference, and the construction of a new polygon is started.

E. Door Manoeuvring

The next explorations phase (phase 2) is initiated when the final room layout is drafted. In this phase the DelFly Explorer will move to the center of the polygon and will attempt a full rotation, as can be seen in Figure 7. This will provide an overview of the room as a whole, including the less or even not explored areas. Simultaneously exploration phase 3 is activated, responsible for identifying door candidates. So that during the full rotation the total overview of the room can be used to identify a good first door can-

didate. If this manoeuvre is not completed within 30 seconds it will be terminated. This threshold was set because sometimes the polygon's centre of gravity is located near a wall and the droplet will be activated constantly before the rotation is finished or started.

In Figure 7 it is clear that during the rotation the door was detected and after the rotation the DelFly Explorer is heading in the direction of this opening. However this method is subject to some disadvantages, due to the locations of the turn matches. These locations influence the position of the rotation location, often in the direction of the walls. This causes an unclear or even no rotation at all. Furthermore, the maximum time reserved for this phase ensures not much time is used in trying to perform this manoeuvre.

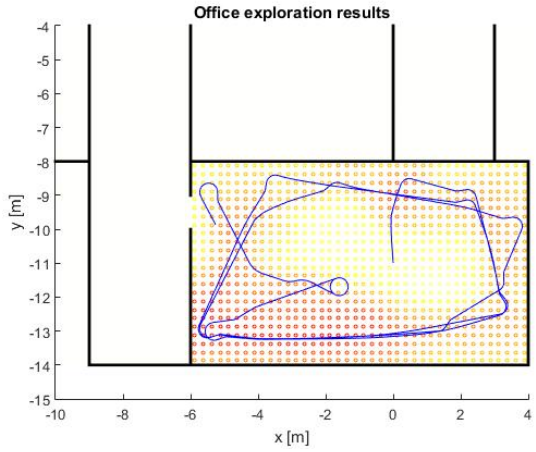


Figure 7: Flown track of wall-following algorithm (phase 1) followed by the center rotation (phase 2).

The door detector algorithm is based on the disparity map. This can be visualized by dividing the disparity map in N different vertical segments, a visualisation for 10 histograms is given in Figure 8. In this figure one can see the higher disparities near the edges of the door. For each of those vertical segments (bins) the average disparity value and total number of detected disparities is calculated. Additionally for each bin the contrast with respect to a number of adjacent bins is calculated. This to represent the door feature of a far opening in a relatively nearby plane. The bin with the lowest disparity (corresponding with textures the furthest away) that is under the minimum disparity threshold which also has a total number of disparities above a certain threshold (this to reduce the effect of false positives and noise) and has a favorable contrast will be selected as target bin. This method is used to recognize room exits.

When a target bin is selected, the final exploration phase (phase 4) is initiated. The directional vector

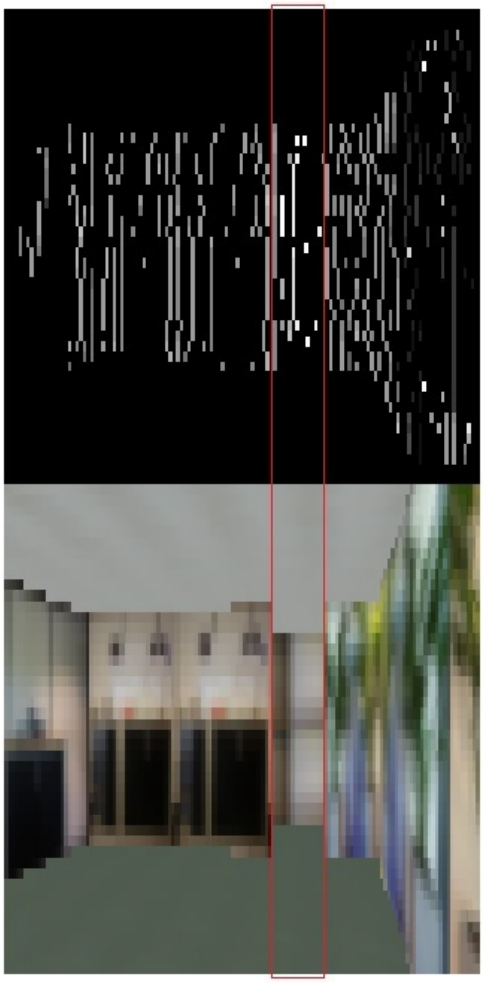
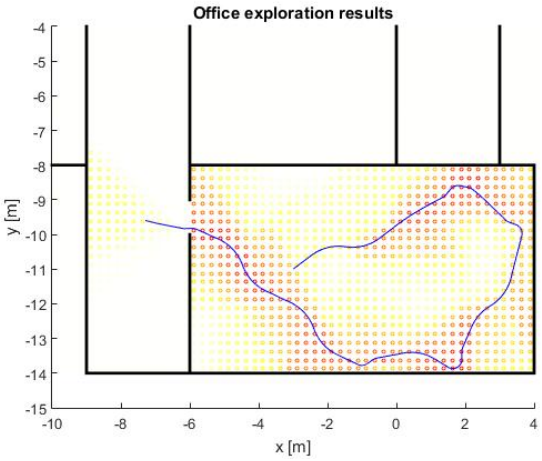


Figure 8: Door detector vertical target bin for 10 histograms, above the disparity map, below the simulator view.

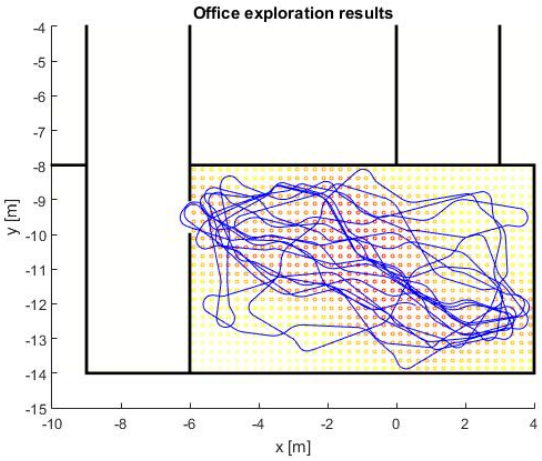
from the current location to the target bin is computed, an S-shaped path is planned and executed accordingly. This S-shaped path always remains within the confines of the droplet, therefore only small heading changes are used.

While the S-shaped manoeuvre is carried out a constant re-evaluation of the target and the path is performed. If needed the current manoeuvre is stopped and a new one is initiated to ensure optimal navigation towards and through the door. By using this S-shape path and the target re-evaluation the DelFly Explorer ends up in front of the door instead of at a suboptimal angle.

In Figure 9a a typical successful path of this algorithm is visualised in a simulation without prior wall-following. Noticeable are the consecutive s-curves taken towards the door opening, due to the constant re-evaluation of the taken flight route.



(a) Successful DelFly Explorer path when performing the S-shaped door tracking and door manoeuvre algorithms (no initial wall-following), with 20 histograms.



(b) Unsuccessful DelFly Explorer path when performing the S-shaped door tracking and door manoeuvre algorithms (no initial wall-following.)

Figure 9: Simulation door manoeuvring results

Figure 9b shows the worst case scenario of the algorithm. Here the DelFly Explorer is constantly approaching the door opening but failing to go through the door because the droplet is activated, thus a crash-free attempt could not be guaranteed. Again the consecutive S-shaped turns are visible in the taken trajectory. After a failed attempt the DelFly Explorer flies to the other end of the room. An improvement to this algorithm could be to let the DelFly Explorer do several attempts on a door candidate, and thus stay in the vicinity, before targeting a new candidate.

When comparing the ability of the door manoeuvring algorithm to pass through doors with respect to the droplet-only method, the former had 2.7 times more successful passages. Note that for two of the three environments the number of successful passages in the droplet-only case was zero. For the other environment (environment 3) the door manoeuvring algorithm had twice as much successful passages (56 versus 28).

120 simulations where performed to identify the effect of the camera's field of view of the ability to locate and manoeuvre trough the door. Field of view angles of 40, 50, 60, 70, 80, and 90 degrees were tested. To be able to perform these tests the droplet properties and simulation settings should be adjusted for each field of view angle. In Figure 10 the results of these simulations are presented. For each of the settings the total number of successful door manoeuvres was counted. The results were then normalized with respect to the best performing setting. A clear optimum is present at a field of view of 70 degrees. Notably one can observe there is only a small performance difference between a field of view of 50 and 60 degrees. A possible explanation for these results is that a larger field of view has a larger view of the room and therefore a dominant false positive will be dominant in more situations. However, a too small field of view will limit the cases in which the room exit is in view and decreases the ability to make an approach with a high success probability. Also the optimal number of bins used for identification of the door might differ per field of view setting, this could be further investigated.

As the effectiveness for a field of view of 70 degrees was highest, p-values with respect to the 60 and 80 degrees where computed. The bootstrap method is used, because of the non-parametric nature of the data and the relative limited population size.[18] For both yields p-value < 0.01.

F. Combined Behaviour

All above discussed modules combined result in the complete exploration algorithm. Which, besides of the different exploration phase modules, also exists of

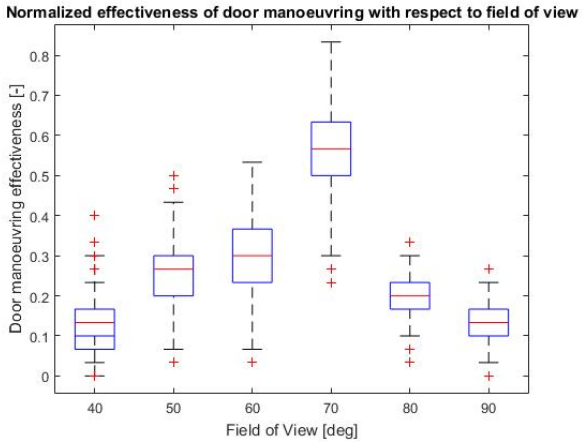


Figure 10: Normalized mean bootstrapped simulated door manoeuvring effectiveness with respect to the field of view of the stereo-vision camera.

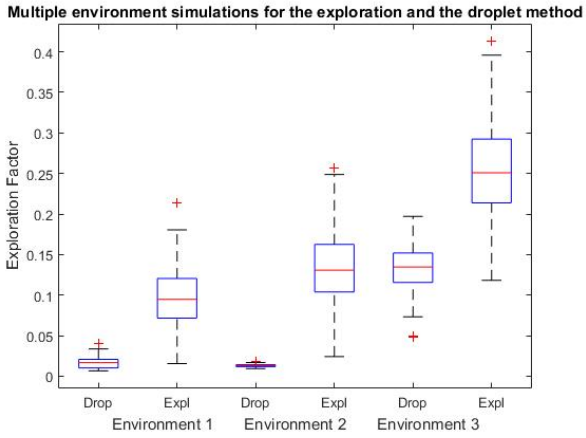


Figure 12: Mean bootstrapped average exploration factor, for over 60 simulations, of the droplet-only and exploration method. Performed in three different simulation environments.

overall modules such as the room detection. Figure 11 shows a successful track of a ten minute simulation of the combined behaviour.

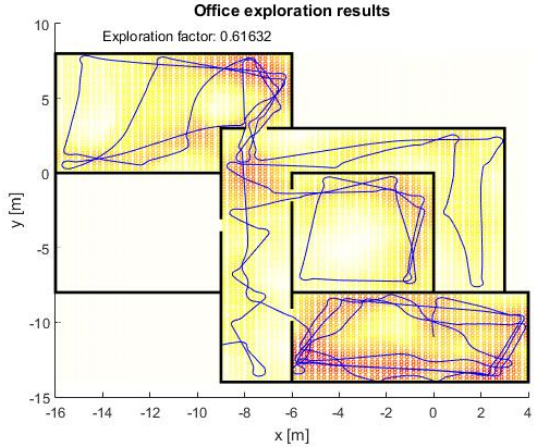


Figure 11: DelFly Explorer 600s simulated flight track for the exploration algorithm.

Results of the exploration method consisting of the combined modules is compared to the old droplet-only method, for all three simulation environments. Figure 12 shows the results. For all three environments the observed difference between the methods had a p-value < 0.01.

Figure 12 shows that for all three environments the performance of the exploration algorithm exceeds the performance of the droplet-only method. Furthermore, one can notice that the deviation in obtained results for the exploration algorithm is higher. This might be due to the fact that, for example in door manoeuvres, a slightly other approach can be the difference between failure (not passing through the door) or success (passing through the door). In case of a failure the approach should be tried again and thus will result in a lower performance. From Figure 12 it can also be seen that some environments, such as environment 3 (Figure 4), have a good layout for a random exploration approach such as the droplet. This is because the droplet-only method easily achieved a relative high exploration factor.

V. Real Flight Experiments

During the real flight experiment the developed modules will be tested on DelFly Explorer in conditions similar as real flight application. The basis modules, stereo-vision altitude estimation, wall-following, and room detection, were tested. The real flight experiments are performed as proof of concept. For future statistical significance of the algorithms in real flight experiments more tests need to be performed.

A. Stereo-vision Altitude Estimate

The first experiment concerned the algorithm which uses the stereo-vision camera for determining the flying altitude of the DelFly Explorer. First tests showed that the exact algorithm of the simulation resulted in a large amount of noise and unreliable outcomes. A solution in which for both the lower and upper part of the image the estimated height of all textures was sorted, was tested. This could then be used to determine the actual altitude based on the 10% of highest/lowest textures. However sorting these values was a relative large computational demand with

respect to the obtained results, as a noisy frame is more apparent than a large amount of consecutive noisy frames. Therefore the method was changed to evaluate the estimated altitude based on the last 40 computations. The resultant would now be the moving average of the data. The computation algorithm was running at 40 Hz on the stereo-board while the stereo camera runs on 24 Hz. This means every second a new independent estimation is given based on 24 frames and some computations use the same frame.

Static Test

The test setup consisted of a number of pre-measured altitudes over which the DelFly Explorer was varied by hand while being moved through the environment. The view of the camera was not obstructed by walls or obstacles. Both the estimated floor- and ceiling-height were registered over time. One of these tests is showed in Figure 13.

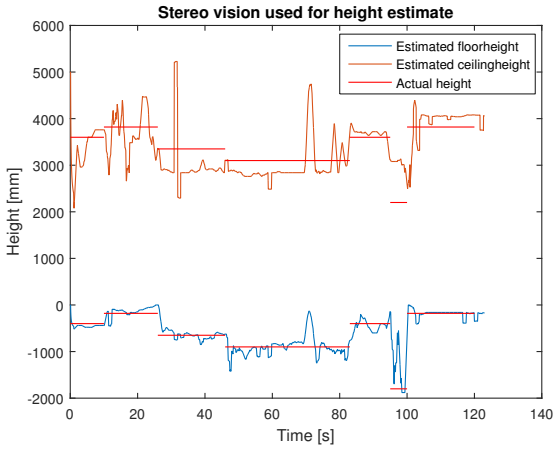


Figure 13: Graph showing estimated and true altitude of ceiling and floor with respect to the DelFly Explorer.

From this figure it is clear that the estimations are, especially for the floor height, quite accurate for the majority of the time. T It might be due to the fact that the floor was closer to the drone and had more textures due to objects standing on the floor. The average root mean squared error over the performed experiments is 116 mm for the floor, and 361 mm for the ceiling height.

Dynamic Test

Besides static tests, flying tests were also performed. These flying tests give a better insight in how, for example, vibrations affect the results. The results are obtained under the same conditions in which the algorithm would actually be deployed. The experiment is performed in the Cyberzoo[†], a fenced area of approximately 10m x 10m, in which the 3D location and attitude of the DelFly Explorer can be accurately tracked. As the ceiling of the Cyberzoo is not within the visual range of the camera, the algorithm will, in the dynamic test, only consider the floor height. The altitudes computed with the stereo-height algorithm can then be compared to the accurately monitored height from the static tracking system. Furhtermore it could be tracked this way whether an obstacle was present in the field of view of the DelFly Explorer. A typical result is given in Figure 14.

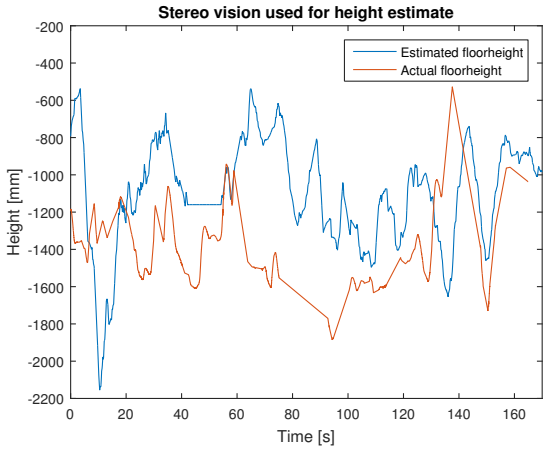


Figure 14: Graph showing estimated and true altitude of the floor with respect to the DelFly Explorer.

The obtained average root mean squared error over the performed experiments is 532mm. Which is a factor 4.6 higher than during static tests. Due to the extra vibrations and pitch angle changes while flying. Further research could investigate hardware design adjustments or algorithm changes to decrease these effects and thus increase altitude reliability.

B. Wall-Following

The wall-following test is again performed in the Cyberzoo. Here the ability of the algorithm to follow the walls and obstacles present in the area is evaluated. Also the droplet-only method is tested as a null test to determine the performance increase of the wall-following method. This algorithm was programmed on the stereo-board. For both methods the paths were recorded, two 2D representations of these paths are visible in Figure 15.

As visible in this figure the wall-following algorithm is better capable of following the contours of the cyberzoo. A more active steering behaviour is visible, indeed ensuring better wall-following. The

[†]http://robotics.tudelft.nl/?q=news/cyber-zoo

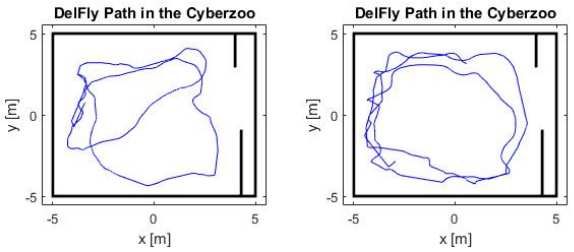


Figure 15: Experimental path results for the droplet-only (left) and the wall-following algorithm (right).

performance is determined by the average distance from the wall. For the wall-following algorithm this is 1.96m on average and for the droplet it is 2.50m, obtained over 3 tests per algorithm. From this it can be concluded that the wall-following algorithm works successfully.

C. Room Detection

When wall-following is correctly implemented the route can form the basis for detecting the contour of the room. By means of this contour a room size estimation can be calculated. Moreover, reoccurring locations help identify when a room has been explored. As explained in Section D for this method a dead-reckoning system and magnetic field properties are used. This algorithm was programmed on the autopilot. In the experiment the DelFly Explorer was flown alongside the boundaries of the Cyberzoo for three rounds. The first round consist primarily of identifying points, based on turning, used for matching consecutive points. During the experiment the pilot aimed to keep speed and pitch constant, as these parameters are assumed to be constant by the algorithm. During flight turns could be matched based on the odometry position estimate and/or the magnetometer orientation. These matches were then saved. The position of the DelFly Explorer, recorded via optitrack, was then matched to these points. In Figure 16 the results are visualized. Only the green points, where both the odometer and the magnetometer had a match, would have made it through both matching filters.

Analysis of this single test showed that in 62.5% the magnetometer matches also had a odometer match. For odometry this percentage was 50% with respect to the magnetometer. Looking at the locations of these matches is that they are all located near the boundaries of the Cyberzoo, which also corresponds to the flown route. Furthermore no clear unbalance in matched locations is visible from the tests.

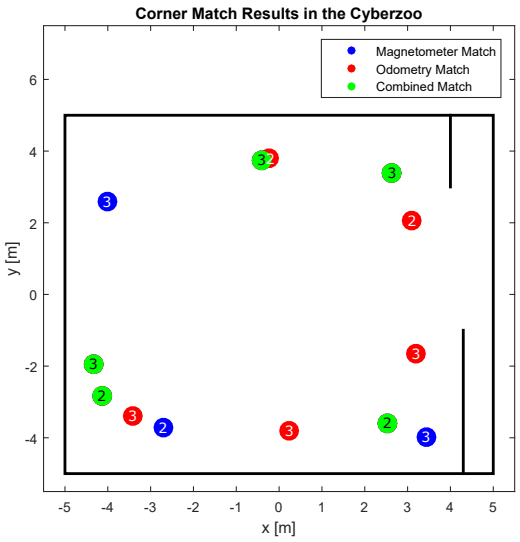


Figure 16: Results for correct odometry, magnetometer, and combined match locations during clockwise flight in the Cyberzoo. Numbers indicate in which round the points were generated.

VI. Conclusions and Recommendations

In this paper a new computationally efficient exploration method for the stereo-vision equipped DelFly Explorer is presented. In order to make the DelFly Explorer autonomous.

The droplet based wall-following method proved to be an adequate method to follow the contours of a room in a simulated environment. Moreover, the door-manoeuvring algorithm increased the ability to pass through doors with a factor of 2.7 with respect to the previous method. These methods performed well in simulation and resulted in a large increase in the exploration factor for indoor environments up to a factor 10 with respect to the previous droplet-only method. It can be concluded that the indoor exploration capabilities indeed increased with this new method.

The static stereo-vision altitude tests have shown to be able to estimate the flying altitude of the DelFly Explorer with a root mean squared error of 116mm with respect to the floor. Performance decreased in the dynamic test to 532mm. At this moment the algorithm could only be used with a moving average filter to calibrate the barometer after certain observation times, but not instantaneously as preferred. The wall-following algorithm performed better than the previous droplet-only method. With decreasing the average distance to the wall from 2.50m to 1.96m.

This however is at the cost of more controller inputs and thus a decrease in flight time.

Concerning the software development, improvements could be made regarding the door-manoeuvring algorithm. When analysing all the simulation runs, this is the part that often took the longest time. A possible addition could be to loiter in front of a possible door candidate to do several attempts before selecting a new candidate. Also solutions in which the flight speed will be adjusted to decrease the droplet size could be investigated to potentially increase the success-rate of the algorithm. When hardware changes to the DelFly Explorer platform are possible, the optimum field of view could be achieved by changing the lens and the number of disparity-map bins should then be optimized for this setting. Furthermore, the algorithm could be more robust for real flight applications, for example with live speed estimation instead of assumed speed. Then this algorithm in combination with wall-following can be more thoroughly tested. For the altitude estimation research in hardware design adjustments or algorithm changes could be performed to decrease noise effects and thus increase altitude reliability. Finally the door manoeuvring should be added to the real flight experiments to test the algorithm in real flight as a whole.

References

[1]Gamini-Dissanayake, M. W. M., Newman, P., Clark, S., Durrant-Whyte, H. F., and Csorba, M., "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, 2001, pp. 229–241.

