Czech technical university in Prague
Faculty of Electrical Engineering

Bachelor thesis

Daniel Brandtner

Coherent swarming of micro aerial vehicles with minimum computational and communication requirements

Department of Cybernetics

Thesis supervisor:	Ing. Martin Saska, Dr. rer. nat.
Study programme:	Cybernetics and Robotics
Specialisation:	Robotics

Prague 2016

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Daniel B r a n d t n e r
Study programme:	Cybernetics and Robotics
Specialisation:	Robotics
Title of Bachelor Project:	Coherent Swarming of Micro Aerial Vehicles with Minimum Computational and Communication Requirements

Guidelines:

An approach for stabilization and navigation of a group of Micro Aerial Vehicles (MAVs) with limited computational resources, onboard sensors and communication will be designed, implemented and experimentally verified.
Work plan:

- To design and implement an extension of the method in [1] for an MAV team flying in 3D space
- To integrate the algorithm into the simulator V-REP and into the system for control of MAVs of MRS group at Department of Cybernetics
- To verify the algorithm in the V-REP simulator, to analyze responses of the MAV swarm in narrow corridors, to detected dynamic obstacles, to sensory drop-out, etc. In case of availability of the HW platform, to verify the system with real MAVs (thesis advisor will decide whether the experiment or more detailed analyses in the simulator will be conducted).

Bibliography/Sources:
[1] C. Melhuis, J. Nembrini, A. Winfield: Minimalist coherent swarming of wireless networked autonomous mobile robots. In ICSAB: Proceedings of the seventh international conference on simulation of adaptive behavior on from animals to animats, 2002.
[2] H. Min, Z. Wang: Design and analysis of Group Escape Behavior for distributed autonomous mobile robots. IEEE International Conference on Robotics and Automation (ICRA), 2011.
[3] M. Saska, J. Vakula and L. Preucil: Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization. In IEEE International Conference on Robotics and Automation (ICRA), 2014.
[4] V. Trianni: Evolutionary Swarm Robotics. Springer, 2008.

Bachelor Project Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the winter semester of academic year 2016/2017

L.S.

doc. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
Head of Department	**Dean**

Prague, September 30, 2015

I would like to express my gratitude to my thesis supervisor Dr. Saska for his support, guidance and interest in my research and prof. Martinoli for introducing me to the field of swarm robotics.

Název práce: Koherentní rojové chování bezpilotních helikoptér s minimálními výpočetními a komunikačními nároky

Autor: Daniel Brandtner

Abstrakt: Tato práce navrhuje rozšíření Minimalistického koherentního rojového algorithmu vyvinutého J. Nembrinim a kolektiv [4] pro roj MAV (Micro Areal Vehicle) - bezpilotních helikoptér. Zreprodukovali jsme nejdůležitější body tohoto článku a základní princip jsme rozšířili, například o odlišný algorithmus pro sledování cíle nebo o pohyb ve 3D. Představený algoritmus umožňuje roji helikoptér zachovat soudržnost a provádět rovinný pohyb s velmi omezenou výpočetní a komunikační kapacitou. Tato metoda je škálovatelná, robustní a plně distribuovaná.

Klíčová slova: MAV, rojová robotika, koherentní rojové chování, bezpilotní roboti

Title: Coherent swarming of micro aerial vehicles with minimum computational and communication requirements

Author: Daniel Brandtner

Abstract: This thesis presents an extension of the Minimalist Coherent Swarming algoritm presented by Nembrini et al. [4] for using on MAVs (micro aerial vehicles). We reproduce the main ideas of this paper as well as introduce new ones, such a different target following algorithm and the expansion into 3D. The presented algoritm enables a swarm of MAVs to maintain coherence and perform planar motion with only very limited computational and communication requirements. The method is highly scalable, robust and fully distributed.