[2]Angeli, A., Doncieux, S., Meyer, J. A., and Filliat, D., "Incremental vision-based topological SLAM," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, 2008, pp. 1031–1036.

[3]Angeli, A., Doncieux, S., Meyer, J. A., and Filliat, D., "Visual topological SLAM and global localization," *IEEE International Conference on Robotics and Automation*, Kobe, 2009, pp. 4300–4305.

[4]Fraundorfer, F., Heng, L., Honegger, D., Hee Lee, G., Meier, L., Tanskanen, P., and Pollefeys, M., "Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV," *IEEE International Conference on Intelligent Robots and Systems*, Vilamoura, 2012, pp. 4557–4564.

[5]Schauwecker, K. and Zell, A., "On-Board Dual-Stereo-Vision for the Navigation of an Autonomous MAV," *Journal of Intelligent & Robotic Systems*, Vol. 74, 2014, pp. 1–16.

[6]Shen, S., Michael, N., and Kumar, V., "Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV," *Proceedings - IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 20–25.

[7]Zohaib, M., Pasha, S. M., Javaid, N., and Iqbal, J., "Intelligent Bug Algorithm (IBA): A Novel Strategy to Navigate Mobile Robots Autonomously," *Springer's International Multi Topic Conference*, Jamshoro, 2013.

[8]Taylor, K. and LaValle, S. M., "I-Bug: An Intensity-Based Bug Algorithm," *IEEE International Conference on Robotics and Automation*, Kobe, 2009, pp. 3981–3986.

[9]de Croon, G. C. H. E., O'Connor, L. M., Nicol, C., and Izzo, D., "Evolutionary robotics approach to odor source localization," *Neurocomputing*, Vol. 121, 2013, pp. 481–497.

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

[11]Verboom, J. L., Tijmons, S., de Wagter, C., Remes, B., Babuska, R., and de Croon, G. C. H. E., "Attitude and Altitude Estimation and Control on board a Flapping Wing Micro Air Vehicle," *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, 2015, pp. 5846–5851.

[12]Keennon, M., Klingebiel, K., Won, H., and Andriukov, A., "Development of the Nano Hummingbird: A Tailles Flapping Wing Micro Air Vehicle," *AIAA Aerospace Sciences Meeting*, No. January, Nashville, 2012, pp. 1–24.

[13]MAVlab, "The DelFly project," 2015.

[14]de Wagter, C., Tijmons, S., Remes, B. D. W., and de Croon, G. C. H. E., "Autonomous Flight of a 20-gram Flapping Wing MAV with a 4-gram Onboard Stereo Vision System," *IEEE International Conference on Robotics & Automation (ICRA)*, No. Section II, Hong Kong, 2014, pp. 4982–4987.

[15]Lucas, B. D. and Kanade, T., "An Iterative Image Registration Technique with an Application to Stereo Vision," *International joint Conference on Artificial Intelligence*, Vancouver, 1981, pp. 674–679.

[16]Tijmons, S., de Croon, G. C. H. E., Remes, B. D. W., de Wagter, C., Ruijsink, R., van Kampen, E. ., and Chu, Q. P., "Off-board processing of Stereo Vision images for Obstacle Avoidance on a Flapping Wing MAV," *PEGASUS - AIAA Student Conference*, Prague, 2014, pp. 1–16.

[17]de Croon, G. C. H. E., Perçin, M., Remes, B. D. W., Ruijsink, R., and De Wagter, C., *The DelFly*, Springer, Dordrecht, 2016.

[18]Mooney, C. Z. R. and Duval, R. D., *Bootstrapping: A Nonparametric Approach to Statistical Inference*, Sage University Papers, quantitati ed., 1993.

Part II

Introduction

Chapter 2

Introduction

The DelFly Explorer is an autonomous flying, dragonfly-like, micro aerial vehicle (MAV). It is the first flapping wing MAV (FWMAV) which is able to perform autonomous flight, while it is just equipped with very basic sensors and technology, and only weighs 20 grams [de Wagter et al., 2014]. The DelFly is developed by TUDelft's MAVLab [1] with the aim to autonomously explore unknown areas and be able to evade any obstacles. It can then be used for various applications, such as search and rescue in collapsed buildings but also as friendly fairy in an amusement park [MAVlab, 2015]. As one can imagine the different possibilities for such a light weight FWMAV are almost unlimited. However the current state of technology is not yet sufficient to fulfill these tasks, as can be read in Chapter 2-1.

The project discussed in this document aims to aid the development of technology to reach a point where the DelFly Explorer could be used for such tasks. As the name of the DelFly Explorer clearly states; its task is to explore. The objective of this project is to increase the 'explored' area of the DelFly by means of keeping track of the explored and the to be explored areas, based on which routing decisions will be made. With routing the author means a algorithm to (based on obtained information and/or pre-programmed behaviour) decide in which direction the DelFly Explorer will fly. Furthermore this also includes the ability of the DelFly to determine when an entire room has been observed. Pathways, such as doors and windows, to proceed to another room or hallway are exploited by a manoeuvring algorithm. This manoeuvring algorithm will provide a manner to use the room exits. How this is done will be determined during the project.

In the first part of this thesis the goal of the project, the related research, and the final research questions will be discussed. This will form the basis of the second part, the simulations. The simulations part will discuss how the simulation was designed and which modules play key roles in the final algorithm. Each of these modules and also the final result will be evaluated. Part of this evaluation will be comparison with the situation beforehand and how well the research objective is met. In part four the working concept of several modules will be tested in real situations. Modules such as the stereo height, wall following and odometry a brought to the real platform. The results of these experiments are primarily preliminary. In the last part the final conclusion will be presented. This conclusion will be followed by recommendations that should be considered when continuing this research.

[1] http://mavlab.lr.tudelft.nl/

2-1 Related Research

For the research objective introduced several different research areas are of interest. As the target of the project is to develop new software for the DelFly Explorer, it is very useful to investigate the background and hardware of the DelFly family, as done in Section 2-1-1. As one of the primary sensors of the DelFly Explorer is its stereo vision camera, there is need for visual footage analysis, as discussed in Section 2-1-2. The goal of the project is to introduce routing capabilities (a logical approach to cover and explore area, based on obtained information and/or pre-programmed behaviour, and decision making for the general flying direction of the DelFly Explorer), therefore also routing algorithms need to be investigated, as done in Section 2-1-3. When by means of the routing and vision algorithm an opening to another room is found, a manoeuvre has to be planned to go through this opening. Manoeuvring algorithms are discussed in Section 2-1-4.

2-1-1 DelFly

As said the project has as final goal to create new, useful, software to increase the exploration capabilities of the DelFly Explorer. The DelFly Explorer is one of the latest designs of the whole DelFly family [de Croon et al., 2009] [Verboom et al., 2015]. The MAVLab, who is the creator of the DelFly, says that *"One of the goals of research on micro air vehicles (MAVs) is to arrive at insect-sized MAVs that can fly autonomously in complex environments"* [de Croon et al., 2009].

The DelFly is a MAV of ornithopter design, using propulsion by means of flapping wings. The DelFly is very comparable with a dragonfly, as can be seen in Figure 2-1. Verboom et al. [2015] describes the DelFly's use of flapping wings as propulsion. This propulsion method is very versatile. It provides a good basis for forward flight, hovering and gliding when necessary. This in contrast to the current conventional designs (such as fixed wing aircraft and heli/multi-copters). The main materials used for the the wings of the DelFly was a Mylar foil with a thickness of 6 μm. For the DelFLy I and II the same material was used for the tail. The presence of a tail ensures the passive stability of the DelFly, in case of a minimal forward velocity. This needed forward velocity creates the effect of a turn radius, whereas a tailless designs such as the Nano Hummingbird [Keennon et al., 2012], which can rotate in-place, has a higher manoeuvrability. For the fuselage of the DelFly's use is made of a carbon tube. The DelFly I,II, and Micro all had a rudder on the tail, the DelFly Explorer has ailerons instead. These ailerons are mounted on the fuselage directly behind the wings, as can be seen in Figure 2-1.

Still a lot of research is performed in the aerodynamics, morphology and kinematics of flapping wings [de Margerie et al., 2007]. de Margerie et al. [2007] provided in their research a thorough analysis on how to increase efficiency of the flapping wing micro aerial vehicles (FWMAVs) by using artificial evolution on bird-like parametrized morphology. The artificial evolution algorithm tested combinations of wing size, shape, and movement with a general aerodynamic analysis, due to which design could easily be disregarded or improved. Multiple solutions provided by the artificial evolution algorithm were tested in simulation. They were able to achieve an average forward speed of 10-12 $\frac{m}{s}$ with a mechanical cost of 15-20 $\frac{W}{Kg}$, which is comparable to real birds and much better than current performance of FWMAVs.

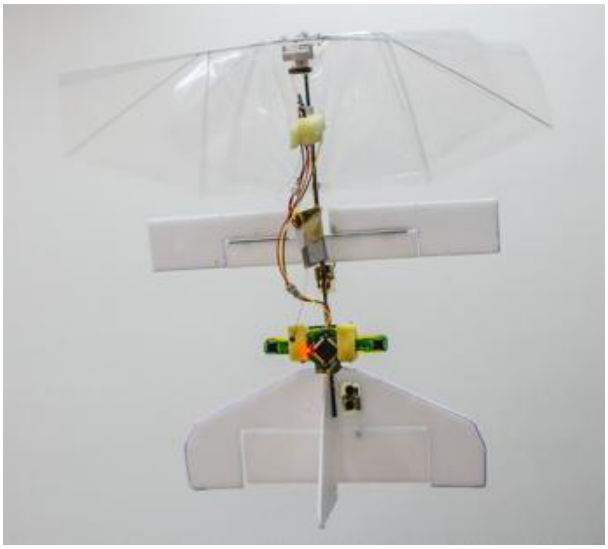


Figure 2-1: Photo of the DelFly Explorer, its wingspan is 28cm, the weight is 20 gram, the stereo-board is located above the tail.[MAVlab, 2015]

Besides the MAVLab, also other institutes are researching the possibilities FWMAVs. DARPA[2], part of the U.S. Department of Defense, is one of these institutes. In 2011, in cooperation with AeroVironment, they unveiled the Nano Hummingbird [Keennon et al., 2012]. This Nano Hummingbird was designed to take the actual appearance of a hummingbird. Furthermore the size, weight and speeds are comparable to that of the DelFly Explorer, this makes it a suitable benchmark to compare performance. The major difference between these two platforms is that the Nano Hummingbird had, besides active stability, no autonomous capabilities. Due to the active stability system the Nano Hummingbird is easy to stabilize despite lack of a tail, which would be needed for passive stability.

The FWMAV developed by Baek et al. [2011] is more advanced in the development of autonomous capabilities. The FWMAV is capable of detecting an infra-red source by means of its integrated infra-red camera. The system constantly evaluates the pitch and heading angle of the source by means of sensor data and dead-reckoning. This information is then used to create a desired attitude change for the FWMAV, which is used by the controller to adjust its flying direction. When the source (target location) is 'out of view', the system is capable of determining the angle under which the source will be (note: the system does not include obstacle avoidance). To do this only the dead-reckoning system is used. A prediction for the heading and pitch angle is created and the path is updated accordingly. Figure 2-2 represents the results of tests performed with this platform, and as can be seen, the UAV is very successful in autonomously reaching the target source. This system was purely designed for one single task and can not complete it autonomously in not ideal situations (when there are obstacles). However, tasks such as these can be part of an autonomous system.

[2]http://www.darpa.mil/

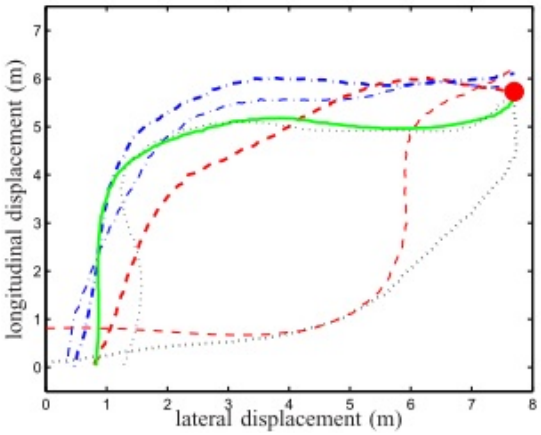


Figure 2-2: A set of trajectories of ornithopter flights toward the target. The dot on the upper right corner represents the target. [Baek et al., 2011]

This short overview of different FWMAVs gives more insight in the current development state. The major trend in the DelFly family is to stretch the boundaries of what is possible in multiple directions. Reduce the size, like the DelFly Micro, or increasing autonomous capabilities, such as in the DelFly Explorer. As the DelFly Explorer is the target platform for the current project, it is essential that the hardware limitations are known. The DelFly Explorer is equipped with a ATmega328P - MLF28 micro-controller. 3-axis accelerometer, magnetometers, gyrometers, and a barometer are part of sensors of the DelFly Explorer. The DelFly Explorer is capable of two-way communication with a ground-station. The battery of the DelFly Explorer provides energy for ten minutes of flight time. The primary sensor on the DelFly Explorer is a stereo-vision camera with a usable resolution of 128x96 pixels on a baseline of 6 cm located just above the tail, as visible in Figure 2-1. The two data-streams of the cameras are received by a complex programmable logic device (CPLD), the CPLD merges the grey-scale components of the images into a single stream. On a STM32F405 processor (168MHz, 192kB RAM) the video stream-data can be analysed [de Wagter et al., 2014]. In the following sections it will become clear that, for now commonly used vision and routing algorithms, more computational power is required than the DelFly Explorer can deliver. Finding a solution within the current DelFly Explorer hardware bounds is therefore a challenging project.

2-1-2 Vision Algorithms

As was learned from the previous section, the DelFly Explorer is equipped with a stereo vision camera as primary sensor. This will induce the need for vision algorithms to analyse the obtained data. Vision algorithms come in a wide variety. Several of these vision algorithms that might prove interesting for this project are discussed in this section.

One of the main vision algorithms is optic flow [Longuet-Higgins and Prazdny, 1980]. Optic flow is the apparent flow of objects in the visual scene of the camera view due to relative motion of the camera and the object, the algorithm uses consecutive images and determines the motion of different points between the time instances. To do this optic flow is depended upon texture, therefore large planes with the same colour can be hard to detect. This is a disadvantage for optic flow, but the same disadvantage is present for also other methods that depend on the detection of texture, some of these will be addressed in this section as well. Besides the needed texture in the environment the camera should also move with respect to the environment. This is needed to create the apparent flow. When using optic flow in a yaw movement, the information obtained is subject to a lot of noise. Schauwecker and Zell [2014] used a downward camera besides the normal front facing camera. When in a yawing movement the front facing camera gives a lot of noise. However, the downward facing camera can use optic flow to give a good estimation of the yaw rate and improve its state estimation. de Croon et al. [2012b] found a way to combine optic flow with appearance variation to increase the detectability of obstacles. The appearance variation cue works on the principle of a lower entropy of textures when an obstacle is nearby. The algorithm selects from a set of small patches (called textons) and compares them to a pre-set database. Based on this database the textons are placed in certain bins in a histogram. Calculation the entropy of this histogram gives the appearance variation of the image. A low entropy corresponds most times with an object in close range and therefore a possible collision danger.

Above mentioned methods are mainly applied to monocular cameras, whereas the DelFly Explorer is equipped with a stereo-vision camera. Stereo-vision cameras have two cameras with a certain offset placed on a baseline (for the DelFly Explorer just above the tail, see Figure 2-1). With stereo-vision cameras a so called disparity map can be constructed [Lucas and Kanade, 1981]. Such a map shows the relative shift of the pixels for each of the two frames of the stereo camera, and via that way an estimate of the distance is obtained. There are several methods to create such a disparity map such as Block Matching [Koschan et al., 1996], Dynamic Programming [Ohta and Kanade, 1985] [Birchfield and Tomasi, 1999], and Semi-Global Matching [Hirschmüller, 2008]. These methods are already tested and compared for the DelFly II by [Tijmons et al., 2014]. It should be noted that this method was performed off-board and is therefore not yet fully suited, as the goal is to make the DelFly Explorer fully autonomous in exploring multiple rooms and corridors.

Besides the construction of disparity maps stereo-vision can be applied to estimate the 3D-coordinates of objects and points in view [Lucas and Kanade, 1981]. These locations in 3D space could then be used to construct maps, which will be further explained in Section 2-1-3. Maps is not the only application, planning of a path to exit a room via door or window can be used. This topic is further discussed in Section 2-1-4.

In the previous paragraphs the methods of optic flow and stereo-vision were discussed. Hrabar et al. [2005] managed to combine these two methods. It was found in research that also insects combine these two sources of information for evading obstacles and for navigation [Weber et al., 1996]. Therefore this was an interesting combination to conduct further research in, as nature often already provides solutions. The results that Hrabar et al. [2005] achieved are visible in Figure 2-3. The method used is as follows. Both the optic flow and stereo-vision algorithm are running simultaneously on the robot. Each of the algorithms provides a turn-rate for the robot. By default, the turn-rate of the optic flow algorithm is used. Whenever within a certain threshold an obstacle is detected, the stereo-vision turn-rate is preferred. As seen in Figure 2-3, this method greatly improves manoeuvre capabilities.

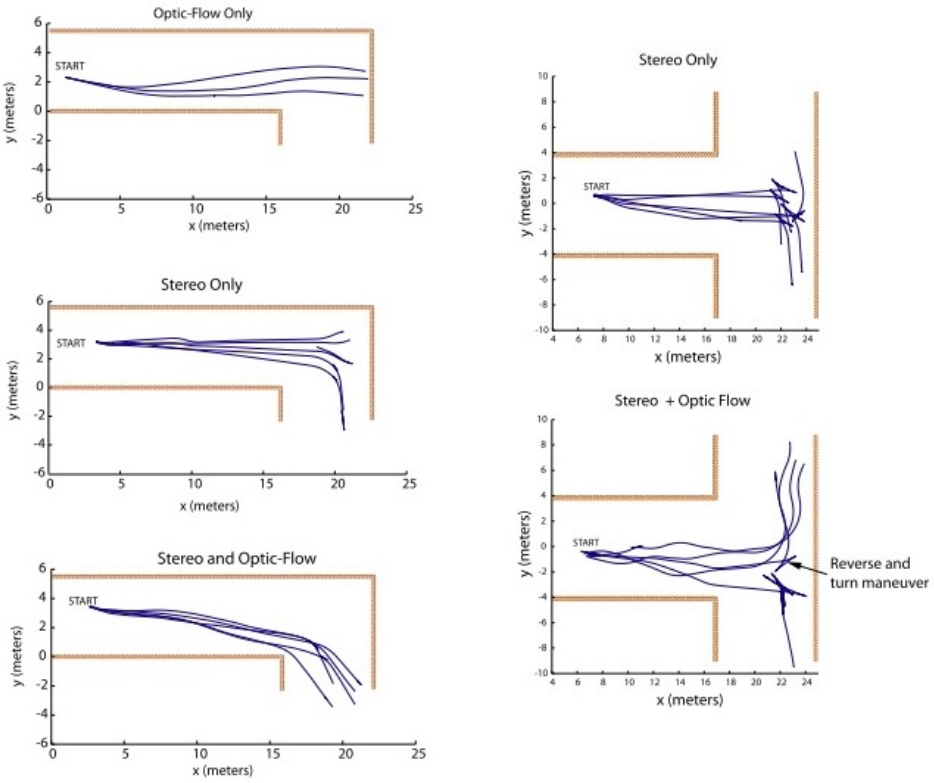


Figure 2-3: Trajectory paths for navigating L- and T-junctions based on different (combinations) of methods. [Hrabar et al., 2005]

Bills et al. [2011] noted that navigating through a environment, by creating a complete model of that environment, is very computationally expensive. This is due to the fact that the algorithm should process a lot of input and put it all within the right perspective, which is constantly checked by and used to update the model of the environment. Instead of this computationally heavy approach a simpler approach was constructed. They identified the main components of an office building (corridors, staircases, and rooms) and constructed classifiers for these. The classifiers are used to easily recognize the building components, these components can be used to create a topological map. A similar method to these classifiers is the so called 'texton method', which was already mentioned before. This method evaluates portions of the images with a database of small texture samples [Varma and Zisserman, 2003] [de Croon et al., 2012b], which makes the method computationally efficient. The textons can, besides appearance variation, be used to identify certain objects or building components, as was done in Bills et al. [2011].

The vision algorithms above still need some computational effort. Decreasing this computational effort can be done by setting regions of interest within the image, or by sub-sampling the images. Sub-sampling makes algorithms more suitable for small platforms [de Croon et al., 2012a]. Sub-sampling works by means of taking only parts of the original image to be analysed. With only these parts the to-be analysed data is decreased. The algorithm still might be able to identify an object, although it has less information. As one would understand, there is a trade-off between computational efficiency and accuracy of the algorithm.

In this section several vision algorithms have been discussed. These methods can be applied to different aspects of the research problem. One method might prove useful as input for the routing algorithm, while the other can easily be used to recognize earlier visited parts. As the DelFly Explorer is equipped with a stereo-vision camera, a good course of action would be to make use of this functionality. Possibly in combination with optic flow methods which yielded results when making turns [Hrabar et al., 2005]. With a stereo-vision camera a disparity map can be constructed and such software is already available for the DelFly Explorer. Recognition of earlier visited locations by means of visual cues is most likely necessary in this project to create good exploration capabilities. Textons may be a method worth investigating for this purpose.

2-1-3 Routing Algorithms

A logical approach for the DelFly Explorer is necessary to explore area, based on obtained information and/or pre-programmed behaviour, and to make decisions for the general flying direction. Such an algorithm will be called a routing algorithm. The hardware limitations of the DelFly Explorer have been discussed, also several vision algorithms have briefly been discussed. The vision algorithms form the basis of many routing and mapping algorithms. The mapping algorithm tries to make a map (all sorts of accuracy levels are possible) of the area. With this map an algorithm can determine a path to a location/object of interest. First these algorithms are discussed which base their routing on the construction of a map. Succeeded by the Mapless algorithms.

Map Based Methods

The most advanced mapping algorithm currently available is SLAM (simultaneous localization and mapping) [Gamini-Dissanayake et al., 2001]. SLAM works by creating a map of the envi-

ronment by means of on-board sensors. Simultaneously it determines the location of the robot within that map. The SLAM problem can be formulated by means of Eqs. 2-1 to 2-3. Where m_t represents the map of the environment at a certain time, o_t the observation, and x_t the position of the robot in the environment. Application of Bayes' rule results in Eq. 2-2, where Λ is Bayes' factor.

$$P(m_t, x_t|o_{1:t}) \tag{2-1}$$

$$P(x_t|o_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t|x_t, m_t) \sum_{x_{t-1}} (x_t|x_{t-1}) P(x_{t-1}|) m_t, o_{1:t-1}/\Lambda \tag{2-2}$$

$$P(m_t|x_t, o_{1:t}) = \sum_{x_t} \sum_{m_t} P(m_t|x_t, m_{t-1}, o_t) P(m_{t-1}, x_t|o_{1:t-1}, m_{t-1}) \tag{2-3}$$

Many SLAM algorithms make use of detecting a landmark within the environment. This landmark can be used to determine the relative change in location of the robot together with other sensory information. This relative change with the previous location and the state observed from the new location can be used to update the map even further [Davison and Murray, 2002]. If one would only use an inertial measurement unit (IMU), the error would integrate over time and become exponentially larger. The landmark method decreases this error that integrates over time and can be used to 'reset' the IMU.

Another implementation of SLAM is topological SLAM. In contrast to metric mapping as happens in metrical SLAM, a topological map creates nodes for discrete locations (or landmarks) and connects them with neighbouring relations [Angeli et al., 2008]. The edges of the nodes link them according to their similarity or distance. Unexplored edges of the topological map represent areas that still can be explored. This is information is necessary for an exploration algorithm. Also for topological SLAM it is necessary for the system to detect whether the current location is an already visited location. If so, the system should combine the current node with the earlier established node, so called 'loop-closure'. A diagram used by Angeli et al. [2008] for this process is provided in Figure 2-4. Topological SLAM is due to its nature less computationally heavy than metrical SLAM.

A major disadvantage of SLAM is that it is relatively computationally heavy. For example, the required hardware for the study of Shen et al. [2011] was a 1.6GHz processor and 1GB RAM. They used it for a quadcopter mapping an indoor office environment. Furthermore the majority of SLAM algorithms are dependent on sensors other than a camera (such as laser range finders) which are not equipped on the DelFly Explorer [Gamini-Dissanayake et al., 2001] [Shen et al., 2011] [Blanco et al., 2008]. Sensors such as laser range finders are still to heavy to be equipped on the DelFly Explorer. As comparison the laser range finder used by Shen et al. [2011] weighs 370 grams (UTM-30LX) [3] compared to the 20 gram DelFly Explorer.

[3] https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html

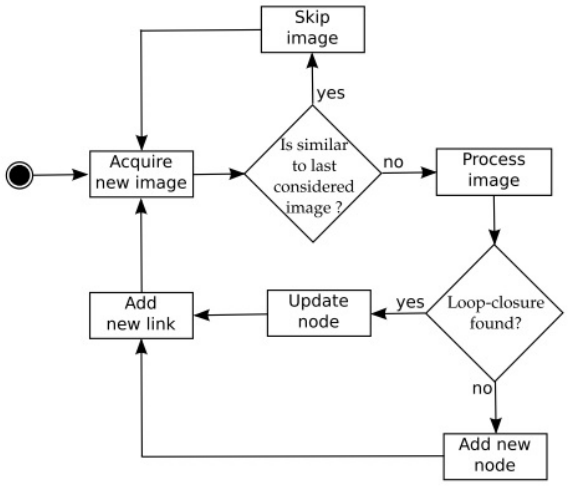


Figure 2-4: Overall SLAM process of the used topological SLAM algorithm used by Angeli et al. [2008].

When only using the camera, visual odometry is mostly used. Visual odometry works by matching features between subsequent camera images to determine the movement of the UAV [Howard, 2008]. The output of a visual odometry algorithm is thus the change of position of the UAV with respect to the location at which the previous image was obtained. However using solely a (monocular) camera, perceptual aliasing increases the difficulty of matching different images. Perceptual aliasing is that sequential images are too similar to obtain any useful information from them. For example, when flying within a corridor the walls tend to all look the same along the corridor, therefore the visual odometry algorithm might conclude there is no movement at all.

Visual odometry is often used in combination with SLAM, as done by Angeli et al. [2008]. During their experiment they were able to map the environment and "close the loop" with 202 nodes within a computational time of 2m58s, whereas the flight time was 5m10s. This proves the method can be easily run real-time, but still a good processor is demanded. A computer with a 2.33GHz processor was necessary. Howard [2008] performed a similar experiment where the computational load required a computer with 2.4GHz of processing power, supporting results obtained by Angeli et al. [2008]. Nister et al. [2004] conducted visual odometry tests, already on stereo-vision cameras. Which are present on the DelFly Explorer. They were also able to achieve good loop-closures, thus proving the method can be suitable for stereo-vision camera platforms.

In a way similar to topological SLAM, Fraundorfer et al. [2012] applied a frontier-based exploration algorithm on a quad-copter. This algorithm builds an occupancy grid (see Figure 2-5) and determines the unexplored frontier. This occupancy grid clearly indicates the boundary of the already discovered area, similar to the edges of the nodes in a topological SLAM map. For the construction of this occupancy grid visual odometry was used, already explained earlier in this section.

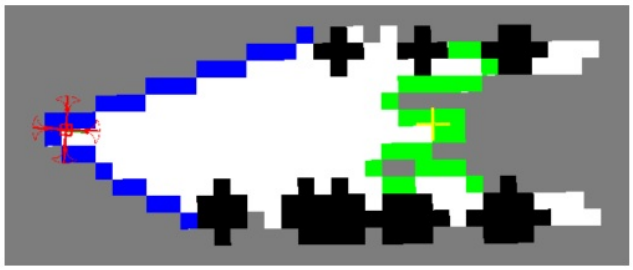


Figure 2-5: 2D slice of initial occupancy map. Unknown cells are colored as grey, occupied cells are black, and free cells are white. The blue frontier behind the MAV is designated as the home frontier. [Fraundorfer et al., 2012]

Mapless Methods

Fraundorfer et al. [2012] also applied a wall-following exploration strategy. This is an exploration strategy part of the family of BUG-algorithms, based on insect behaviour. Insects are a major source of inspiration. Several BUG-algorithms have been developed for finding the most optimal route from a source to a target. A BUG-algorithm does not have a global model of the world. From the source it travels a straight path towards the target. When an obstacle is met it follows the contours of the obstacle. When the path is clear again, it will continue to travel in the direction of the target location Zohaib et al. [2013]. A lot of different BUG-algorithms have been developed. How some of these BUG-algorithms handle an object is represented in Figure 2-6.

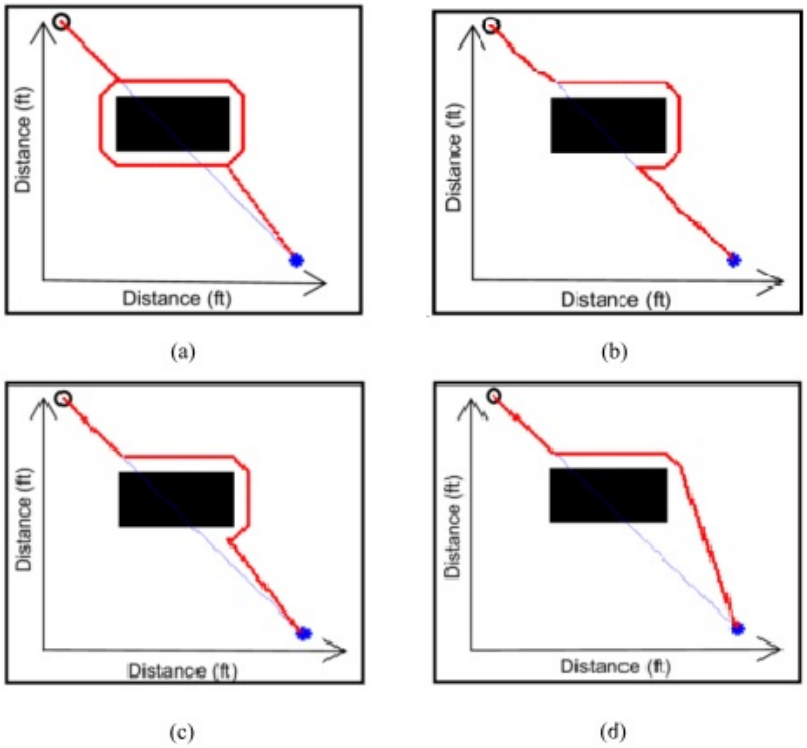


Figure 2-6: Trajectories of (a) Bug-1 algorithm (b) Bug-2 algorithm (c) Dist-Bug algorithm (d) IBA. [Zohaib et al., 2013]

BUG-algorithms can also be intensity based. This means that the robot senses an intensity of a signal, such as an odor-source [de Croon et al., 2013] or radio signal [Taylor and LaValle, 2009]. Within such problems the target location can be determined during the process based on the sensor input. When the direction of the target is determined objects can again be evaded by the several methods, including the ones shown in Figure 2-6. Within the current project a target or the direction towards a target is not always known. The DelFly Explorer may not yet be aware of such targets as they may still be unexplored.

The Pursuit-Evasion BUG-algorithm is a method that could prove applicable for this research as it tries to explore as much area as possible [Rajko and LaValle, 2001] [Tovar et al., 2003]. The main benefits are that there is no predefined target or map needed for the algorithm, only the ability to follow a wall and detect interruptions in the wall (indicating a corridor or a non-visible area). This also means that the sensing requirements of the robot are easily achieved. The task of this algorithm is to clear the environment from moving evaders. This means that when the robot cleared an area it should ensure it doesn't get contaminated again. Figure 2-7 represents a situation with this algorithm. The robot is located in the middle of a cave like area. The white parts indicate the visible area, from which a discontinuity map can than be constructed indicating the directions of unidentified areas, called gaps. As the DelFly Explorer is not capable of detecting all directions and depths at once, a derivation of this method would be necessary.

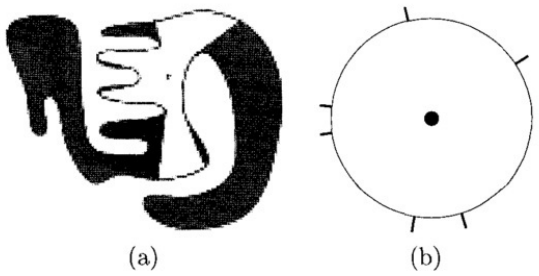


Figure 2-7: Discontinuities in depth measurements partition the set of viewing directions. [Rajko and LaValle, 2001]

The Pursuit-Evasion BUG-algorithm assumes that the robot has the ability to detect the heading of those gaps with respect to it's own attitude. Gaps can merge, meaning that the robot moves is such a way that the entry of two area's become one. If one of those areas was not cleared yet, both areas are contaminated again. Also gaps can split, meaning that the robot is going towards those gaps. The Pursuit-Evasion BUG-algorithm uses this fact to keep track of which gaps originated from a initial gap. Similar to the Gap Navigation Tree as used by Tovar et al. [2007]. The Gap Navigation Tree keeps track of which area followed from which gap and can therefore create a map of the environment by connecting zones. The map would be in the form of a tree-diagram connecting which room or area (originating from a gap) is connected to preceding or following room or area.

At first the robot is exploring the area. Until the envisioning process determines that based on the current environment knowledge the task can be solved (to explore all areas without re-contamination). When exploring a part of the environment the robot moves from the left wall to the right wall while moving forwards. When the part is completed it follows the wall to minimize the number of wasted motions until a new, unexplored part is reached.

Another on insects inspired technique is described by Baddeley et al. [2011]. They use simple Haar-feature detection to give either a positive or negative feedback on a certain view, based on a single episode learned 'waypoints' on the route from the source to the target. Haar-features compare the sum of pixels of different rectangle regions within the image. To increase the efficiency of this procedure use is made of integral images, which can easily sum regions of images [Viola and Jones, 2001].

The recognition of the path by Baddeley et al. [2011] can furthermore be based on the motivational context of the robot. For ants that can depend whether they are looking for the food source or for the nest. Based on that motivation they can recognize parts of the paths to their motivated goal [Baddeley et al., 2012]. And they will not recognize the paths that are within their memory, but do not fit the motivational context. To ensure that the needed memory does not increase as the route becomes larger, a neural network was used. This neural network was presented with the images along the training path to create familiarity discrimination using an InfoMax learning rule. *"InfoMax learning means that the next control action is chosen according to Bayesian estimation given what we have learnt until now, given the actual state that we have at this moment."* [Jeni et al., 2003]. A similar approach as used by Baddeley et al. [2012] could be applied for the DelFly when it is either exploring the room or looking for the room exit.

Scheper et al. [2015] used the stereo-vision camera on the DelFly Explorer to develop, via an evolutionary robotics approach, a behaviour tree framework. A behaviour tree consist of a finite number of preprogrammed behaviours that are executed when certain threshold conditions are met (multiple behaviours can be running simultaneously). The goal of this behaviour tree was to let the DelFly Explorer autonomously search for and fly through a window. Flying through a room exit will also be one of the requirements in this project. A first approach on the behaviour tree was user designed and consisted of four main sub-behaviours: window tracking, straight flight when disparity is low, wall avoidance when disparity is high, and action hold (so the wall is not evaded when an evasion is already occurring). For this user designed behaviour tree a success-rate of 82% was obtained in simulation. Now a behaviour tree was constructed by means of an evolutionary algorithm. The fitness function was chosen such that the score would increase when coming closer to the window, with a maximum score when the vehicle was able to fly through the window. In simulations performed, the success rate of the optimized behaviour for flying through a window was 88%. The simulation results of the evolutionary algorithm were thus higher than of the user designed behaviour tree. Applying the same optimized behaviour tree in real-world experiments resulted in a success rate of 54%. This number should be increased to obtain satisfactory implementation in a final exploration algorithm [Scheper et al., 2015].

Table 2-1 gives in a concise manner a summary of the discussed routing algorithms and the main pros and cons that are the most influential on this project. As testing, used hardware, and experimental set-ups differs a lot between the method it was chosen to represent it without measurable statements.

Table 2-1: Overview of the reviewed routing algorithm strategies with main pros and cons.

Method	Description	Pros	Cons
Map based			
SLAM	Simultaneous construction of a metric map and location determination	Detailed map	High computational loads, large memory demand
Topological SLAM	Symbolic represented map with connected nodes for locations, more abstract than SLAM	Relative small memory demand, detailed map	High computational loads
Visual Odometry	Matching features of consecutive images to determine the motion of the UAV	Only camera needed, also determines ego-motion	High computational loads
Mapless			
BUG - Algorithm	Insect inspired path following from source to target location with obstacle avoidance	Easy to implement, low computation cost, small memory demand	Often target location needed
Baddeley {et al.}	Path recognition based on Haar-features	Low computation cost, relative small memory demand, use of motivational context	Route should be known
Behaviour trees	A finite tree with preprogrammed behaviours that are executed when certain conditions are met	Low computation cost, small memory demand, can be trained	Relative lower success rate

2-1-4 Manoeuvring Algorithms

As said in the introduction of this chapter the routing and vision algorithms will provide the location of an opening to another room, and the moment when flying trough this opening should be attempted. To attempt such an action a manoeuvring algorithm method might be necessary to increase the chances of success. With manoeuvring algorithm is meant the procedure used to go from the location before the opening, door or window, safely to a location on the other-side to continue exploration from that point on. For this manoeuvring planning several methods were studied in more detail: motion primitives, behaviour trees, random forest, and Zufferey's *et al.* optic flow based method.

Motion primitives work on the principle of describing a limited number of basic movements. These basic movements can be subdivided into two main categories namely a trim state and a transition state (transition between two trim states) [Bottasso et al., 2008]. During flight the drone can use, based on the required action, any of these 'preprogrammed' basic movements. All these movements are collected in a motion library. The main advantage of such a library is that the movements can be optimized in training flights or simulations with respect to a certain cost function [Cohen et al., 2010]. This cost function can then be optimized for whatever the user determines most valuable, for example: time, energy consumption, distance travelled, etc. or a trade-off between such parameters. This method can be very useful for the DelFly Explorer when it has to perform a certain manoeuvre through an opening. As noted by Bottasso et al. [2008] the success of motion primitives depends on the quality of the UAV simulation model to create a good motion library. Furthermore a challenge with this method is being able to detect disturbances in the environment that cause divergence from the planned path. Different methods can be used, such as the already discussed vision algorithms or inertial measurement units.

Another technique is the use of behaviour trees. The working principle of behaviour trees was discussed in Section 2-1-3. Behaviour trees can be trained to increase the success-rate. This method was already applied to the DelFly for flying through a window by Scheper et al. [2015]. As recap: in simulations performed, the success rate of the optimized behaviour for flying through a windows was 88%, while the user designed behaviour tree had a success rate of 82%. The simulation results of the evolutionary algorithm were thus higher than of the user designed behaviour tree. Applying the same optimized behaviour tree in real-world experiments resulted in a success rate of 54%, this number should be increased to obtain satisfactory implementation in a final exploration algorithm [Scheper et al., 2015].

Ferguson et al. [2006] discusses the possibility of using Rapidly-exploring Random Trees (RRTs) for manoeuvring planning. The algorithm has a certain target location (other side of the room exit) to which a manoeuvre has to be found. From the start position random branches are created, at the end of these branches new branches are created until the target is reached, an obstacle is encountered or when a certain threshold is passed to ensure unsuccessful paths are discontinued. Li and Shie [2004] used a random forest approach, consisting of multiple RRTs originating from different locations, creating a forest. The algorithm is able to connect those different trees to increase the probability of a successful manoeuvre. When new obstacles are detected only the affected trees have to be disregarded instead of the entire forest. The random forest approach can be used by the DelFly Explorer to evaluate different trajectories through the room exit.

Zufferey et al. [2007] developed a method to estimate altitude and pitch for a micro aerial vehicle with data provided by an inertial measurement unit, airspeed sensors, and simple vision sensors. They managed to obtain collision free flight with a 10 gram microflyer. The vision system was based on optic flow, already discussed in Section 2-1-2. In this optic flow method, the optic flow difference between the image halves determines the needed movement [Zufferey and Floreano, 2006]. When the optic flow is highest at the right side of the image, then there is probably an obstacle on the right side and the microflyer should turn left. They combined this method with 'saccades'. Saccades are fast turns based on insect movements. During these movements camera images are disregarded. When using optic flow in a yaw movement, the information obtained is subject to a lot of noise. A gyroscope or a downward facing camera can be used to still be able to determine the turn rate. The visual steering used by Zufferey and Floreano [2006] can be applied for the manoeuvring algorithm to fly trough the room exits. This algorithm would run in a constant loop during which the optic flow determines the direction of the room exit and the UAV's heading is adjusted towards it. This optic flow method can maybe be combined with the stereo vision functionalities of the DelFly Explorer. Also a possible combination with behaviour trees can increase the success-rate of this method. For example, different saccades can be pre-programmed that are triggered according to the behaviour tree. This way the algorithm could be able to successfully cope with different situations.

2-2 Research Question

As already introduced in Chapter 2 the main goal of the project is to increase the 'explored' area of the DelFly Explorer by means of keeping track of the explored and the to be explored areas, based on which routing decisions will be made. This to increase the DelFly Explorer's usability in a wide range of scenarios.
Currently the flapping wing micro aerial vehicles (FWMAVs) is a rather new field of research in which still a lot of progress is to gain. As could be read in Chapter 2-1 there is still quite a knowledge gap between the random routing (currently applied in the DelFly Explorer) and advanced navigation methods such as SLAM. At this moment it seems that the project will occur exactly within that gap due to the limited hardware capabilities of the DelFly Explorer. When this project is successful the knowledge gap will decrease, aiding further research in similar topics.

However, due to the fact that there is a large knowledge gap there where this research is conducted it is more difficult to draw conclusion about the feasibility of this research. From the literature study in Section 2-1-3 it became clear that there are already some algorithms that can, may it be in reduced form, implemented in the DelFly which, as these researches describe, can increase the efficiency of the exploration with regard to the current, random, case. With this knowledge in mind there is a large possibility that indeed this project will succeed in increasing the DelFly Explorer's exploring capabilities.

With the topic goals introduced and the short literature review conducted the following research question could be obtained:

How to increase the indoor explored area of the DelFly Explorer by means of computationally efficient routing decisions?

Accompanied with this research question several sub-question for the project can already be formulated.

1. How to structure the exploration process?
2. How do routing decisions affect the path of the DelFly?
3. How to effectively recognize earlier visited locations?
4. Which algorithms are needed to recognize room exits?
5. How is successful manoeuvring through room exits obtained?

Chapter 3

Design Choices

As became clear from the research question and the accompanying sub questions several different tasks will be performed by the algorithm. These combined tasks will results in the overall behaviour of the algorithm. These tasks will be executed by different modules. This is done to ensure optimal programming and performance efficiency. Furthermore the tasks that the DelFly Explorer needs to perform can be subdivided in different stages, or so called *exploration phases*. Exploration Phase 1 will identify the general contour layout of the room by means of wall-following. Exploration Phase 2 will, when the contour of the room is known, explore the remainder. Exploration Phase 3 consist of an algorithm which will identify door candidates. This exploration phase will provide the location of the best candidate to the algorithm of Exploration Phase 4, which will head for the door. These last two phases interchange quickly for better convergence. Finally some overall modules will handle obstacles, odometry, and a topological map. Which modules are active per exploration phase will be discussed in more detail in Chapter 4-2.

Each of the different exploration phases will have, as described, its own set of subtasks. This subtasks can be fulfilled by means of different algorithms. These algorithm will be chosen based on literature, personal insights, operational and platform constraints, and performance. Some of the considered design concepts will be discussed in the coming chapters.

Part III

Simulations

Chapter 4

Simulator

From theory several ideas and methods were derived. To give an indication if a method might be successful in real world situations simulations are an ideal method. In simulations the real world is mimicked with certain assumptions and simplified models. Also simulators can easily perform multiple runs with the same or different settings. This data can be used to determine statistical significance and the sensitivity of the models. Based on literature, hardware constraints, and personal insight a modular system was developed in which each of the modules has a specific task. Within the various modules different approaches to tackle the module's task were tested and evaluated. Later on these modules where then further optimized to improve overall performance. At first the simulation software will be explained in Section 4-1, this software will form the foundation of the simulation and aides in incorporating already performed research. In Section 4-2 will be discussed, here the different modules, their primary task, and how they are connected and with which variables will be showed.

4-1 Simulation Software

SmartUAV[1] is an in-house software package that can be adapted to simulate several unmanned aerial vehicles (UAVs). SmartUAV, developed by the MavLab, contains a library of these different UAVs and their dynamics, layout, in-, and outputs. For the DelFly Explorer this data is already available within the SmartUAV environment. Also SmartUAV builds an graphic environment based on the layouts provided, the used layouts will be further discussed in Chapter 5. By means of the generated graphic environment also camera images for the UAV can be generated.

The programming language used for SmartUAV is C++. Each of the modules that will be constructed will be programmed in this language. Furthermore inputs from and outputs to existing modules are communicated. How these modules are connected in SmartUAV is visible in Figure 4-1. As one can see in this figure a module can generate certain outputs which can then be connected to the input of the subsequent module.