Keywords: MAV, swarm robotics, coherent swarming, autonomous robots

Contents

Chapter 1

Introduction

Swarm robotics attempts to implement a collective behaviour to a group of autonomous robots without explicit central control. This behaviour emerges from local interactions between the robots and their environment and between the robots themselves. This method enables the creation highly scalable, flexible and robust autonomous swarms. Swarm robotics is a application of swarm intelligence, a discipline dealing with natural and artificial decentralized, self-organized multi-agent systems [1]. It has been influenced by studies in biology, many swarming algorithms are implementations of behaviours observed on animal interactions. BOID models are influenced by bird flocking [5]. Nesting and foraging habits of various species of insects also serve as inspiration [6] as well as fish schooling [7, 3].

Micro aerial vehicles (MAVs) have undergone a big boom in the recent years, both among the scientific community and the broader public. An MAV is generally composed of a platform and multiple propellers (Figure 1.2). Studies have successfully implemented various swarming algorithm on MAV swarms [13, 8]. MAV swarms are well suited to tasks of aerial reconnaissance or surveillance, especially in difficult terrain. Examples of utilization include terrain mapping or assistance with the detection of survivor during disasters.

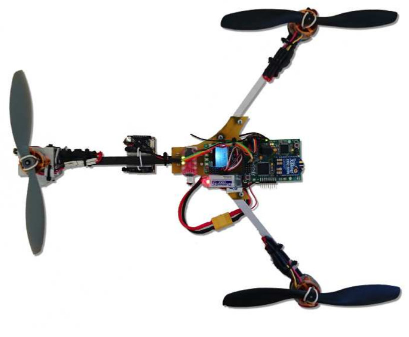

Figure 1.1: Example of an MAV (image taken from [2])

Julien Nembrini, Alan Winfield and Chris Melhuish introduced a minimalist, scalable and robust method of swarming for autonomous mobile ground-based robots [4] and in this work, we will modify and implement it for uses on MAVs. This chapter presents the basic principle of the swarm cohesion mechanism described in [4].

1.1 Basic Algorithm

The algorithm presented [4] is designed for a homogeneous swarm of ground-based robots. Every robot is equipped with a sensor enabling it to discern the presence adjacent robots (called "neighbors"). The crucial property of this sensor is a known and constant range of detection of other robots. The basic premise of the algorithm is fairly simple and is sketched on Figure 1.2.

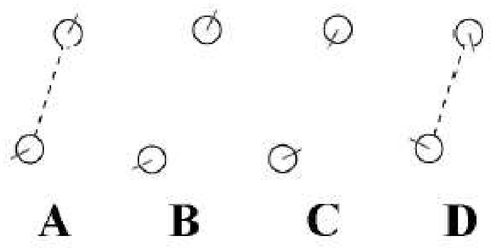

Figure 1.2: Basic Algorithm (image taken from [4])

We can consider three basic behaviours or states for each robot: Forward (default), Coherence and Avoidance. A robot in the Forward state (section **A** on Figure 1.2) maintains a constant velocity as long as nothing disturbs it. When connection with a neighbor is lost (**B**) the robot enters into the Coherence state and turns around (**C**) and goes back until it regains its connection. When it returns within range, it performs a random turn (**D**) and returns into the Forward state. The state of Avoidance is entered if two or more robots get too close and they have to perform an evasive manoeuvre.

1.2 Alpha Algorithm

The more advanced variant of the basic algorithm, which is used in this thesis, is the so called Alpha Algorithm. A constant *Alpha* is introduced indicating the minimal number of neighbors that have to be connected with the robot to remain in the Forward state, ignoring any additional neighbor loss. Meaning, the robot doesn't go into the Coherence state at least as long as it detects *Alpha* neighbors. This enables a wider and better structured swarm where, ideally, each robot has *Alpha* neighbors.

1.3 Advantages and application

This approach has several advantages:

- The system is fully distributed. Each robot performs its running on-board based on outputs of its own sensors. Only local interaction is enabled between pairs of identical robots. This makes the swarm highly scalable as the computation complexity increases linearly with the number of robots.
- The system is robust. In extreme environments, due to technical issues or when the system is poorly parametrised (in Chapter 3, an overview of parameters influencing stability is presented), individual robots or smaller sub-swarms might break apart from the main group. But loosing a member does not prevent the swarm from continuing its task.
- The knowledge of neither absolute or relative position is needed. Cohesion is maintained with the binary information of the presence of neighbors in the robot's proximity.

The designed communication network enables application in exploration, mapping and network sensing. The independence of the swarm from external control, computation and positioning references make it suitable for missions in uncharted, inaccessible locations. Its high scalability allows the deployment of large swarms (the number of individuals is theoretically unlimited).

Chapter 2

Improvement for MAVs

In this chapter, we will introduce the process of building our swarm coherence algorithm implemented specifically to MAV (Micro Air Vehicle) models.

2.1 V-REP

V-REP, or Virtual Robot Experimentation Platform, is a general purpose robot simulator with an integrated development environment by Coppelia Robotics. It offers a large scale of models of mobile and non-mobile robots with all sorts of sensors. A complex environment can be built with basic three dimensional shapes, light sources and cameras. Every object has a set of parameters, some of them are specific to a given object (such as the number of articulation of an industrial robot or the range of a sensor), some are general (weight, color, visibility). Most of the prepared models come with a pre-written script in Lua that handles several basic functions. Lua is a lightweight multi-paradigm programming language very similar to C. Its libraries offer a wide array of function designed for robotic uses. These controllers can be also written in C or C++, Python, Java, Matlab, Octave and Urbi. V-REP simulations are run with a time step of 50 ms. Graphs presented in this thesis use this time step as the basic unit on the X axis. This value can be changed, but one has to be careful since the models may not behave in the same way after that.

2.2 Introduction to the MAV

The MAV model provided by V-REP (shown on Figure 2.1) is composed of a base, four propellers and a virtual object named quadricoper_target, called a "set point". Most land-based mobile robots control motion by directly setting the wheel speeds. Motion of MAVs is handled very conveniently, we can set the position of the set point and a PID controller handles automatically the movement of the MAV towards this set point. A implementation of a similar principle on real MAVs has been performed in [2]. The movement algorithm can be divided into three parts, controlling vertical, horizontal and rotary motion. We will now give a quick overview of all of them by analysing the response of the MAV to changes in position or rotation of the set point.

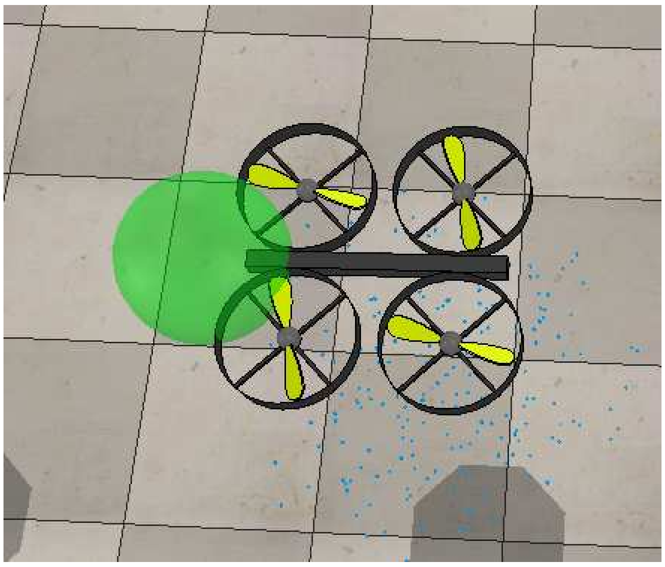
Figure 2.1: MAV model in V-REP, the set point is depicted in green

2.2.1 Vertical control

The MAV is most easily controlled in the vertical dimension. In this case, the algorithm is fairly straightforward. The thrust of all the propellers is adjusted uniformly according to the difference of altitude between the set point and the MAV. A series of step responses of the model has been prepared, where the set point height suddenly jumps by a given value (Figure 2.2). This movement is very smooth due to the fact that the robot does not have to tilt in order to change its velocity (as it is the case with the horizontal movement, as shown later). The time necessary to reach the required elevation is roughly 40 time steps (2 seconds) for step changes of the set point altitude to 1 m. We can also observe that the responses to positive and negative elevation changes are symmetrical, which is not obvious when dealing with ascending/descending movements.