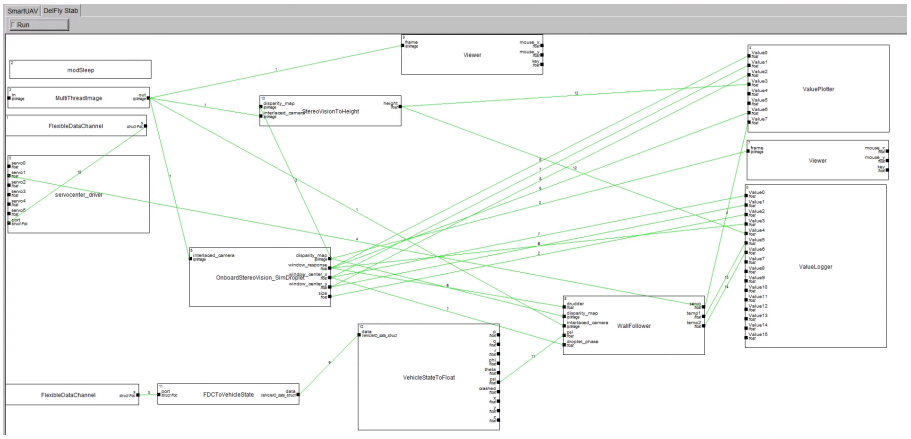


Figure 4-1: Layout of how the modules are connected within the SmartUAV simulation software.

How all these modules are connected is specified in a .xml file. This file is specific to a certain experiment/simulation setting and also contains information about the model, the used environment, the location of the log-file, and other settings. As said the simulation produces log-file, besides the standard SmartUAV log-file the user can also create a simulation specific log-file. In this log-file the value of different variables of interest can be logged. This process is done by the value-logger module, which is also visible in Figure 4-1.

In Matlab a script was developed to plot the route and determine the performance of the simulation. More information on how this performance is determined can be found in Chapter 6. Furthermore several scripts in Matlab will be developed to analyse certain parameters or the performance of sub-modules.

[1] https://svn.lr.tudelft.nl/trac/ADIO-CS/SMARTUAV/

4-2 Simulation Layout

As became clear from the research question and the accompanying sub-questions the algorithm should be able to cope with different aspects. Such as identifying earlier visited locations, being able to have a effective path through the room, and be able to manoeuvre through a room exit. Therefore it was decided to subdivide the algorithm in different exploration phases, each of these phases deals with the primary task at that moment within the entire algorithm. Exploration Phase 1 will identify the general contour layout of the room by means of wall-following. Exploration Phase 2 will, when the contour of the room is known, explorer the remainder. Exploration Phase 3 consist of an algorithm which will identify door candidates. This exploration phase will provide the location of the best candidate to the algorithm of Exploration Phase 4, which will head for the door. These last two phases interchange quickly for better convergence. Finally some overall modules will handle obstacles, odometry, and a topological map.

Each of these exploration phases consist of a couple of functions executed during that phase. Which functions are used and how these are interconnected is shown in Figures 4-2 - 4-5.

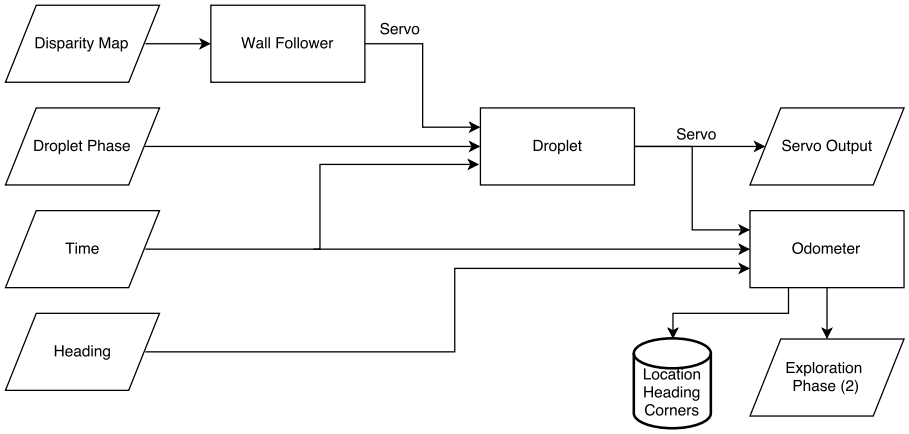


Figure 4-2: Flow diagram of the exploration phase 1.

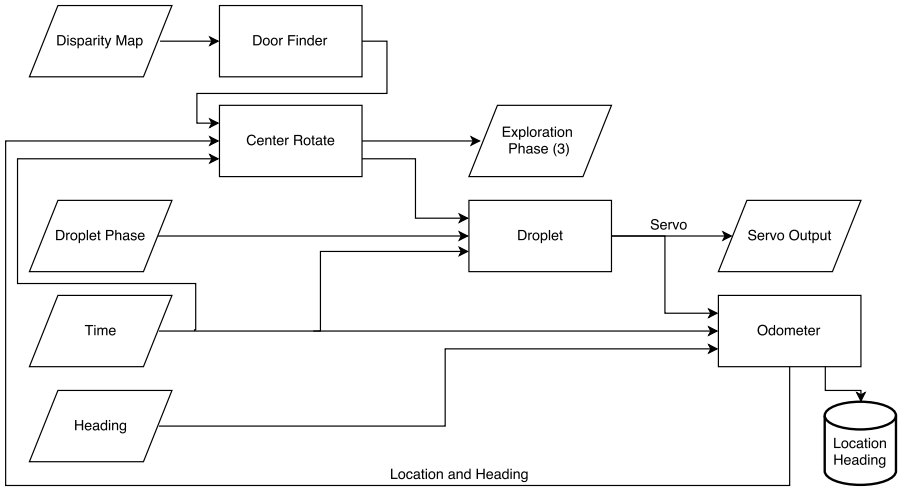


Figure 4-3: Flow diagram of the exploration phase 2.

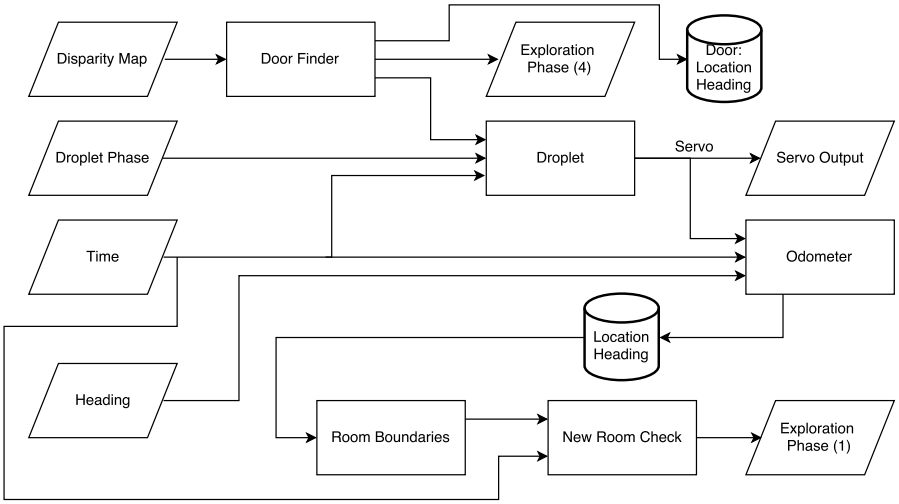


Figure 4-4: Flow diagram of the exploration phase 3.

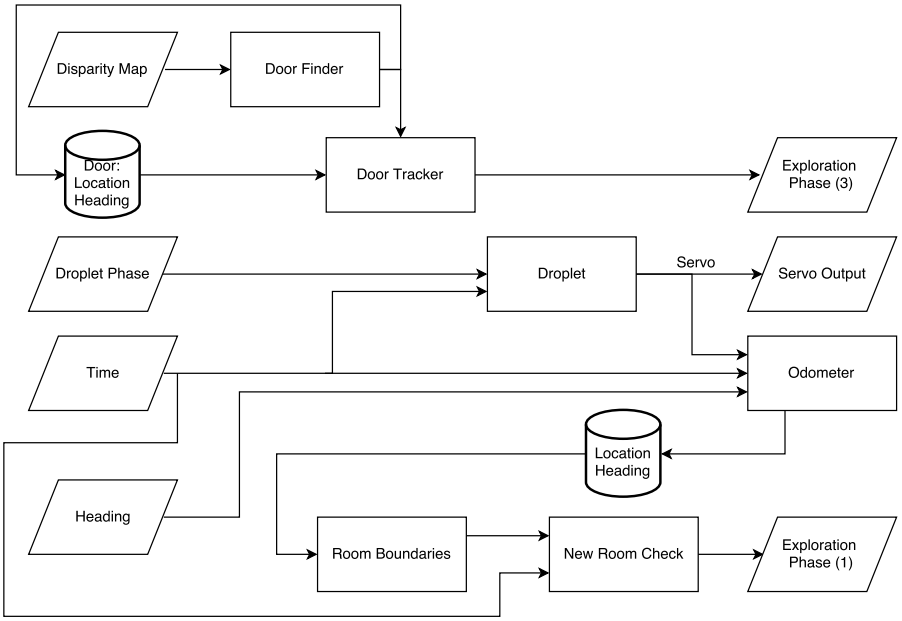


Figure 4-5: Flow diagram of the exploration phase 4.

These different phases, modules, and task combined can also be represented in a finite state machine. A graphical representation of the finite state machine is given in Figure 4-6. The algorithm starts with wall-following, while this procedure is carried out obstacles are being evaded when encountered. This last behaviour is applicable for all phases of the algorithm. After matching three corners by means of odometry the second phase is initiated. In the second phase the DelFly Explorer will, based on the obtained odometry data, guided more towards the centre of the room to make a turn and head towards a possible door candidate. Now phase 3 and 4 are constantly interchanging until a door passage has been achieved. Phase 3 evaluates possible door candidates while phase 4 adjusts the heading to make an optimal approach. When finally a new area is reached, the state will return to wall-following to start the whole procedure again.

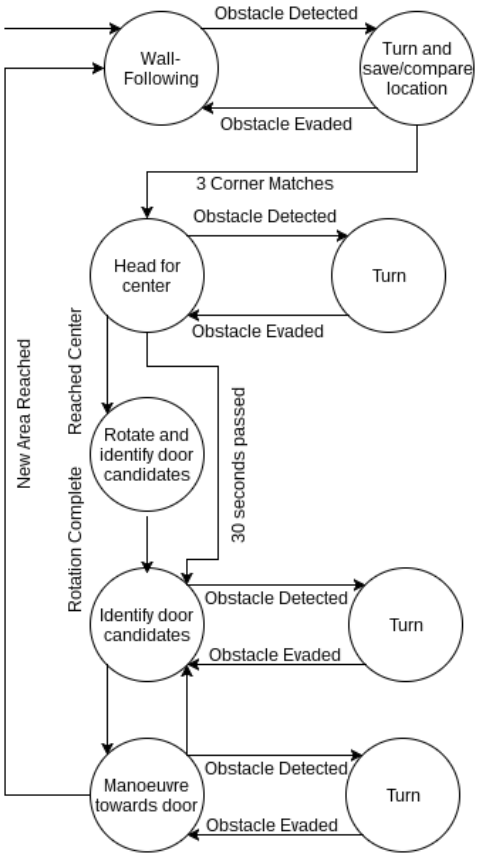


Figure 4-6: Finite state machine representation of the exploration algorithm.

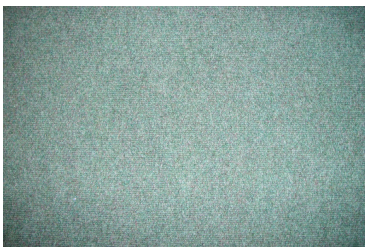
Chapter 5

Simulation Environment

The environment of the simulation is a file containing the layout information of the 3D environment the DelFly Explorer simulation will be performed. This environment is necessary to mimic real world situations and provide the same visual cues as would be obtained during experiment. In simulation it is also possible to test a larger set of differing environments. Further more the behaviour of the DelFly Explorer within the environment can be analysed and based on this the performance of different methods can be analysed.

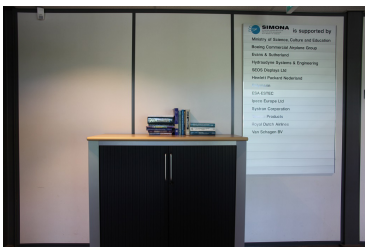
SmartUAV uses a .ac file type as an input for the environment. The .ac file is used to create a 3D model of the environment and contains 3D resources such as meshes, texture paths, and material definitions.

In the .ac file the different faces (such as walls, floors, and ceilings) are specified by means of their corner coordinates. A connection of several of these faces can be a room or a corridor. On these faces textures are placed, often multiple next to each other (depending on the size of the face). Examples of such textures can be found in Figure 5-1. Each sort of face has a typical sort of texture, for the walls of the rooms different textures are used but all of the same nature as can be seen in Figure 5-1c. The textures are important as a major part of the algorithms that are developed are based on the stereo-vision camera and the disparity map.



(a) Floor textur.e

(b) Corridor wall texture.



(c) Typical room wall texture.

Figure 5-1: Examples of textures that are used for the different environment.

5-1 Environment 1

The first environment is representation by its layout in Figure 5-2. It consists of a U-shaped corridor with which four different sized rooms are connected. The U-shaped corridor ensures that the corridor has (when flying to the end and back) both counter- and clockwise turns. Furthermore there is a dead end where the DelFly Explorer could get stuck.

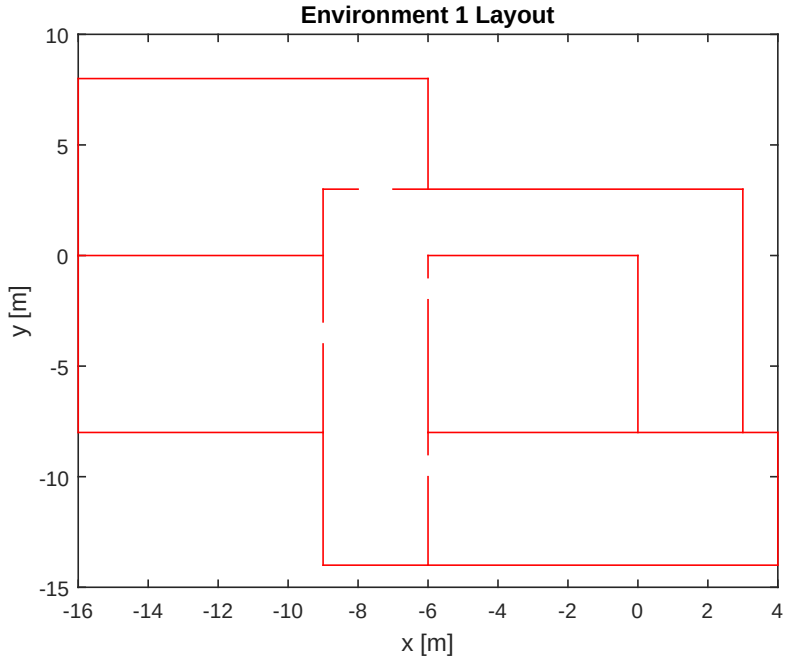


Figure 5-2: Layout of environment 1, the walls are represented by the red lines.

5-2 Environment 2

The second environment is representation by its layout in Figure 5-3. It consists of a large room with a small hallway directly attached to it giving access to three differently shaped rooms. Each of these room has a door on a different part of the general shape.

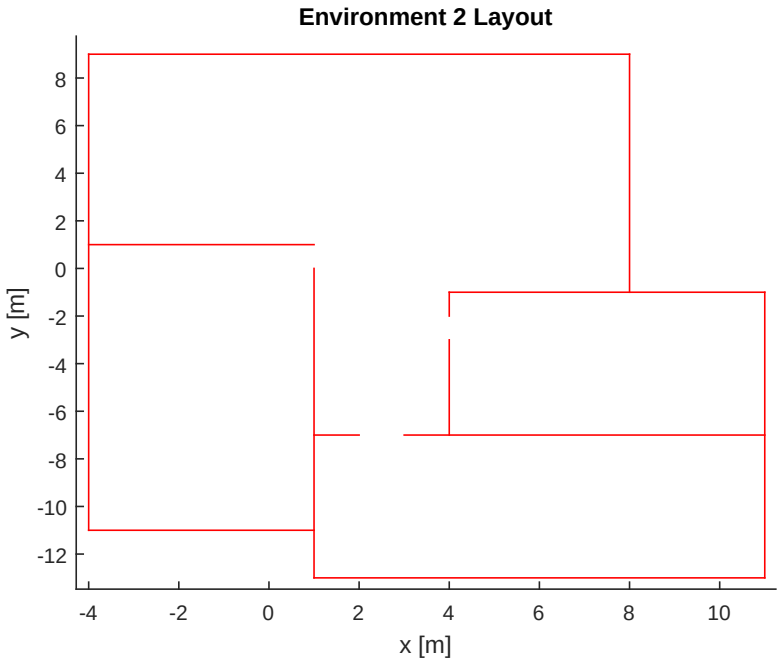


Figure 5-3: Layout of environment 2, the walls are represented by the red lines.

5-3 Environment 3

The third environment is representation by its layout in Figure 5-4. It consists of several offices attached to long corridor with in the middle a coffee corner. This introduces a large central obstacle within the same area of the environment.

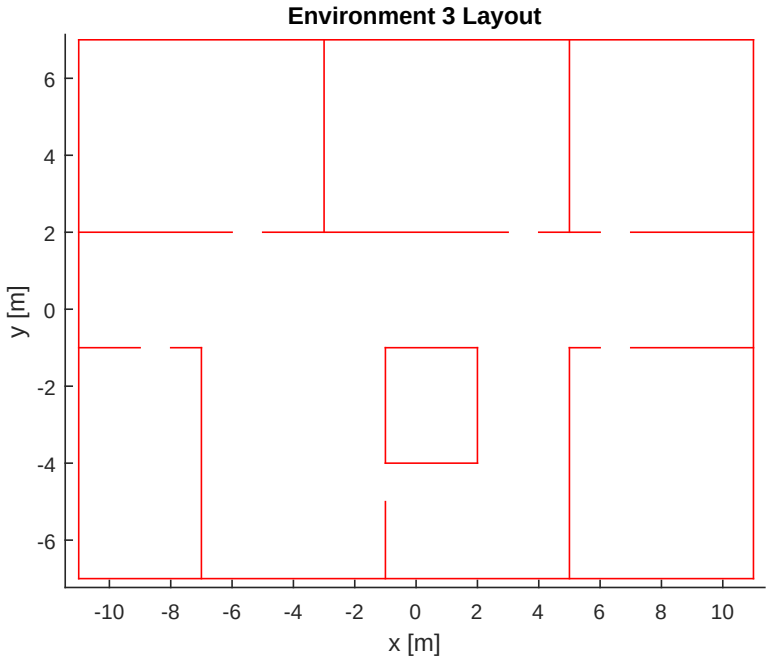


Figure 5-4: Layout of environment 3, the walls are represented by the red lines.

Chapter 6

Exploration Factor

During the simulation phase several different methods and environments will be tested. During development most of the times it is clearly visible whether a methods works or not. But when different methods both work, or when one would like to find the increase of performance with respect to the initial case, a figure of merit is needed. As the main focus of this project is to increase the exploration capabilities of the DelFly Explorer (see Chapter 2) the main figure of merit will also focus on this aspect. Therefore it is called the exploration factor (EF).

For determining the success of the exploration of the DelFly Explorer a number of factors are of importance. First and most important is the total area that is visited. Secondly, one of the sub-goals of this project is to be able to visit multiple rooms. Therefore the ratio of rooms over the total rooms that could have been visited will be incorporated in the EF. Lastly the speed and flying time of the DelFly Explorer will be incorporated. Multiplication of those parameters will result in the total distance travelled, which again with the total explored area can give an insight in the efficiency of the taken route. The final result of the EF can be found in Equation 6-1.

$$EF = \frac{nRooms}{nRoom_{total}} \cdot \frac{m_{explored}}{m_{total}} \cdot \frac{m_{explored}}{V_{average} \cdot t} \tag{6-1}$$

From this equation directly some limitations of the figure of merit are visible. For example if two runs are compared where one of the runs has a much larger total area, that run would have a lower exploration factor due to the fact that it has explored a lower percentage. They other way around however, if it wouldn't be ratio, in the bigger room there would be more exploration possible while in the smaller room exploration is finished already. Therefore it is recommended, that when comparing results, equal or similar total areas should be considered. Similar argumentation can be used for the number of rooms. Furthermore, the total simulation time (t) is, when it differs, compensates in the last term the decreased explored area from the second factor. However, this compensation is not adequate for the number of rooms visited. Therefore it is recommended to keep simulation time constant over the different compared experiments.

6-1 Matlab

From every simulation performed SmartUAV saves a log-file. In this log-file different sorts of data can be logged, also depending on what is considered as useful information for analysis. In MATLAB [1] a script was developed to analyse these log-files. The needed information for the EF was computed or obtained from the log-file. The MATLAB-script has a plan of the different environments. Based on this plan and the DelFly Explorer location in the simulation obtained from the log-file it can be determined which rooms were visited. When besides the location also the heading is obtained from the log-file, the script can computed the explored area. The explored area is the area that is 'seen' by the DelFly Explorer. For this one should still specify the exact visual range and angle of the camera, which is depended on the task of the DelFly Explorer whether one wants to be able to just recognize the environment or really wants to be able to identify people. The final result of this script will be the EF of the simulation run and a figure which visualizes the visited area, as can be seen in Figure 6-1.

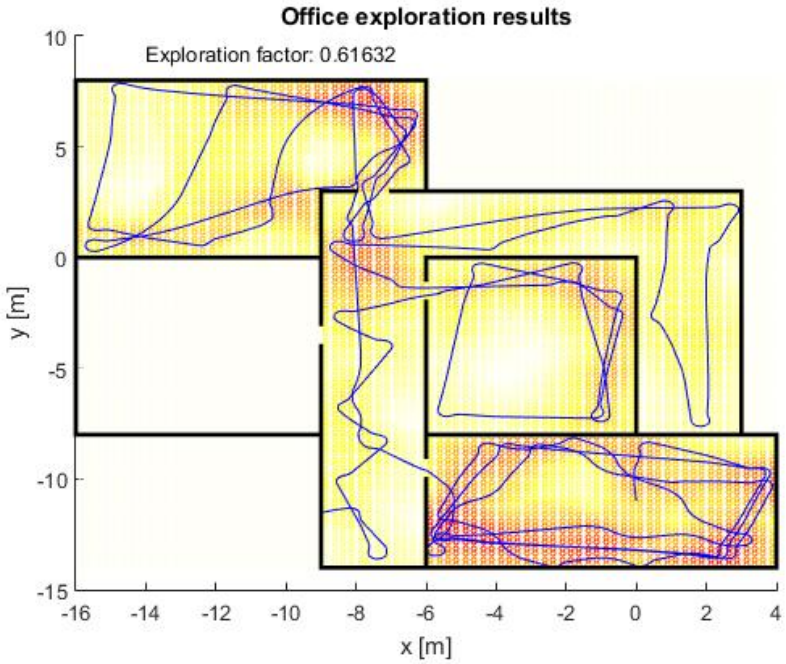


Figure 6-1: Here the typical output of the exploration factor script, showing the route, seen area, environment boundaries, and exploration factor.