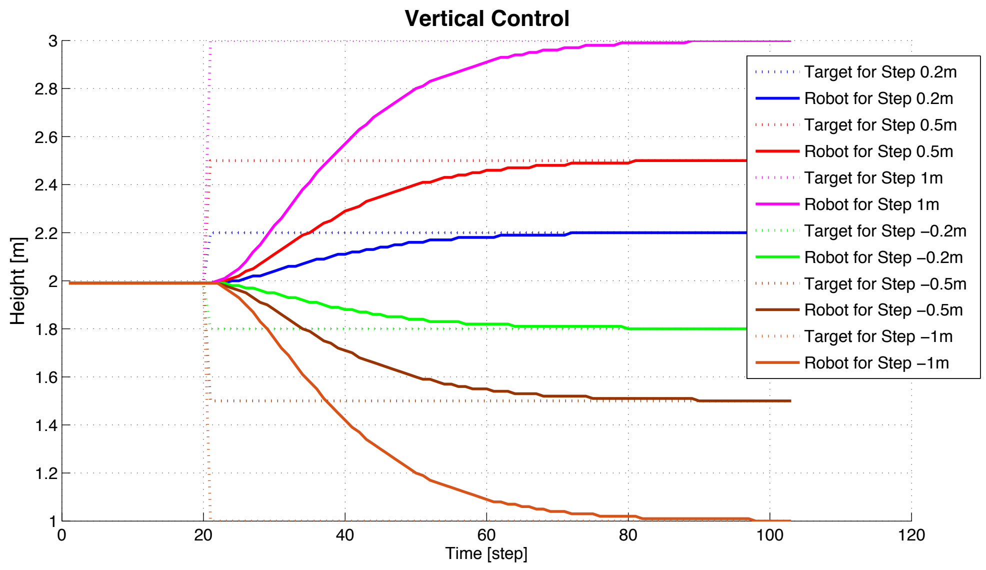

Figure 2.2: Vertical control

Another desirable quality of the tested controller is robustness. This is simulated by changing quickly the position of the set point and observe the way the controller handles it. The vertical control manages consecutive changes very well (Figure 2.3). There is only a small overshoot caused by inertia but the system has no problem with stabilization.

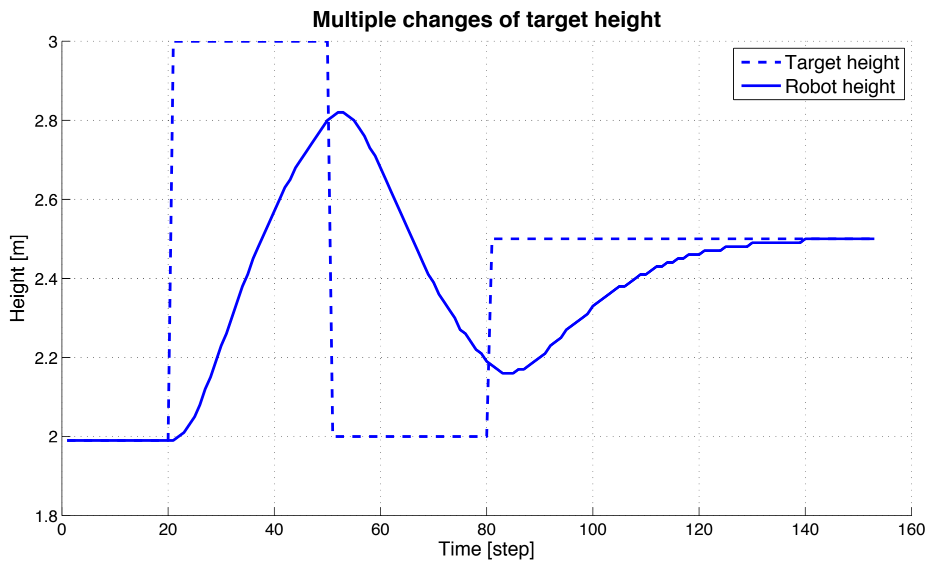

Figure 2.3: Vertical control - consecutive changes

An important specific aspect of MAVs is their mutual interference. Air flow generated by one robot affects its neighbors. This is especially true if they are placed one under the other. For this reason the swarm is forced to maintain a constant height, for now. The effect of this interference and the extension of the swarming algorithm into 3D is discussed in Chapter 5.