Figure 6-1 represents the output of the exploration factor script. The axis of the plot represent the location of the DelFly Explorer. The blue line indicates the taken path during the simulation. The red lines on the figure are the (internal-)boundaries of the environment. As one can see in the figure the different internal rooms & corridors are connected to each other via gaps (doors). Lastly on the background a grid is visible which represents how good certain areas are seen. Blue is the standard color of the grid and represents unseen areas of the environment. A higher local brightness of the grid represents the fact that the specific area means the camera of the DelFly Explorer exposed that area for a longer time.

[1] https://www.mathworks.com/products/matlab/

Chapter 7

Droplet

The environment in which the DelFly Explorer will fly, whether that is in simulation or in real world cases, obstacles will be part of such an environment. Which can be the walls of the room or cabinets and tables. A method should be used to be able to safely and consistently be able to avoid these obstacles. When determining such an obstacle avoidance strategy the constraints of the DelFly Explorer should be taken into account. Such constraints are the turn radius (r), field of view (α), and safety margin (m). Such a method is already developed for the DelFly Explorer by Tijmons et al. [2014]. This method consist of a droplet shaped area in front of the DelFly Explorer, this area will be called 'droplet'. The droplet is visible in Figure 7-1 [de Croon et al., 2016].

The strategy is to keep the area of the droplet, in which the DelFly Explorer can still safely make a full turn, clear of any obstacles. The area of the droplet is within the field of view from the position of the DelFly Explorer, visualized in Figure 7-1. This is the reason why the droplet shape is determined by the turn radius (r), field of view (α), and safety margin (m). The aforementioned constraints also determine the total length (l) of the droplet, which with current settings is equal to 3 meters. The DelFly Explorer is programmed such that when a obstacle is detected the DelFly Explorer will continue it's path until it reaches the turning point. At the turning point the DelFly Explorer starts its clockwise turn. While making the clockwise turn the droplet script is constantly evaluation the new droplet area in front of the DelFly Explorer. When this area is clear the DelFly Explorer will continue straight flight while maintaining a short sensitivity period. In this period an obstacle in the droplet zone will trigger an immediate turn. All these different stages within the droplet protocol are numbered with phases. Normal flight is phase 1, triggered droplet is phase 2, turning is phase 3, and the sensitive period is phase 4. Because in the current configuration the DelFly Explorer will always make a clockwise turn it was determined to put the cameras under an angle β. This optimizes the droplet without need for extra manoeuvres.

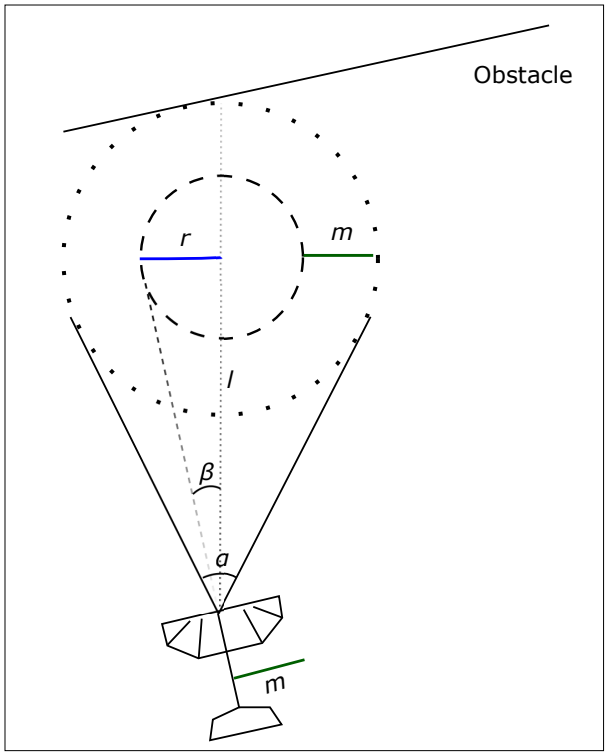


Figure 7-1: Graphical droplet representation with stereo-camera under angle β. [de Croon et al., 2016]

.

The method used to determine if an obstacle in present in the droplet area (and thus the droplet should be activated) is a disparity map. The disparity map is constructed by stereo matching the left and right image of the stereo-vision camera equipped on the DelFly Explorer [Okutami and Kanade, 1994]. Only texture can be adequately matched, non textured areas can not provide a solid match. A sparse disparity map is then constructed showing the relative shift between the two stereo images (with respect to the left image) [Hajebi and Zelek, 2006]. This relative shift provides a cue of the distance of the texture with respect to the DelFly Explorer, a more detailed explanation on how this is implemented can be found in Tijmons et al. [2014]. The disparities are matched by means of a cost function, a best fit of the lowest three results is made and presented as a final result. The final disparity map has a disparity resolution of 96.

Not always is a disparity map sufficient to detect possible obstacles in the droplet. As said in the paragraph above, to construct a disparity map textures are needed. Sometimes these textures are not present, for example when heading straight for an uniformly coloured wall. At first a strategy was used which evaluated the entropy of the stereo-images. When this entropy was below a certain threshold (meaning there was not enough colour variation in the images) the droplet would be activated. However this method proved not to be robust enough. In certain situations and environments there was a large amount of false positives resulting in a large number of unnecessary consecutive droplet activations. A solution for this problem was developed. Instead of evaluating the entropy of the images it is evaluated how many useful textures on the image are present for constructing a disparity map. When the number of textures is below a certain threshold (often due to large uniformly coloured surfaces) for a number of consecutive frames the droplet is activated. This often happens when the DelFly Explorer is close to a uniformly coloured surface, therefore the number of false positives with respect to the entropy method was greatly reduced.

7-1 New Droplet

The old droplet method which could only turn clockwise is no longer optimal for the current project. Rooms and corridors are explored counter-clockwise. Most corridors have a relative narrow layout, due to this fact it is possible the droplet is triggered by the corridor wall on the right while the DelFly Explorer is following the left wall. A clockwise turn would then result in flying back in the direction the DelFly Explorer just came from. Therefore the possibility should exists to make a counter-clockwise turn. This is determined in the wall-following script, further explained in Chapter 8.

Because turns in either the clockwise and counter-clockwise direction should be possible, the camera angles β are reduced to zero, meaning the cameras are oriented along the flying direction. Creating the situation as sketched in Figure 7-2.

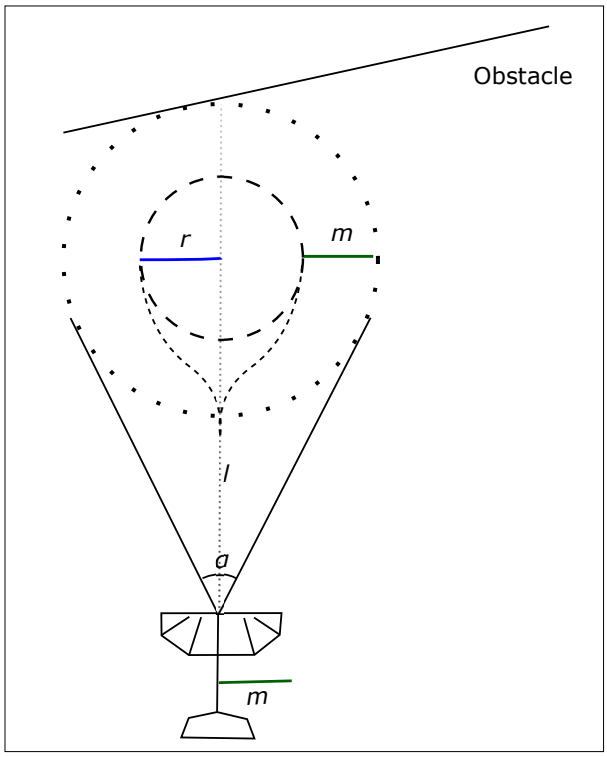


Figure 7-2: Graphical representation of the new-droplet with stereo-camera under pointed in the flying direction, both counter- & clockwise turns are possible.

As visualized in Figure 7-2 the turning circle now lies directly ahead of the DelFly Explorer. This means that at the moment the droplet is activated (phase 2) it should be determined if the DelFly Explorer is making a clockwise or counter-clockwise turn. A small heading change should be made accordingly. This heading change is made halfway the activation of phase 2 and phase 3, this is due to the fact that the turn should be in an area earlier cleared safe by the droplet. These actions are performed by the wall-following script.

The outputs of the droplet script are as follows:

- Disparity Map
- Droplet Phase (1, 2, 3, or 4)

- Droplet Response (0 or 1)

Chapter 8

Wall Following

The wall following script is developed such that rooms and corridors can easily be explored and to aid in the determination of the size of room. The wall following script is programmed to explore rooms and corridors clockwise. Therefore the script tries to keep the wall on the left side of the DelFly Explorer. To do this use is made of a disparity map, which is an output of the droplet script explained in Chapter 7, and the settings of the droplet. The output of the wall following script will be an actuator input, either for correctly following the wall, because the droplet was activated, or to adjust for the corridor direction, which will later on be discussed in more detail.

8-1 Contourlines

A method that was applied for detecting and following a wall was the use of contour-lines. The idea behind this method is that clear texture difference between the wall, floor, and ceiling are very clear. For this method the cvCanny and the cvHoughLines2 method of the openCV library where used. These were applied to the lower portion of the disparity image to clearly identify the contour-lines between wall and floor. For this method other than semi-horizonal lines were filtered out to decrease the effect of other contours. A screenshot of the applied method is visible in Figure 8-1.

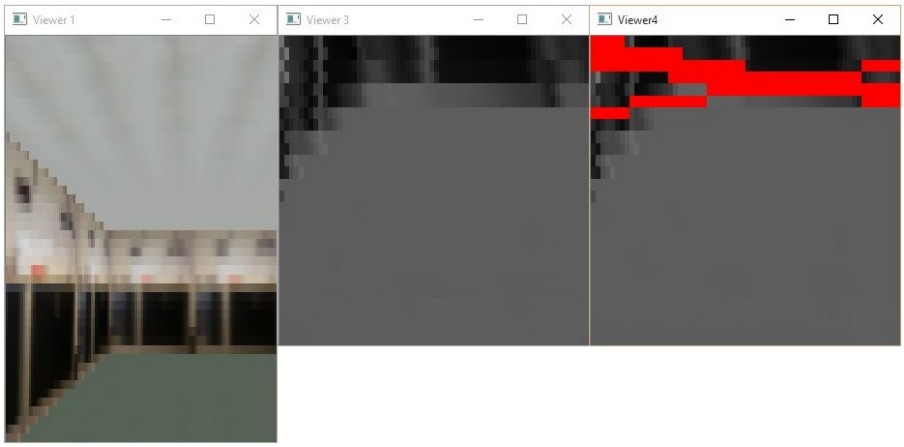
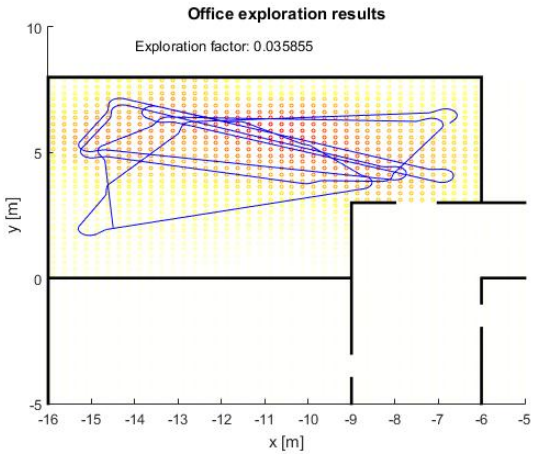
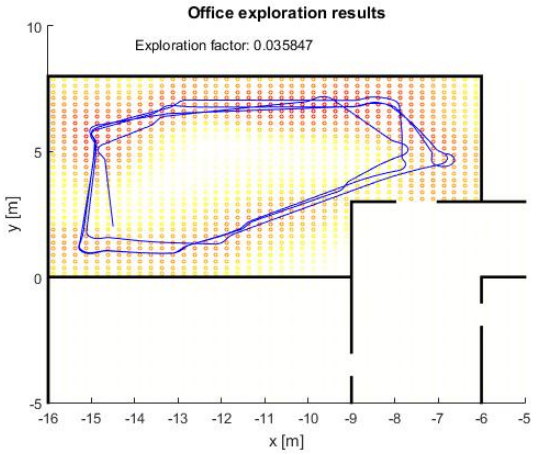


Figure 8-1: Contourlines fitting on a region of interest of the images provided by the DelFly Explorer camera.

However this method has some major drawbacks that became very apparent during simulation. When the room is small or the DelFly Explorer is closing on a wall the floor-wall contour-lines are no longer visible, thus no information about the fall-following. Furthermore, when following a wall, only contour-lines further away could be identified. The effect of this was that the algorithm based on the contour-lines had more difficulty with following the wall directly to its side correctly. These issues might be less apparent with a disparity based method, so this was the next method to be tested and evaluated.



(a) Droplet-only behaviour.



(b) Wall-following behaviour.

Figure 8-2: Simulated flight behaviour of the DelFly Explorer for different settings with a 150 seconds simulation time.

8-2 Disparity Based

The disparity based wall-following algorithm uses the droplet as basis. The droplet counts for each vertical image line the number of high disparities (disparities above a certain droplet shaped threshold). The wall-following has on the left half of the image slightly lower threshold, this ensures, when the wall is to close, a slight heading change to the right. This heading changes is performed before the droplet would be activated. Also when the number of disparities with a certain value or higher on the left half image plane is too low, the wall-following algorithm will steer the DelFly Explorer more to the left (in the direction of the wall) to ensure proper wall-following. The wall-following algorithm is therefore better in following the walls of a room than the droplet only method, as can be seen in Figures 8-2b - 8-2a.

The performance is determined by the average distance from the wall. Over the different simulation the mean average distance for the wall-following algorithm this is 2.03m and for the droplet it is 2.69m, obtained in environment 1. These results are also visualised in Figure 8-3 From this it can be concluded that the wall-following algorithm works successfully.

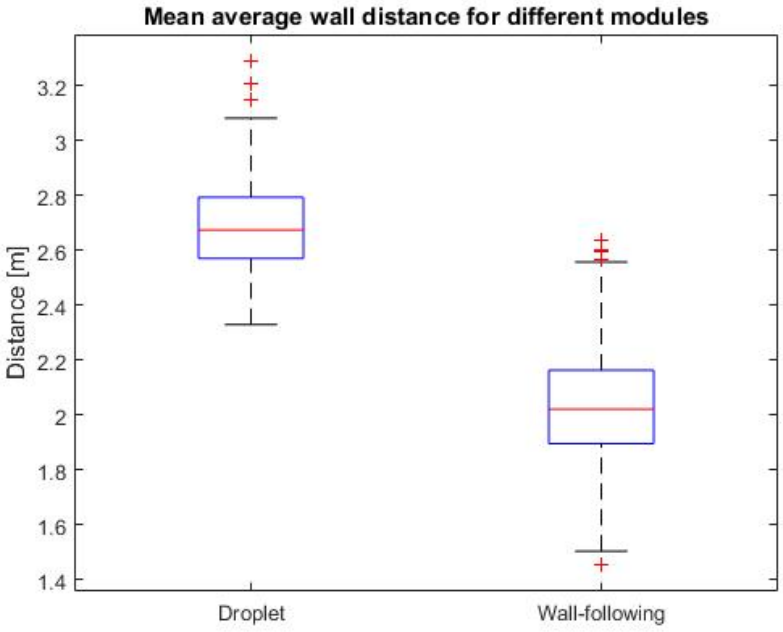


Figure 8-3: Boxplot of the average distance to the room wall for the droplet and wall-following method.

8-3 Corridor Behaviour

A separate, but important, part of the wall-following algorithm is the corridor behaviour. The corridor behaviour deals with when the DelFly Explorer is in a corridor and compensates for overturns, and performs counter clockwise droplets. When heading straight for a wall, but there is no wall directly on the left side, normally the wall-following algorithm would adjust the heading of the DelFly Explorer more to the left, but the corridor behaviour prevents this to ensure a clockwise turn can still be made at the upcoming wall. Furthermore, when in a corridor a clockwise turn, standard for the droplet, would sometimes results in heading back they way the DelFly Explorer came through the corridor. This process would be very inefficient, therefore this part of the algorithm allows in these cases a counter clockwise turn. In Figure 8-4 the result of this corridor behaviour is shown with respect to the droplet only method shown in Figure 8-5.

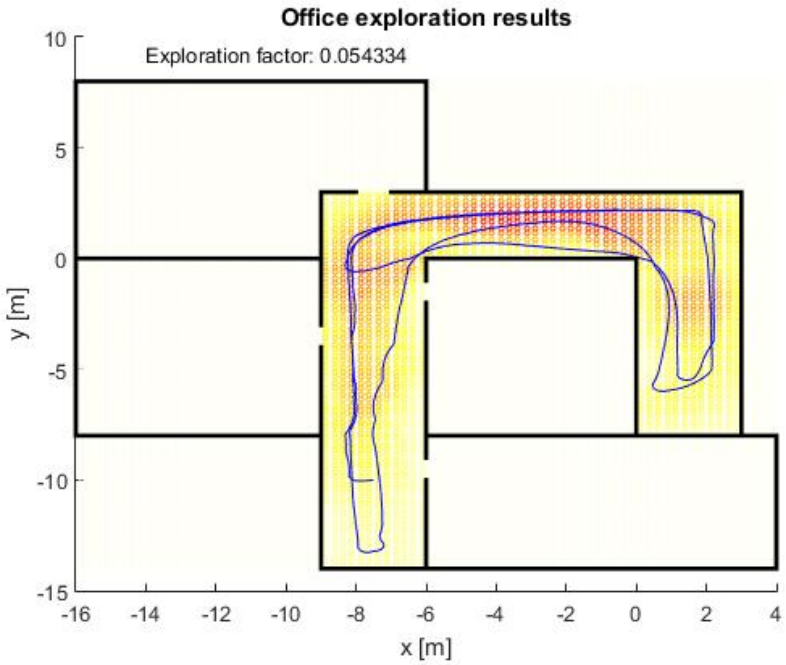


Figure 8-4: Corridor wall-following behaviour of the DelFly Explorer in a corridor.

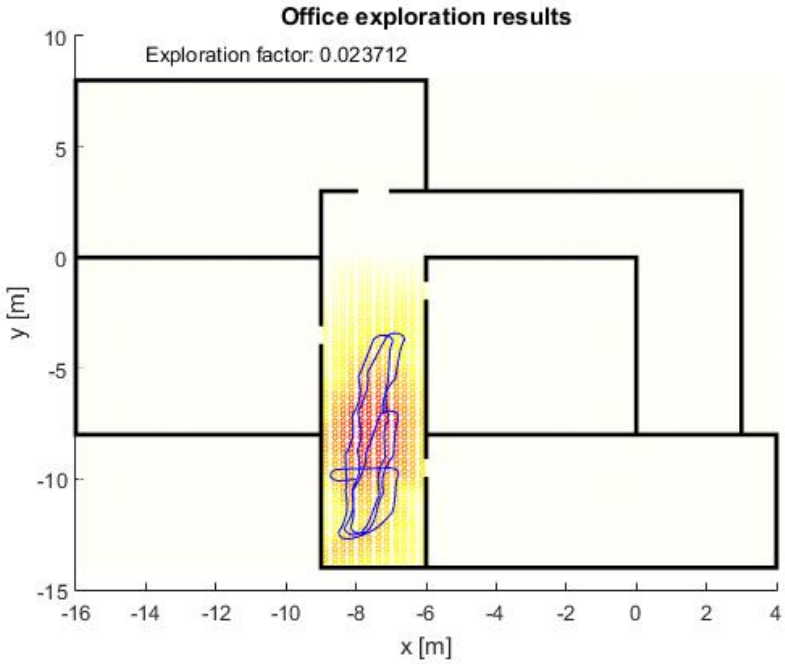


Figure 8-5: Droplet behaviour of the DelFly Explorer in a corridor.

Chapter 9

Room Size Detector

The room size detector is the part of the algorithm that will make an estimation of the size of a specific room/corridor the DelFly Explorer is exploring. The main part is that by means of keeping track of the DelFly Explorer by means of odometry aides in deciding at witch point the DelFly Explorer has followed the wall all the way around the room. When this point is reached the DelFly Explorer might need to make some diagonal crossings to cover the whole room, this is based on the size estimated by the room size detector.

As said the room size and location of the DelFly Explorer are based on odometry. On-board of the DelFly Explorer a magnetometer is present, as discussed in Section 2-1-1. This magnetometer can determine the heading of the DelFly Explorer. In simulation the speed of the DelFly Explorer is fixed, in real world experiments it can also be fixed, but the actual speed is then depended on external factors such as wind. Algorithm such as the droplet and wall-following provide a control output. Internally there is also a clock keeping track of the time.

With each of these factors known the position of the DelFly Explorer can be computed. Each time when there is a heading change, a vector based on heading, speed, and flight time can be computed with respect to the last position. With this vector the coordinates can be determined.

9-1 Corner Identification

Corner identification is an import factor for this algorithm. All rooms are made up of corners connected via walls. Via these corners the size of the room can be determined. Furthermore corners are limited and constant in location, therefore they can be used to identify whether the DelFly Explorer already visited this corner previously and thus have made a full round in the room.

9-1-1 Color Histograms

Whenever the droplet is triggered, a substantial corner is made. The first approach was to, at the moment the droplet would be triggered, save the image histogram. The image histogram contains for each of the primary colours (blue, green, and red) a number of bins, each bin corresponding with a different value for these primary colours. For every pixel the corresponding blue, green, and red value would coupled to a bin and the value for that bin would be increased by one. This process is done for all the pixels in the image resulting in a histogram as can be seen in Figure 9-1. Also in that figure the corresponding camera image is shown.

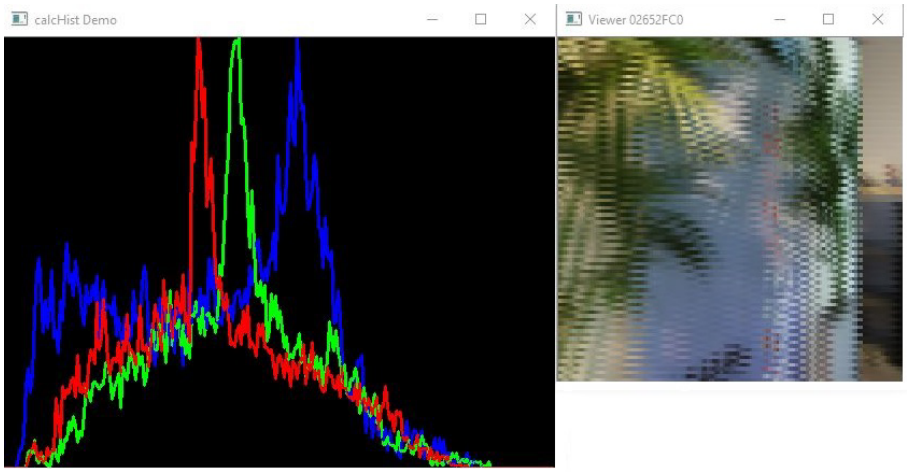


Figure 9-1: Corner approach one and colour histogram.