2.2.2 Horizontal control

The horizontal control is more complex. This is caused by the structure of the MAV. As was mentioned earlier, only one variable is required to control the elevation, but horizontal movement is obtained by balancing the thrusts of the different propellers. The craft tilts in the direction of the set point to accelerate and tilts in the opposite direction to break. The response to a sudden change of the position of the set point oscillates (Figure 2.4) and overshoots by approximately 20%. It takes about 60 time step (3 seconds) for the system to stabilize around its final position for set point position jumps smaller than 0.5 m. These oscillations are more significant for higher values, which is why the set point should not be put too far from the MAV. To travel greater distances, the set point should be displaced gradually, not in large steps.

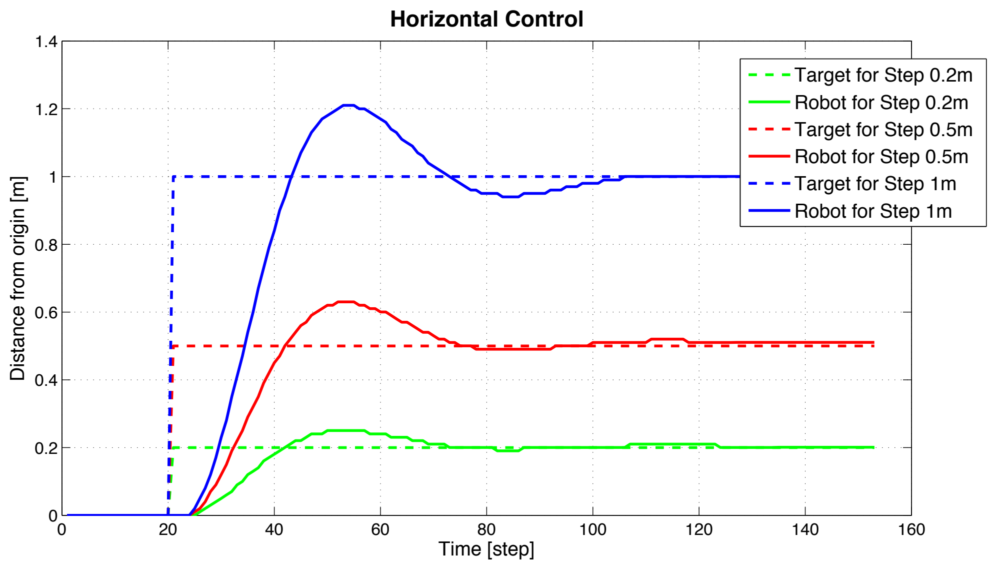

Figure 2.4: Horizontal control

It is also important to notice that the horizontal controller does not handle consecutive changes of direction as well as the vertical controller. The system is stable but creates greater overshoots and stabilizes later (Figure 2.5).

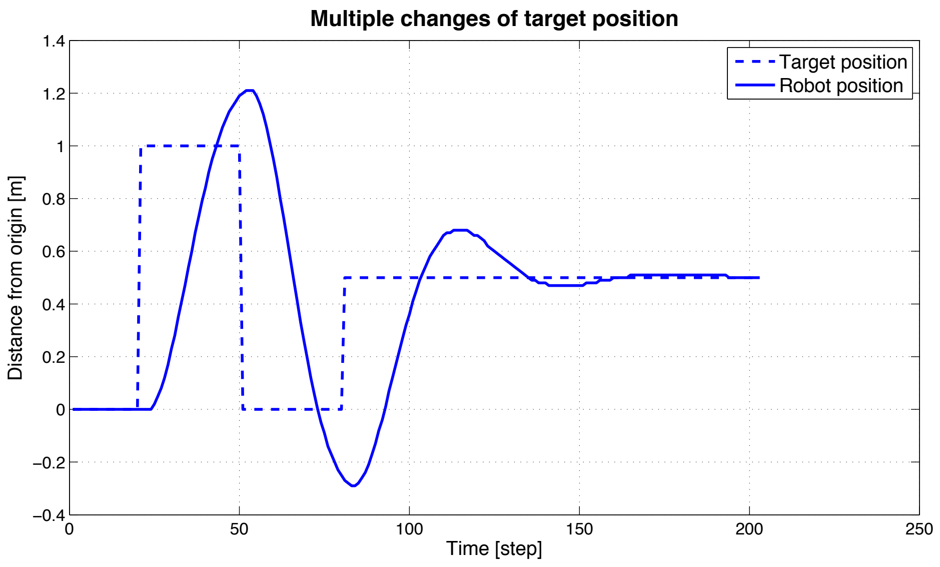

Figure 2.5: Horizontal control - consecutive changes

Another interesting analysis would be to compare the MAV movements in different directions (Figure 2.6). The main axis of the MAV is the axis going in the direction of the body of the MAV (X on the figure). Movement in the direction of this axis, the direction perpendicular to it (Y) and diagonal (D) is compared and analyzed.

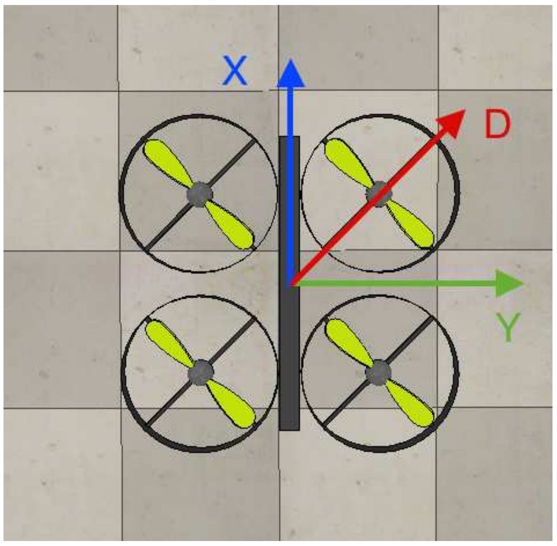

Figure 2.6: Different direction of the movement

Simulations have been run in all of these three directions (Figure 2.7) and the following is noticed. There is, as expected, no crucial difference in the movement along the X and Y axis. The slight asymmetric distribution of mass is negligible. The movement along the diagonal is different from the other two. It does not reach the required position as fast, first stabilizing approximately at 95% of the desired distance, then slowly reaches 100%. The same phenomenon has been observe in all of the diagonal directions. This is probably an inaccuracy in the motion controller but it can be considered negligible as the difference is small.