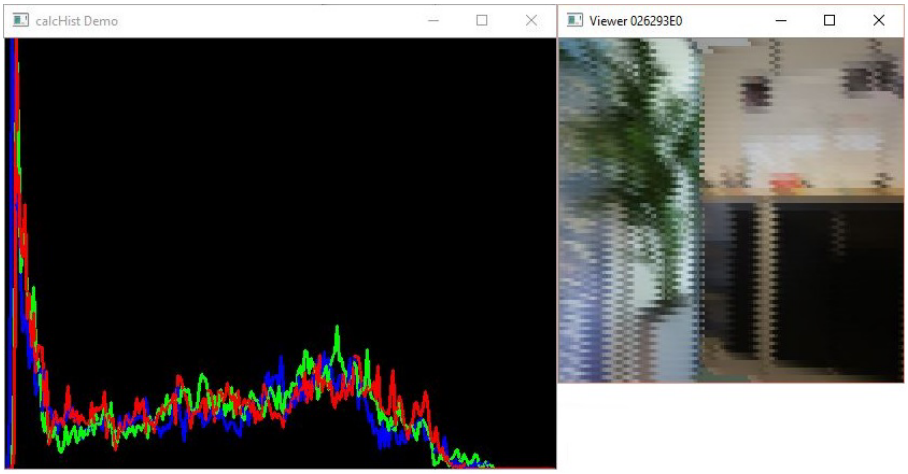


Figure 9-2: Corner approach two and colour histogram.

In Figure 9-1 and 9-2 show an approach to the same corner. As visible from the accompanying colour histograms, there is a large distance in colour distribution and a very low correlation.Therefore the DelFly Explorer would need to visit the same corner very often and save all these different approaches before matches can be made. Therefore it is very difficult to exactly identify the total number of corners. Therefore this method was disregarded.

9-1-2 Odometry Match

Another method is to entirely rely on odometry for identifying and matching corners. As explained in the beginning of this chapter by means of the magnetometer, time, and flying speed the coordinates of the DelFly Explorer can be determined. Whenever in a short time period the heading of the DelFly Explorer, which can be measured with the magnetometer, changes above a threshold these turning coordinates will be considered a corner. For each of these corners different information will be saved: x,y location of the corner, x,y location where droplet phase 2 activated, the inbound heading, the outbound heading, and the heading change.

Whenever these coordinates are within a certain radius of one of the previous corner registered in the database, a match will be made. When multiple of these matches are made an good estimate can be made of the actual size of the room, and provide this information for the next exploration phase. In the next phase, based on the size of the room, the remainder will be explored.

The big assumptions within this odometry simulation module are a constant speed and a well known turning angle. In real world cases these parameters will have a certain uncertainty. This will influence the matching radius implemented in the real world application.

9-1-3 Odometry Performance

When introducing an random error of maximum 5% on the magnetometer in simulation the results in Figures 9-3 - 9-4 are obtained. One can see from these figures that the track has an absolute error of not greater than 2.5m, but on average stays within 1m of the actual position.

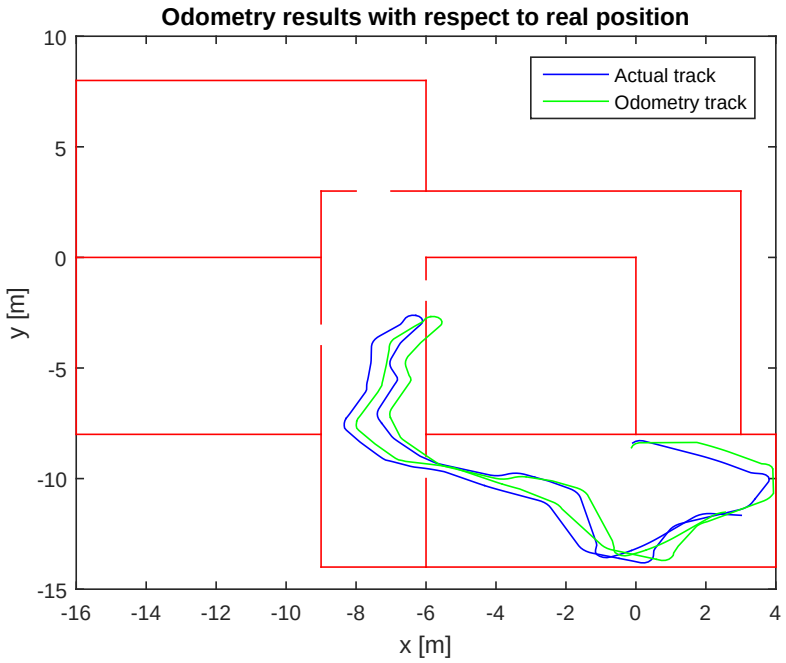


Figure 9-3: Track of odometry algorithm with respect ot actual track.

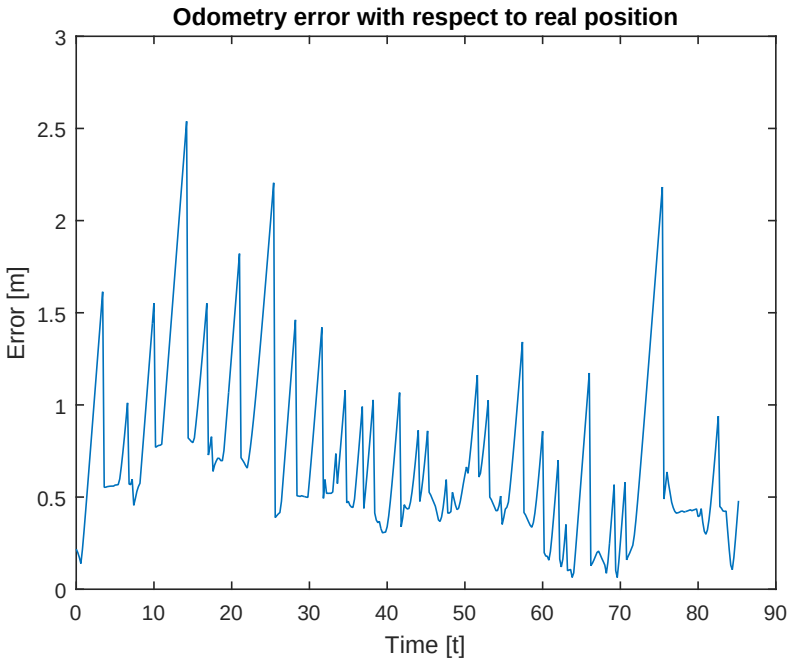


Figure 9-4: Error plot of odometry algorithm with respect ot actual track.

Chapter 10

Room Exploration

Phase 2 of the exploration algorithm is the most simple part. When exploration phase 1 is finished, a polygon based on the room corners can be constructed. Then the centre of this polygon is calculated and the a route is planned for the DelFly Explorer to reach that position and to make a full turn. This turn will provide a clear overview of most of the rooms. During this full turn exploration phase 3 is activated to already check for possible door candidates. Sometimes however, it is not possible to make a turn in the centre. Because the room is oddly shaped or there is an obstacle near/at the centre. Then, after 30 seconds, the algorithm will move on to the next phase.

In Figure 10-1 a track is presented where the rotation is successful and clearly visible. During the rotation the door opening is spotted as possible candidate. After the rotation the DelFly Explorer heads for the door to attempt a manoeuvre.

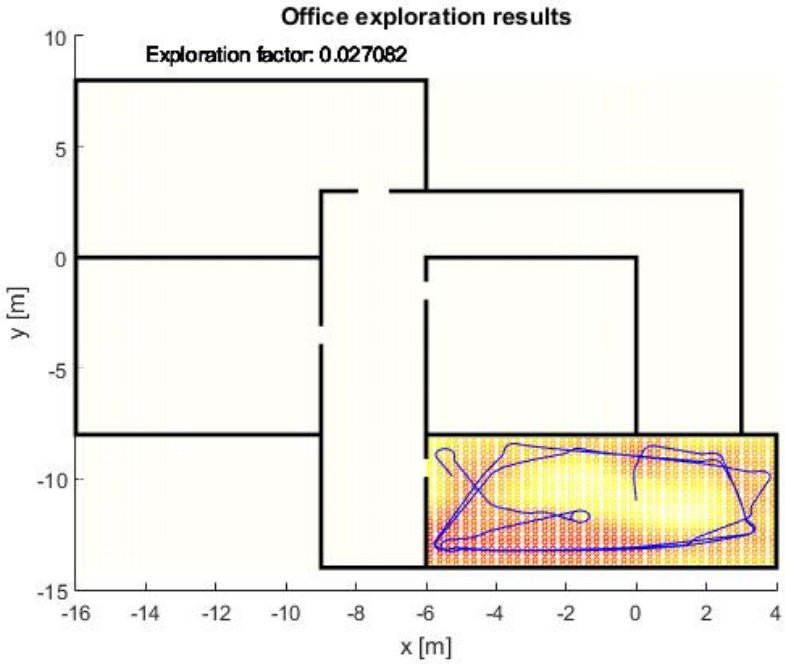


Figure 10-1: Flown track of wall-following algorithm (phase 1) followed by the center rotation (phase 2).

Chapter 11

Height Calculations

A rather stand-alone subject is the altitude of the DelFly Explorer. The main control loops in this simulation focus on the horizontal plane. In a large part of the simulations the flying altitude of the DelFly Explorer was fixed. However in real world scenario's it is not possible to fix the height in such a case. Furthermore one would want to prevent the DelFly Explorer hitting the floor or the roof and keep a relative constant altitude to easy the process of matching visual cues. In Section 2-1-1 it was explained that one of the sensors on-board is a barometer. A barometer is a useful device as it senses the air-pressure and when this air-pressure is changing it correlates to a change in altitude. To determine which altitude exactly, a table is necessary which matches a pressure with an altitude. However the corresponding air-pressure for a certain altitude can even differ per room in the environment. Therefore it is useful to develop a method which could calibrate the barometer for each room.

In Chapter 7 in short the working principle behind obtaining a disparity map was explained. From this disparity map a 3D-coordinate with respect to the left-camera for each detected pixel can be calculated. These coordinates can be calculated via Equation 11-1 to 11-3 [Hartley and Zisserman, 2003].

$$Z` = \frac{d \cdot f}{\Delta} \tag{11-1}$$

$$X` = (i - \frac{w}{2}) \cdot \frac{Z`}{f} \tag{11-2}$$

$$Y` = (j - \frac{h}{2}) \cdot \frac{Z`}{f} \tag{11-3}$$

First the Z' coordinate (the depth) of the pixel can be computed based on the pixel's disparity (Δ) and properties of the stereo-vision camera such as the distance between the two cameras (d) and the focal length of the lens (f). With this known Z' coordinate the $X`$ coordinate (horizontal position) and $Y`$ coordinate (vertical position) can be computed. To do this also the relative position of the pixel with respect to the centre of the image needs to be computed. In the case of the horizontal position this can be done by substracting half of the image width in pixels ($\frac{w}{2}$) from the horizontal position of that pixel on the image (i). This factor is then multiplied by the depth of the pixel over the focal-length of the lens. The same method can be applied for the vertical position. The final output are then the Cartesian coordinates (Z', X', Y') with respect to the left lens of the camera for that pixel.

The lowest value obtained for the vertical output of the different pixels corresponds to the height of the floor with respect to the DelFly Explorer (when the floor is in visual range). This value can then be used to calibrate the barometer.

As one can directly imagine this brings some limitations to the system. When closing into a wall the floor is in most cases no longer within the visual range of the DelFly Explorer therefore a much lower estimate of the floor height would be provided because it is based on the lowest visible part of the wall. This effect is visible in Figure 11-1.

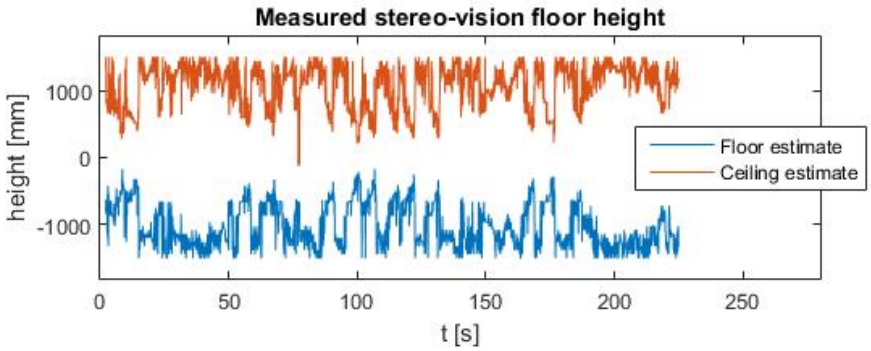


Figure 11-1: The measured floor/ceiling altitude by the stereo-height calculations algorithm during simulation.

As clearly is visible in Figure 11-1 is that the estimated floor/ceiling height based on the disparity map differs a lot during the course of the simulation. This is, as said earlier, due to the fact that the actual floor is not within the visual range and therefore the height is based on the lowest point on the wall. Furthermore one can see that the lowest value ($-1500mm$ with respect to the DelFly Explorer) is the most dominant value for the floor and is never exceeded in negative direction, for the ceiling ($1500mm$ with respect to the DelFly Explorer). This is the actual height of the floor (as in this simulation the altitude of the DelFly Explorer was fixed on $1500mm$). The route flown during this simulation is shown in Figure 11-2.

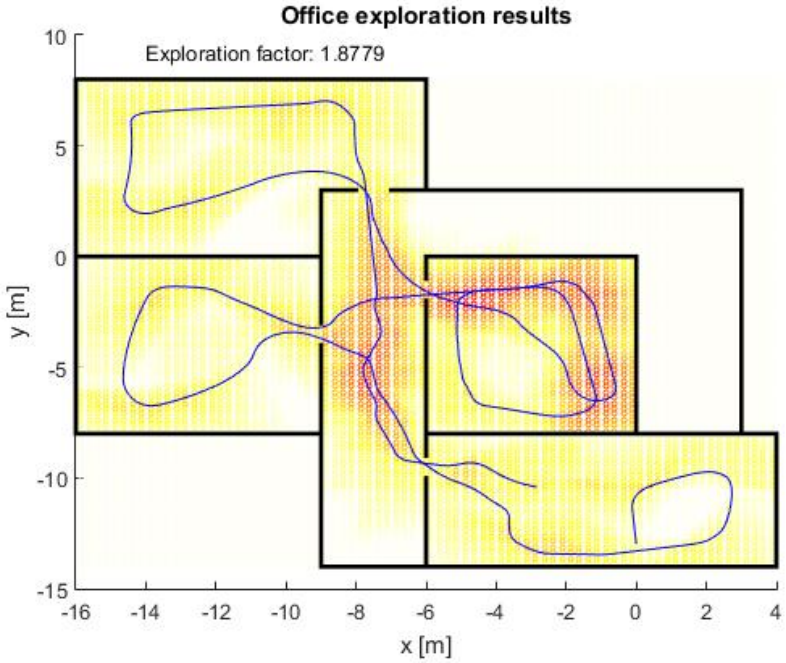


Figure 11-2: Route flown during the stereo-height simulation.

However using this varying altitude output to constantly calibrate the barometer is due to all the noise not very useful and will results in erratic flight behaviour. Therefore an addition to this algorithm would be necessary. The idea is to take the lowest dominant value within the last measurement period to calibrate the barometer. The result of this addition can be seen in Figure 11-3. This figure gives a very constant result, namely the actual height of the floor. Several peaks are visible. Analysis showed that these peaks correspond to the moments the floor and ceiling are no longer visible, thus the DelFly Explorer is approaching a wall. When the height difference between the floor and ceiling is below 1.5 m, the measurements are rejected, as shown in the figure.

The input for the height calculation script is as follows:

- Disparity map

The output of the height calculation script is as follows:

- Floor height
- Ceiling height

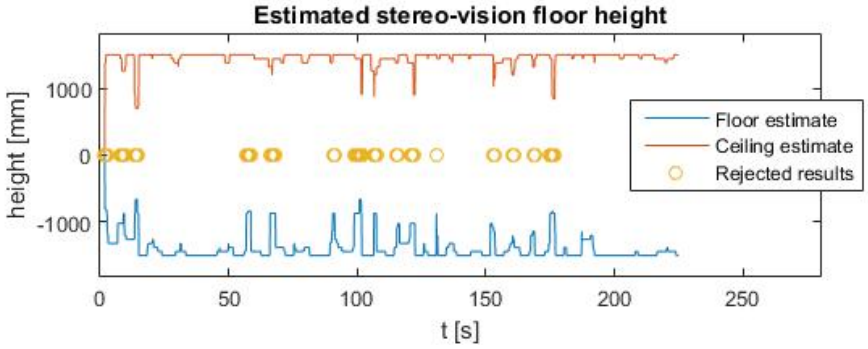


Figure 11-3: The estimated floor/ceiling altitude estimated over a period by the height calculations algorithm during simulation. Rejected measurements due to a detected wall are also shown.

Chapter 12

Manoeuvring

When the room exploration phase is finished the next exploration phase will aim at finding a new room or corridor. The manoeuvring algorithm is responsible for this task. The manoeuvring algorithm will aid in detecting possible door locations and planning the manoeuvre trough the door. Firstly, how the door will be identified will be explained in Section 12-1. Secondly, the actual tracking of the door and manoeuvring through the door will be explained in Section 12-2.

12-1 Door Detector

The door detector algorithm is based on the disparity map. A number of histograms is formed and based on the position on the disparity map a certain disparity is placed in a certain bin of the histogram. Another way this can be visualized is that the disparity map is split into n different vertical segments. For each of those segments (bins) the average disparity value and total number of detected disparities is calculated. The bin with the lowest disparity (corresponding with textures the furthest away) that is under the minimum disparity threshold which also has a total number of disparities above a certain threshold (this to reduce the effect of false positives and noise) will be selected as target bin. One of such bins that was selected as target bin can be seen in Figure 12-1. For the sake of visualisation the disparity grey values were inverted, normally nearby objects would be white and objects further away light grey.

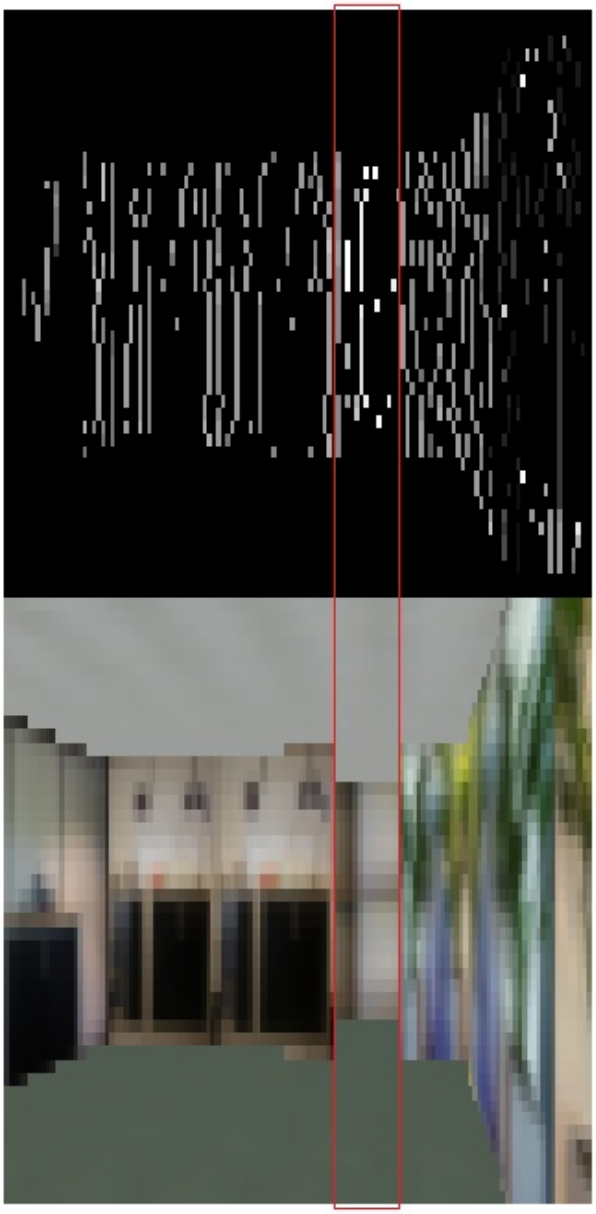


Figure 12-1: Door detector vertical target bin for 10 histograms, above the disparity map, below the simulator view.

As can be seen in Figure 12-1 the target bin contains a number of pixels with a very low disparity, bright coloured. These pixels correspond to the opening in the wall (the door) which can be seen in the bottom part of the figure. From this bin, which is now selected as target bin, the relative 3D coordinates are computed. This can be done by means of Equations 11-1 to 11-3 as discussed in Chapter 11.

12-2 Door Tracker

The door detector provides, as discussed in the section above, the relative 3D coordinates of a potential door. The door tracker then calculates the needed s-turn that has to be made such that the target location is straight ahead and with the same heading. The last part proved to be the most beneficial as the DelFly Explorer will most times fly parallel to a wall and with most room this means straight for a perpendicular wall. The s-turns are not directly initiated as the turn should be in the visual area of the droplet, as explained in Chapter 7. When the manoeuvre is finished the door detector algorithm will again analyse the incoming images and determine a new target. Resulting in a new s-turn and coming closer to the lateral position of the door while the distance of the door decreases. The whole process can be seen in Figures 12-2 to 12-4, for respectively 10, 20, and 30 bins in the histogram.