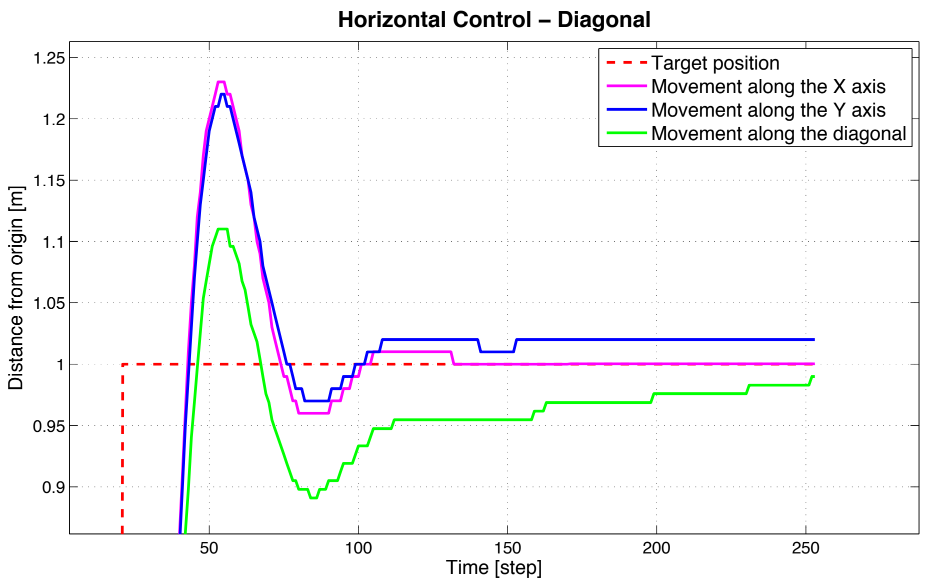

Figure 2.7: Effect of the direction on the step response

2.2.3 Rotation control

The third dimension in which the MAV can be controlled is the rotation (the yaw - rotation around the Z axis). Every adjacent rotor must turn in the opposite direction to stabilize yaw. In this way the resulting torque is null. By adding thrust on the clockwise turning propellers and subtracting the same amount on the counter-clockwise propellers, the MAV starts spinning clockwise on the spot. Rotation is handled much worse by the controller in V-REP than translation in the horizontal or vertical direction. The overshoot is about 40% and it takes more than 10 seconds for the system to be stabilized (Figure 2.8).

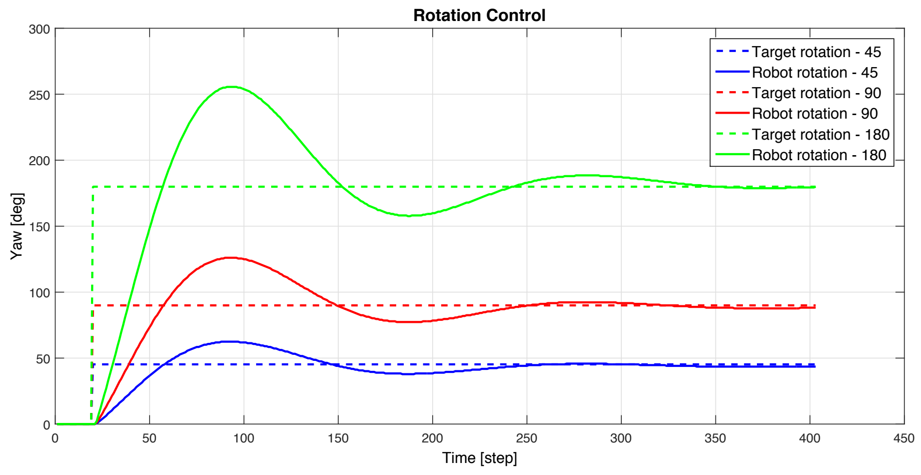

Figure 2.8: Rotation control

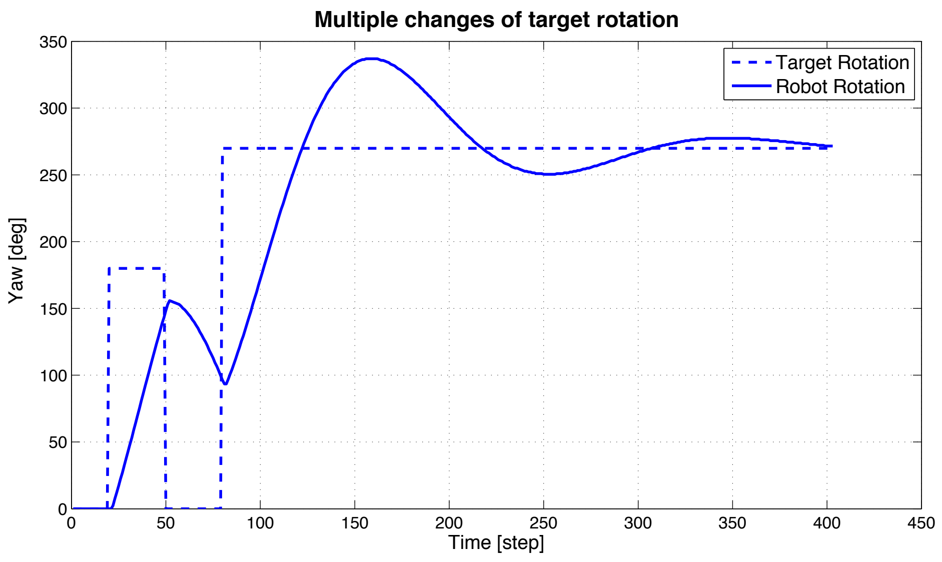

Figure 2.9: Rotation control - consecutive changes

On both Figure 2.8 and Figure 2.9 the MAV reacts very quickly, almost immediately, to changes in the yaw of the set point. It has, however, problems reaching a stable state, oscillating for more than 10 seconds around the desired value. We deduce that the simulator's controller is not well designed for rotary movement. This is the reason why rotation is not used in this work, as it is not essential to the implementation of the algorithm. The MAV itself can perform any translation in 3D without needing to rotate. Yaw could be useful if a special component, such as a camera or a sensor, is headed in one specific direction and we need to turn it in another.