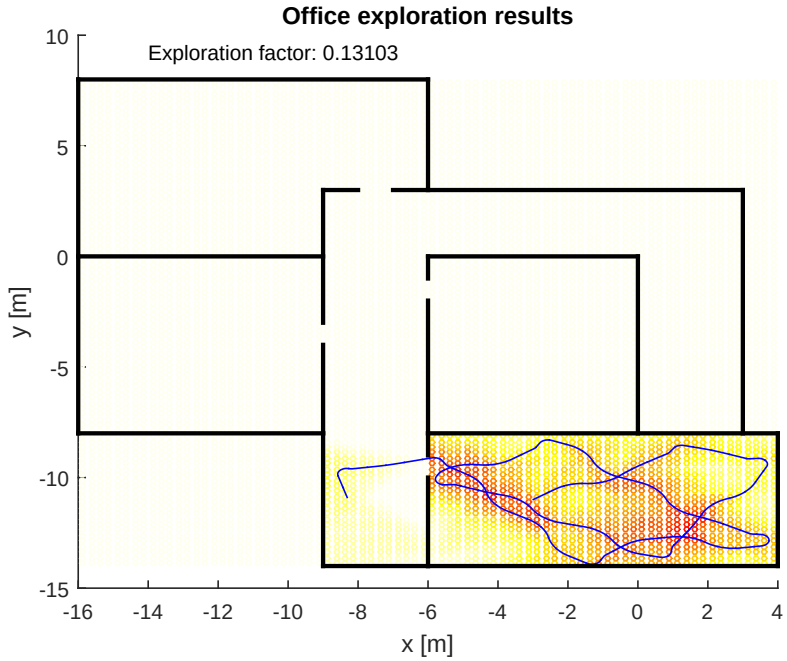


Figure 12-2: Representation of the flown route for the door tracker with 10 histograms.

Simultaneously with the manoeuvring algorithm also the the droplet is still active. This means that whenever the droplet is activated (due to an obstacle) the DelFly Explorer will turn accordingly. When the droplet turn is finished the door tracker will again determine a new target location, as the old one is probably no longer within the visual range of the DelFly Explorer.

When comparing the door detector and tracking algorithm with respect to the droplet the success rate of the algorithm can be determined. After performing 10 simulations per settins the results could be analysed. The absolute number of successful passages for each environment was counted. This is given in Table 12-1.

As one might observe from Table 12-1 is that in two of the three environment the droplet was not able to pass through the door opening at all. In environment 3 the droplet method only had half as many successful passages ad the manoeuvring algorithm. From this it can be concluded that the manoeuvring algorithm indeed has a large added value when exploring office environments.

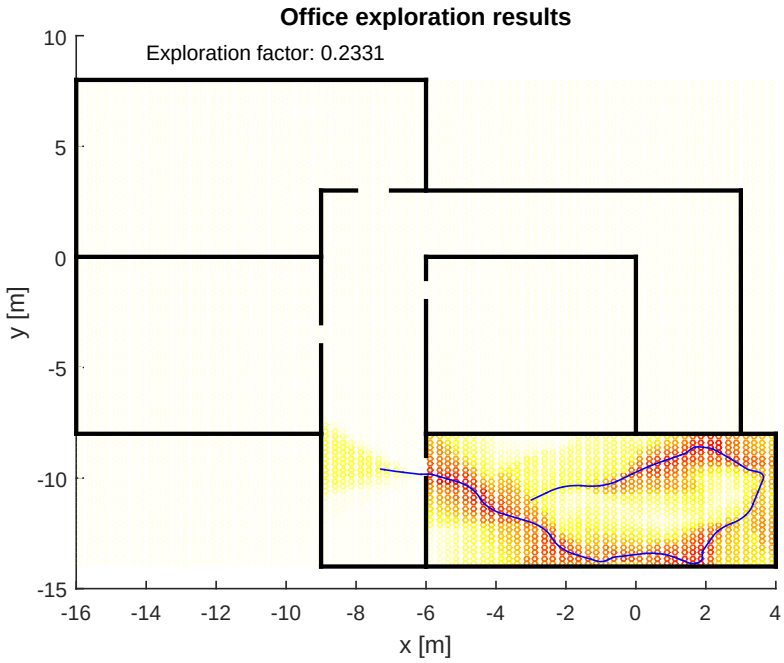


Figure 12-3: Representation of the flown route for the door tracker with 20 histograms.

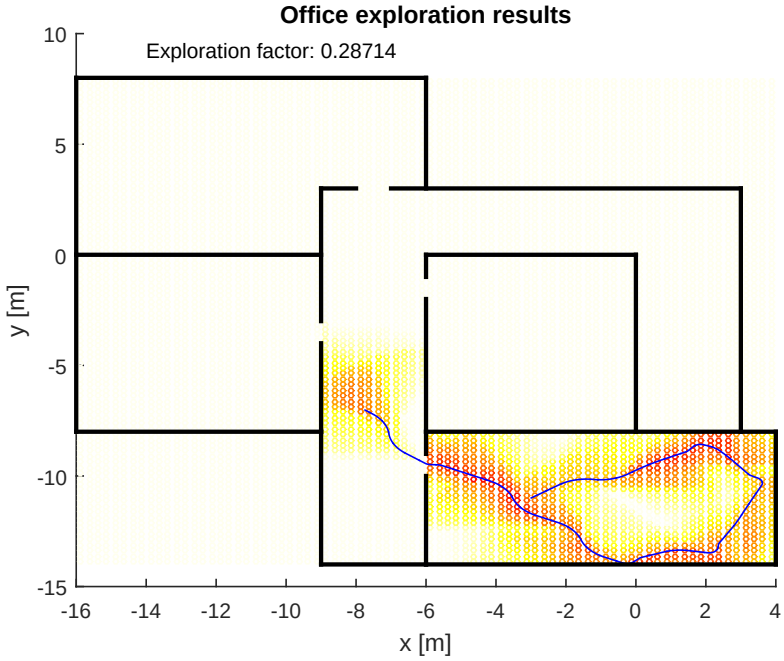


Figure 12-4: Representation of the flown route for the door tracker with 30 histograms.

Table 12-1: Number of successful passages of the droplet with respect to the manoeuvring algorithm for the three different environments

Environment #	Manoeuvring	Droplet
1	11	0
2	9	0
3	56	28

12-3 Field of View Analysis

120 simulations where performed to identify the effect of the camera's field of view of the ability to locate and manoeuvre trough the door. Field of view angles of 40, 50, 60, 70, 80, and 90 degrees were tested. To be able to perform these test the droplet properties and simulation settings should be adjusted for each field of view angle. In Figure 12-5 the results of these simulations are presented. For each of the settings the total number of successful door manoeuvres was counted. The results were then normalized with respect to the best performing setting. A clear optimum is present at a field of view of 70 degrees. From that point there is a declining performance when increasing or decreasing the field of view. Notably one can observe there is only a small performance difference between a field of view of 50 and 60 degrees. A possible explanation for these results is that a larger field of view has a larger view of the room and therefore a dominant false positive will be dominant in more situations. However, a too small field of view will limit the cases in which the room exit is in view and decreases the ability to make an approach with a high success probability. Also the optimal number of bins used for identification of the door might differ per field of view setting, this could be further investigated.

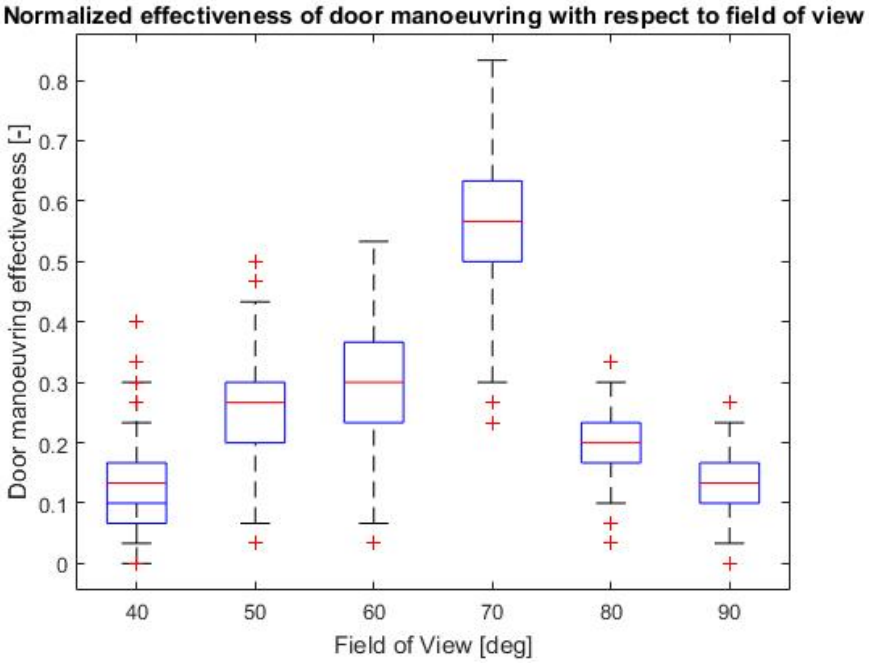


Figure 12-5: Normalized mean bootstrapped simulated door manoeuvring effectiveness with respect to the field of view of the stereo-vision camera.

As the results for a field of view of 70 degrees was most outstanding, p-values with respect to the 60 and 80 degrees where computed. The bootstrap method is used, because of the non-parametric nature of the data and the relative limited population size.[Mooney and Duval, 1993] For both yields p-value < 0.01.

Chapter 13

Overall Behaviour

Overall behaviour is needed to connect the different modules and see the result of the sum of the parts. For each of the 3 environments both multiple droplet and exploration algorithm simulations were performed. These were analysed based on their exploration factor to determine how well the algorithm performed. The results are represented in Figure 13-1.

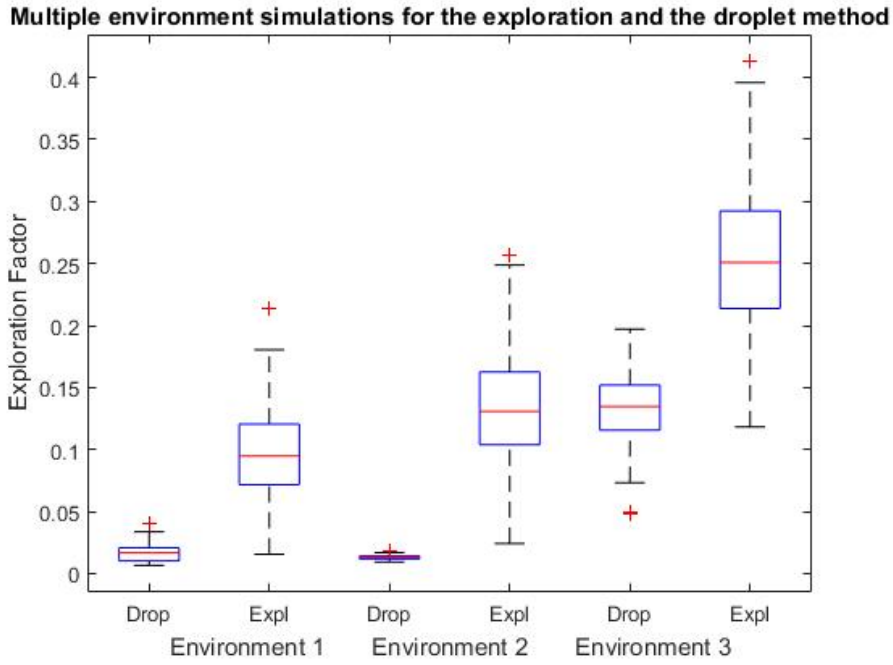


Figure 13-1: Mean bootstrapped average exploration factor, for over 60 simulations, of the droplet-only and exploration method. Performed in three different simulation environments.

From this figure it is clear that over all three environments the performance of the exploration algorithm exceeds the performance of the droplet only method. Furthermore, one can notice that the deviation in obtained results for the exploration algorithm is higher. This might be explained due to the fact that, for example door manoeuvres, a small distance can be the difference between failure or success in a certain approach. Then the approach should be done all over again and thus will result in a lower performance. From Figure 13-1 it can also be seen that some environments, such as environment 3, have a good layout for a random exploration approach such as the droplet. Because a high exploration factor is easily achieved. For all three environments the observed difference between the methods had a p-value < 0.01. This p-value was computed by means of the bootstrap method.[Mooney and Duval, 1993]

A very successful run is presented in Figure 13-2. Here it is very clear that the wall-following algorithm is first active when entering a new room and the layout is thoroughly followed. After that very successful door manoeuvres are performed with high success-rates.

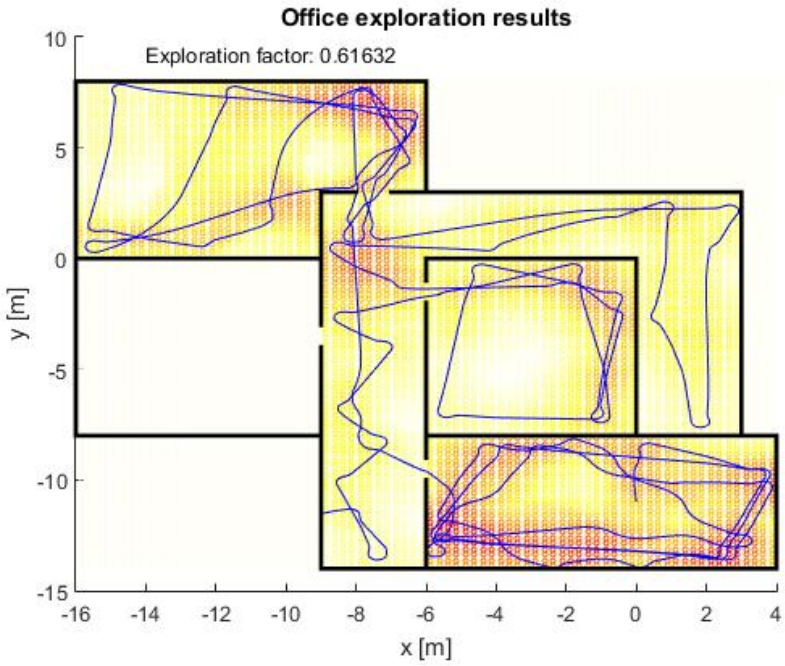


Figure 13-2: DelFly Explorer 600s simulated flight track for the exploration algorithm.

A failure case run is presented in Figure 13-3. Here the door manoeuvring algorithm is constantly unable to manoeuvre trough the door. Therefore the entire simulation time the DelFly Explorer is stuck in the same room.

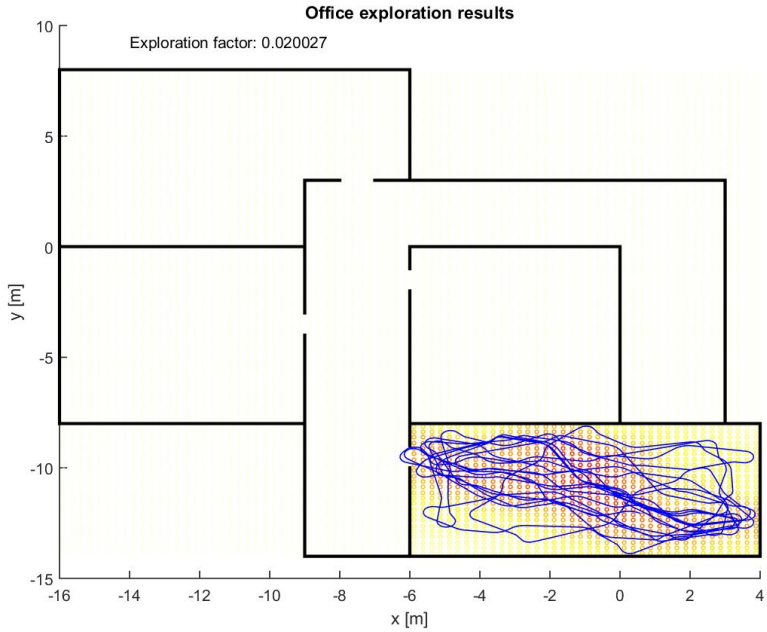


Figure 13-3: Unsuccessful DelFly Explorer simulation flight track for the exploration algorithm.

13-1 Overall Behaviour Conclusion

When looking at the performance figures for the full exploration method, mainly denoted by Figure 13-1. It can be concluded that the exploration method delivers for varying environments a big increase in the exploration factor. However some individual runs encountered problems, these problem mainly occurred for the center-rotate and door manoeuvring phases. The center-rotation is often not fully performed because the method used to construct the polygon might results in different weights for corners. Further more the door manoeuvring algorithm could be further optimized to increase its success-ratio and thus decrease time spend in an already explored room.

Part IV

Experiments

Chapter 14

Stereo-Height

As discussed in Chapter 11 the stereo-vision camera of the DelFly can be used to construct a disparity map. Based on the disparity map and the position of texture on the camera image 3D-coordinates for a certain texture can be determined. Based on these textures the stereo-height algorithm will determine the distance of the DelFly Explorer from the floor and ceiling. This information can be used to estimate the flying altitude, and whether or not a wall is covering the entire image.

14-1 Static Test

The first test type consist of static tests. For these tests the DelFly is manually moved through a series of pre-determined and measured heights. The resultant distance with respect to the floor and ceiling are shown on the data graph, see Figure 14-1. There is a clear view of the room with no direct obstructions of walls. The estimated height computed by the DelFly Explorer is communicated back to the ground station.

The first experiment concerned the algorithm which uses the stereo-vision camera for determining the flying height of the DelFly Explorer. First tests showed that the exact algorithm of the simulation resulted in a large amount of noise and unreliable outcomes. A solution in which for both the lower and upper part of the image the estimated height of all textures was sorted, was tested. This could then be used to determine the actual height based on the x% of highest/lowest textures. However sorting these values was a relative large computational demand for the obtained results, as a noisy frame is more apparent than a large amount of consecutive noisy frames. Therefore the method was changed to evaluate the estimated height based on the last 40 computations. As the algorithm was running at 40Hz this meant every second yields a totally independent measurement. Note, the stereo camera runs only on 24 Hz, so an image might be used two times for analysis. The test setup consisted of a number of pre-measured heights over which the DelFly Explorer was varied while being moved through the environment. Both the estimated floor- and ceiling-height were registered over time. One of these test is showed in Figure 14-1.

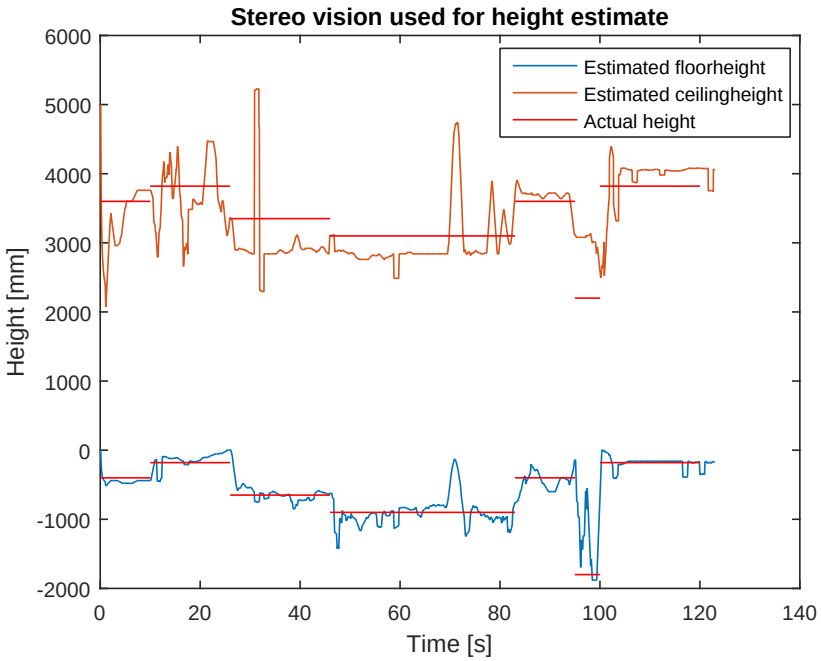


Figure 14-1: Graph showing estimated and true altitude of ceiling and floor with respect to the DelFly Explorer.

From this figure it is clear that the estimations are, especially for the floor height, quite accurate for the majority of the time. The absolute average error obtained is 116 mm for the floor, and 361 mm for the ceiling height.

A recommendation to improve this method is to look in more detail on how to also filter within an image. The current method takes the lowest and highest texture of each image. Which may be more prone to noise and to counteract that the current method takes into account many consecutive measurements. This makes the method quite interdependent and thus not fast. With less noise per frame this interdependency can be decreased.

14-2 Dynamic Test

Besides static tests, flying tests were also performed. These flying tests give a better insight in how, for example, vibrations affect the results. Also the results are obtained with the same conditions in which the algorithm would actually be deployed. The experiment is performed in the Cyberzoo, this is an fenced area of approximately 10m x 10m, in which the 3D location and attitude of the DelFly Explorer can be accurately tracked. Via this method the exact route of the DelFly Explorer can be saved. As the ceiling of the Cyberzoo is very high and thus not within the visual range of the camera, the algorithm will in the dynamic test only consider the floor height. The heights generated by the stereo-height algorithm can than be compared to the actual flying height of the DelFly Explorer and whether an obstacle was in view. A typical result is given in Figure 14-2.

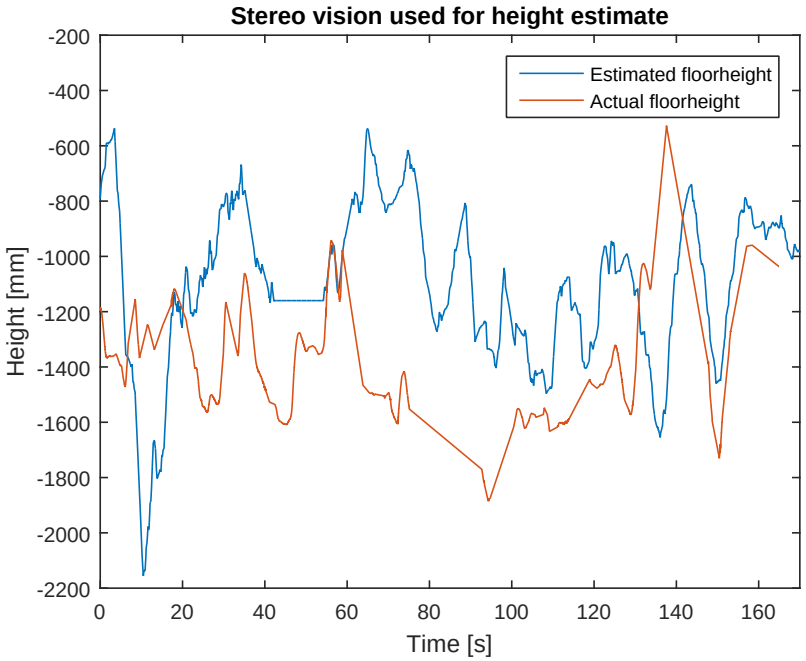


Figure 14-2: Graph showing estimated and true altitude of the floor with respect to the DelFly Explorer.

The obtained average root mean squared error over the performed experiments is 532mm. This is a factor 4.6 higher than during static tests. Likely this is cause by the extra vibrations and pitch angle changes while flying. Further research could investigate design adjustments or algorithm changes to decrease these effects and thus increase height reliability.

Chapter 15

Wall Following

The wall-following test is, just like the last stereo height experiment, performed in the Cyberzoo. This is an fenced area of approximately 10m x 10m, in which the 3D location and attitude of the DelFly Explorer can be accurately tracked via the optitrack system. Via this method the exact route of the DelFly Explorer can be saved. Here the ability of the algorithm to follow the walls and obstacles present in the area is evaluated.

Also the droplet only method is tested to be able to visualize the difference and determine the performance increase of the wall-following method. For both methods the paths were recorded, two 2D representations of these paths are visible in Figure 15-2.

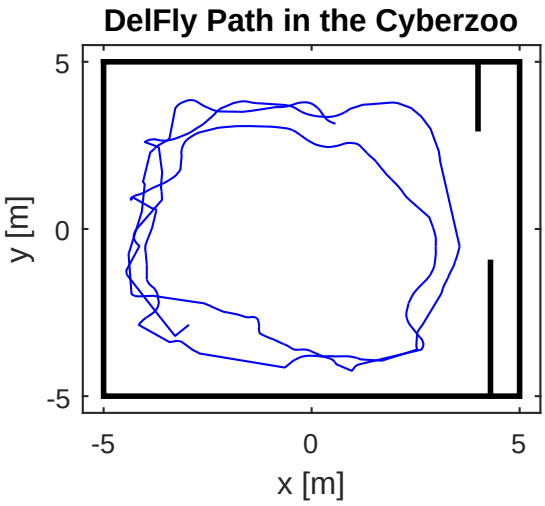


Figure 15-1: Experimental path results for the droplet only algorithm.

As visible in Figures 15-2 and 15-1 the wall-following algorithm is better capable of following the contours of the cyberzoo. To ensure this, a more active steering behaviour is visible. Furthermore the paths for each circulation are more similar. This part is of importance for the odometry algorithm, which depends on revisiting the locations where corner turns are made.

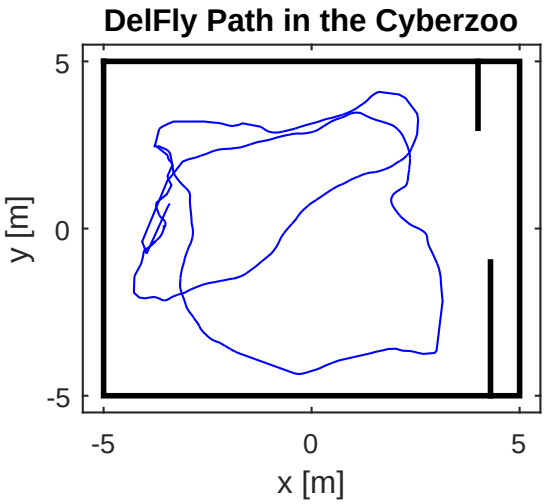


Figure 15-2: Experimental path results for the wall-following algorithm.

Table 15-1: Average distance from flying path with respect to the wall for wall-following and droplet only

Run #	Wall-Following	Droplet
1	2.17	2.23
2	1.77	2.78
3	1.94	2.49

The performance is determined by the average distance from the wall. For the wall-following algorithm this is 1.96m on average and for the droplet it is 2.50m, obtained over 3 tests form each. These results are presented in Table 15-1. From this it can be concluded that the wall-following script works successfully.

Chapter 16

Room Detection

When wall-following is correctly implemented the route can form the basis for detecting the the contour of the room. By means of this contour a room size estimation can be created. Furthermore reoccurring locations can help to identify when a room has been explored. As explained in Section 9 for this method a dead-reckoning system and magnetic field properties are used. In the experiment the DelFly Explorer was flown alongside the boundaries of the Cyberzoo for three rounds. The first round consist primarily of identifying points, based on turning, used for matching consecutive points. During the experiment the pilot aimed to keep speed and pitch constant, as these parameters are assumed to be constant by the algorithm. During flight corners could be matched based on the odometry position estimate and/or the magnetometer orientation. These matches were then saved. The actual position of the DelFly Explorer, that was recorded via optitrack, was then matched to these points. In Figure 16-1 the results are visualized. Only the green points, there were both the odometer and the magnetometer had a match, would have made it through both matching filters.

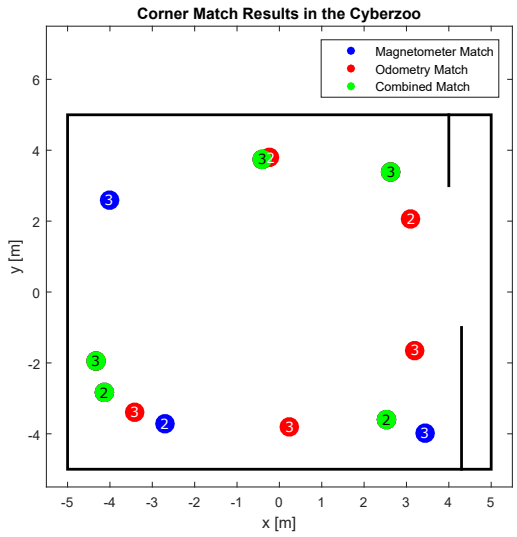


Figure 16-1: Results for correct odometry, magnetometer, and combined match locations during flight in the Cyberzoo. Numbers indicate in which round the points were generated.

Analysis of this test showed that in 62.5% the magnetometer matches also had a odometer match. For odometry this percentage was 50% with respect to the magnetometer. Looking at the locations of these matches is that they are all located near the boundaries of the Cyberzoo, which also corresponds to the flown route. Furthermore no clear unbalance in matched locations is visible from the tests.

Part V

Conclusion

Chapter 17

Conclusion

The goal of this project was to aid the development of technology in which the DelFly Explorer could be used for autonomous exploration in office environments. This technology will be usable in a wide range of applications, from search and rescue up to entertainment. The goal was formulated in the research question 'How to increase the indoor explored area of the DelFly Explorer by means of computationally efficient routing decisions?'. The accompanying sub-questions where dealt with and answered throughout the process of this thesis.

From related research a couple of restrictions based on the platform where found. The stereo-board has a used resolution of 128x96 pixels, a 168MHz processor, and 192kB RAM. This lead to the fact that indeed a computationally efficient method would be necessary. Furthermore literature provided insights about different methods for the range of tasks the algorithm should be able to perform. These tasks later translated themselves in the different modules used for simulation and experimentation (4). For example, from the BUG-algorithm wall-following proved to be a very useful property for this research.

As first step the simulation process was structured in different modules (1). In simulation all modules were developed, tested, and the new behaviour compared to the old (2). Finally also their combined behaviour was analysed. The droplet based wall-following method proved to be an adequate method to follow the contours of the room in simulation. Also the door-manoeuvring algorithm increased the ability to pass through doors with a factor 2.7 with respect to the old method (5). Room odometry was able to identify when a room was explored and a new area was entered (3). These methods performed well in simulation and resulted in a large increase in the exploration factor for indoor environments up to a factor 10 with respect to the old droplet-only method. From these results it can be concluded that the indoor exploration capabilities indeed increased with this new method.

The static stereo-height experiment gave very promising results in the static tests. The average root mean squared error over the performed experiments is 116 mm for the floor, and 361 mm for the ceiling height. Performance decreased in the dynamic test with an average root mean squared error over the performed experiments is 532 mm. At this moment the algorithm could only be used with a moving average filter to calibrate the barometer after certain observation times, but not instantaneously as preferred and still with a relative large uncertainty. The wall-following algorithm performed really well in the tests performed. With decreasing the average distance to the wall from 2.50 m obtained with the droplet to 1.96m. This at the cost of more controller inputs leading to a less smooth flown route and thus a decrease in effective range. Finally the room detection experiment showed the algorithm is in real flight experiments able to identify when corners where visited multiple times by means of an odometer and magnetometer.

Chapter 18

Recommendations

Concerning the software development, improvements could be made regarding the door-manoeuvring algorithm. When analysing all the simulation runs, this is the part that often took the longest time. A possible addition could be to loiter in front of a possible door candidate to do several attempts before selecting a new candidate. Also solutions in which the flight speed will be adjusted to decrease the droplet size could be investigated to potentially increase the success-rate of the algorithm.

Furthermore, the algorithm could be made more be robust for real flight applications, for example with live speed estimation instead of assumed speed. This will make the odometry used for room detection more reliable. The corners that are detected are used to determine the target for the centre rotation, however this location is often close to a wall and therefore not executed. By improving this estimation the usefulness of this centre rotation can be enhanced.

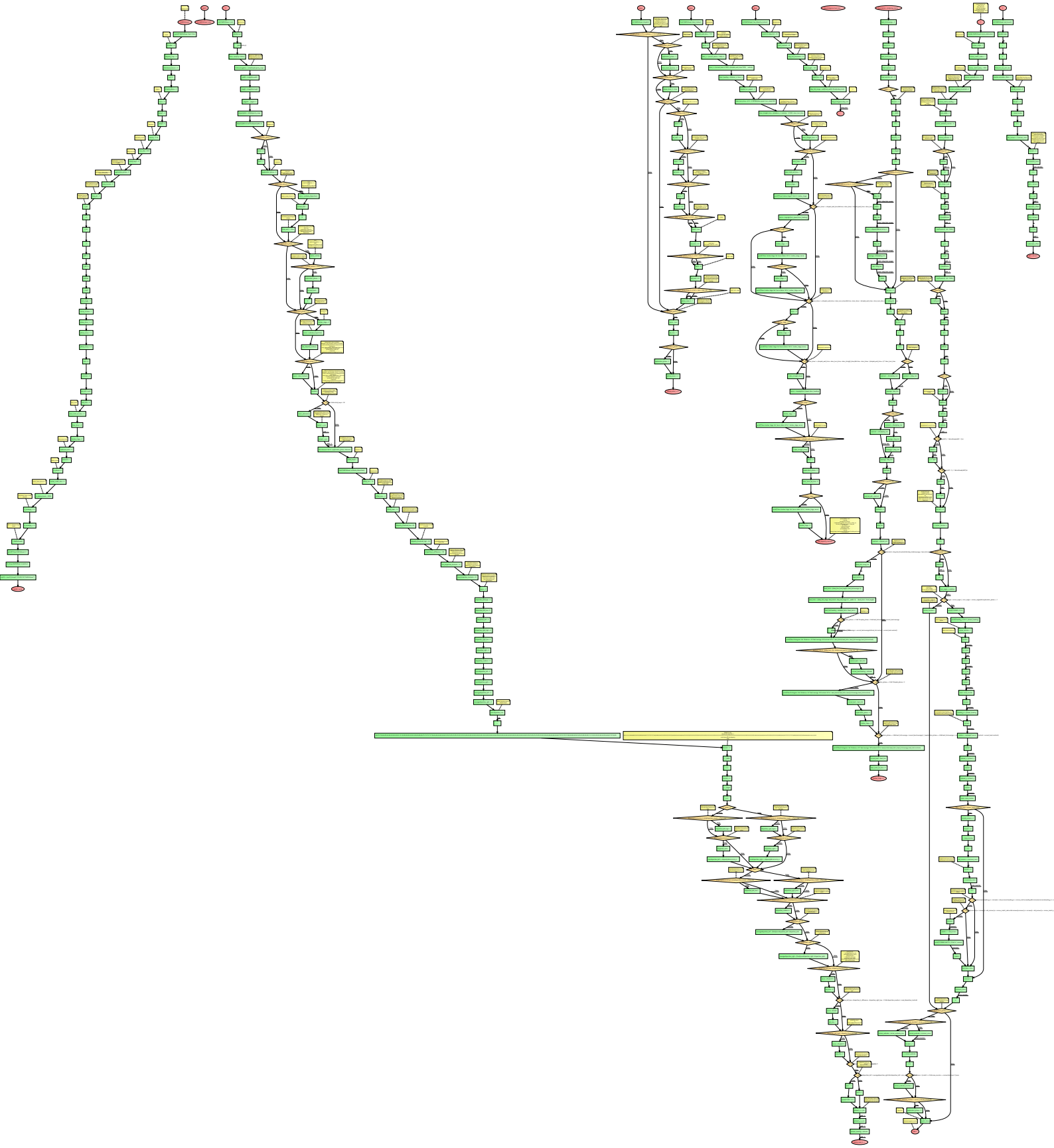
In real flight tests this room detection algorithm should be tested in combination with wall-following. This to review performance of the room detection algorithm when combined with autonomous flight. Also the effectiveness of the wall-following algorithm with respect to the droplet-only method can be further evaluated in non-square rooms, simulations predict that in these settings it will outperform the droplet-only even more. Finally the door manoeuvring should be added to the real flight experiments to test the algorithm in real flight as a whole.

Appendices

Appendix A

Appendices

Flowchart representation of the exploration algorithm. Only readable in pdf-viewer.



Bibliography

A. Angeli, S. Doncieux, J. A. Meyer, and D. Filliat. Incremental vision-based topological SLAM. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1031–1036, Nice, 2008. ISBN 9781424420582. doi: 10.1109/IROS.2008.4650675.

B. Baddeley, P. Graham, A. Philippides, and P. Husbands. Holistic visual encoding of ant-like routes: Navigation without waypoints. *Adaptive Behavior*, 19(1):3–15, 2011. ISSN 1059-7123. doi: 10.1177/1059712310395410.

B. Baddeley, P. Graham, P. Husbands, and A. Philippides. A Model of Ant Route Navigation Driven by Scene Familiarity. *PLoS Computational Biology*, 8(1), 2012. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002336.

S. S. Baek, F. L. Garcia Bermudez, and R. S. Fearing. Flight Control for Target Seeking by 13 gram Ornithopter. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2674–2681, 2011. ISBN 9781612844541. doi: 10.1109/IROS.2011.6048246.

C. Bills, J. Chen, and A. Saxena. Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues. In *IEEE International Conference on Robotics and Automation*, pages 5776–5783, Shanghai, 2011. ISBN 9781612843865. doi: 10.1109/ICRA.2011.5980136.

S. Birchfield and C. Tomasi. Depth Discontinuities by Pixel-to-Pixel Stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999. ISSN 09205691. doi: 10.1023/A:1008160311296.

J. L. Blanco, J. A. Fernandez-Madrigal, and J. Gonzalez. Toward a Unified Bayesian Approach to Hybrid Metric–Topological SLAM. *IEEE Transactions on Robotics*, 24(2):259–270, 2008. ISSN 1552-3098. doi: 10.1109/TRO.2008.918049.

C. L. Bottasso, D. Leonello, and B. Savini. Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives. *IEEE Transactions on Control Systems Technology*, 16(6):1152–1168, 2008. ISSN 1063-6536. doi: 10.1109/TCST.2008.917870.

B. J. Cohen, S. Chitta, and M. Likhachev. Search-based Planning for Manipulation with Motion Primitives. In *IEEE International Conference on Robotics and Automation*, pages 2902–2908, Anchorage, 2010. ISBN 9781424450404.

A. J. Davison and D. W. Murray. Simultaneous Localization and Map-Building Using Active Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865–880, 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017615.

G. C. H. E. de Croon, K. M. E. de Clercq, R. Ruijsink, B. Remes, and C. de Wagter. Design, aerodynamics, and vision-based control of the DelFly. *International Journal of Micro Air Vehicles*, 1(2):71–98, 2009.

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, 2012a. ISSN 09218890. doi: 10.1016/j.robot.2011.10.001.

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, 2012b. ISSN 15523098. doi: 10.1109/TRO.2011.2170754.

G. C. H. E. de Croon, L. M. O'Connor, C. Nicol, and D. Izzo. Evolutionary robotics approach to odor source localization. *Neurocomputing*, 121:481–497, 2013. ISSN 09252312. doi: 10.1016/j.neucom.2013.05.028.

G. C. H. E. de Croon, M. Perçin, B. D. W. Remes, R. Ruijsink, and C. De Wagter. *The DelFly*. Springer, Dordrecht, 2016. ISBN 9789401792073.

E. de Margerie, J. B. Mouret, S. Doncieux, and J. A. Meyer. Artificial evolution of the morphology and kinematics in a flapping-wing mini-UAV. *Bioinspiration & Biomimetics*, 2(4): 65–82, 2007. ISSN 1748-3182. doi: 10.1088/1748-3182/2/4/002.

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 *IEEE International Conference on Robotics & Automation (ICRA)*, number Section II, pages 4982–4987, Hong Kong, 2014. ISBN 9781479936847. doi: 10.1109/ICRA.2014.6907589.

D. Ferguson, N. Kalra, and A. Stentz. Replanning with RRTs. In *IEEE International Conference on Robotics and Automation*, number May, pages 1243–1248, Orlando, 2006. ISBN 0780395069. doi: 10.1109/ROBOT.2006.1641879.

F. Fraundorfer, L. Heng, D. Honegger, G. Hee Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4557–4564, Vilamoura, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6385934.

M. W. M. Gamini-Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001. ISSN 1042296X. doi: 10.1109/70.938381.

K. Hajebi and J. S. Zelek. Sparse disparity map from uncalibrated infrared stereo images. *Third Canadian Conference on Computer and Robot Vision, CRV 2006*, 2006, 2006. doi: 10.1109/CRV.2006.68.

R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.

H. Hirschmüller. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):328–41, 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1166.

A. Howard. Real-Time Stereo Visual Odometry for Autonomous Ground Vehicles. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 3946–3952, Nice, 2008. ISBN 9781424420582. doi: 10.1109/IROS.2008.4651147.

S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined Optic-Flow and Stereo-Based Navigation of Urban Canyons for a UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 302–309, Edmonton, 2005. ISBN 0780389123. doi: 10.1109/IROS.2005.1544998.

L. A. Jeni, G. Flórea, and A. Lörincz. InfoMax Bayesian Learning of the Furuta Pendulum. *Acta Cybernetica*, 18:637–649, 2003.

M. Keennon, K. Klingebiel, H. Won, and A. Andriukov. Development of the Nano Hummingbird: A Tailles Flapping Wing Micro Air Vehicle. In *AIAA Aerospace Sciences Meeting*, number January, pages 1–24, Nashville, 2012. ISBN 978-1-60086-936-5. doi: 10.2514/6.2012-588.

A. Koschan, V. Rodehorst, and K. Spiller. Color Stereo Vision Using Hierarchical Block Matching and Active Color Illumination. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 835–839, Vienna, 1996. ISBN 0-8186-7282-X. doi: 10.1109/ICPR.1996.546141.

T. Li and Y. Shie. An Incremental Learning Approach to Motion Planning with Roadmap Management. In *IEEE International Conference on Robotics and Automation*, New Orleans, 2004.

H. C. Longuet-Higgins and K. Prazdny. The Interpretation of a Moving Retinal Image. *Royal Society of London*, 208:385–397, 1980.

B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, 1981. ISBN 0001-0782. doi: Doi 10.1145/358669.358692.

MAVlab. The DelFly project, 2015. URL `http://www.delfly.nl/home.html`.

C. Z. Robert Mooney and R. D. Duval. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Sage University Papers, quantitati edition, 1993. ISBN 978-0803953819.

D. Nister, O. Naroditsky, and J. Bergen. Visual Odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, Washington, 2004. ISBN 0-7695-2158-4. doi: 10.1109/CVPR.2004.1315094.

Y. Ohta and T. Kanade. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE transactions on pattern analysis and machine intelligence*, 7(2):139–154, 1985. ISSN 0162-8828. doi: 10.1109/TPAMI.1985.4767639.

M. Okutami and T. Kanade. A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment. *IEEE TRANSACTIONS ON PATTERN ANAYLSIS AND MACHINE INTELLIGENCE*, 16(9):920–932, 1994.

S. Rajko and S. M. LaValle. A Pursuit-Evasion BUG Algorithm. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1954–1960, Seoul, 2001. ISBN 0-7803-6576-3. doi: 10.1109/ROBOT.2001.932894.

K. Schauwecker and A. Zell. On-Board Dual-Stereo-Vision for the Navigation of an Autonomous MAV. *Journal of Intelligent & Robotic Systems*, 74:1–16, 2014. ISSN 0921-0296. doi: 10.1007/s10846-013-9907-6.

K. Y. W. Scheper, S. Tijmons, C. C. de Visser, and G. C. H. E. de Croon. Behaviour Trees for Evolutionary Robotics Behaviour Trees for Evolutionary Robotics. *Artificial Life*, 22(1): 23–48, 2015.

S. Shen, N. Michael, and V. Kumar. Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 20–25, Shanghai, 2011. ISBN 9781612843865. doi: 10.1109/ICRA.2011.5980357.

K. Taylor and S. M. LaValle. I-Bug: An Intensity-Based Bug Algorithm. In *IEEE International Conference on Robotics and Automation*, pages 3981–3986, Kobe, 2009. ISBN 1050-4729 U9 - We have successfully determined that a robot can, in theory, navigate to a signal source using its intensity and gradient, in spite of having no other information about its configuration, the obstacles, or even the intensity mapping (except that its level sets are topological circles). The robot uses a contact sensor, an intensity sensor, and an alignment sensor to achieve the task of reaching a goal, which is the signal source. The robot does not have access to perfect clocks, o. doi: 10.1109/ROBOT.2009.5152728.

S. Tijmons, G. C. H. E. de Croon, B. D. W. Remes, C. de Wagter, R. Ruijsink, E . van Kampen, and Q. P. Chu. Off-board processing of Stereo Vision images for Obstacle Avoidance on a Flapping Wing MAV. In *PEGASUS - AIAA Student Conference*, pages 1–16, Prague, 2014.

B. Tovar, S. M. LaValle, and R. Murrieta. Locally-optimal navigation in multiply-connected environments without geometric maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 4, pages 3491–3497, Las Vegas, 2003. ISBN 0780378601.

B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-Optimal Navigation in an Unknown Environment Without Sensing Distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007. ISSN 1552-3098. doi: 10.1109/TRO.2007.898962.

M. Varma and A. Zisserman. Texture Classification: Are Filter Banks Necessary? In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 691–698, Madison, 2003. ISBN 0-7695-1900-8. doi: 10.1109/CVPR.2003.1211534.

J. L. Verboom, S. Tijmons, C. de Wagter, B. Remes, R. Babuska, and G. C. H. E. de Croon. Attitude and Altitude Estimation and Control on board a Flapping Wing Micro Air Vehicle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5846–5851, Seattle, 2015. ISBN 9781479969234.

P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Kauai, 2001. ISBN 0-7695-1272-0. doi: 10.1109/CVPR.2001.990517.

K. Weber, S. Venkatesh, and M. V. Srinivasan. Insect Inspired Behaviours for the Autonomous Control of Mobile Robots. In *International Conference on Pattern Recognition*, number iii, pages 156–160, Vienna, 1996.

M. Zohaib, S. M. Pasha, N. Javaid, and J. Iqbal. Intelligent Bug Algorithm (IBA): A Novel Strategy to Navigate Mobile Robots Autonomously. In *Springer's International Multi Topic Conference*, Jamshoro, 2013. ISBN 978-3-319-10987-9.

J. C. Zufferey and D. Floreano. Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft. *IEEE Transactions on Robotics*, 22(1):137–146, 2006.

J. C. Zufferey, A. Klaptocz, A. Beyeler, J. D. Nicoud, and D. Floreano. A 10-gram vision-based flying robot. *Advanced Robotics*, 21(14):1671–1684, 2007. ISSN 01691864. doi: 10.1163/156855307782227417.