2.3 Adaptation of the coherence swarming algorithm for MAVs

We will now describe the steps of implementing the algorithm introduced in chapter 1 on an MAV swarm with the software components available in V-REP. This will enable the algorithm to be simply transferred to real MAVs, as most of these components have already been implemented on real vehicles. We will implement successively the different states introduced above: Forward, Avoidance and Coherence.

2.3.1 Forward state

In the Forward state of the algorithm, the MAV is constantly moving at a set speed. We implemented this behaviour, using the available PID controller in V-REP, by placing at every time step the set point indicator to a determined distance in the direction of the robot's movement (Figure 2.10).

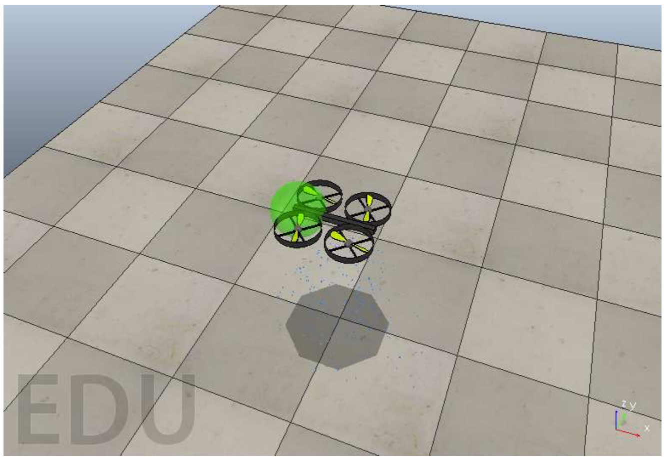

Figure 2.10: Constant motion

2.3.2 Avoidance state

The second behaviour that has to be implemented is obstacle avoidance. A cylindrical omnidirectional proximity sensor is used in V-REP to detect objects in proximity to each MAV (Figure 2.11). The sensor is composed of two concentric rings. The outer ring represents the maximal range of the sensor and the inner ring represents the minimal range, which is necessary to avoid detecting the MAV's own propellers.

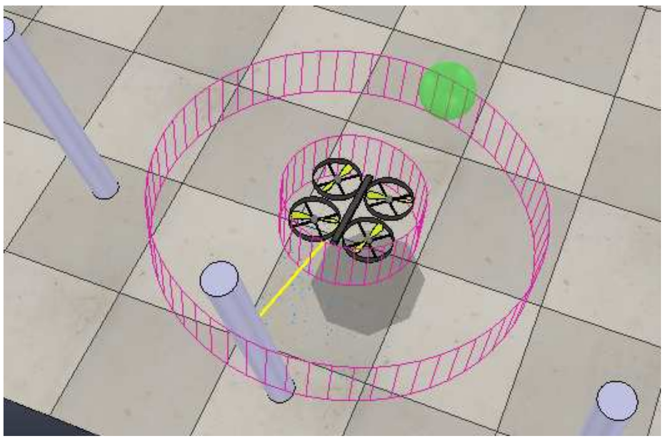
Figure 2.11: Collision sensor

If the sensor detects an obstacle, the set point is placed at a certain distance in the opposite direction. The MAV performs a swift evasive manoeuvre and continues its journey in this new direction. This is a very basic method of obstacle avoidance, but it is robust and computationally inexpensive. To test and tune-up this behaviour, we build an environment (Figure 2.12) with a ring of obstacles in which MAVs fly in straight lines until forced to avoid collision with themselves of the surrounding obstacles.

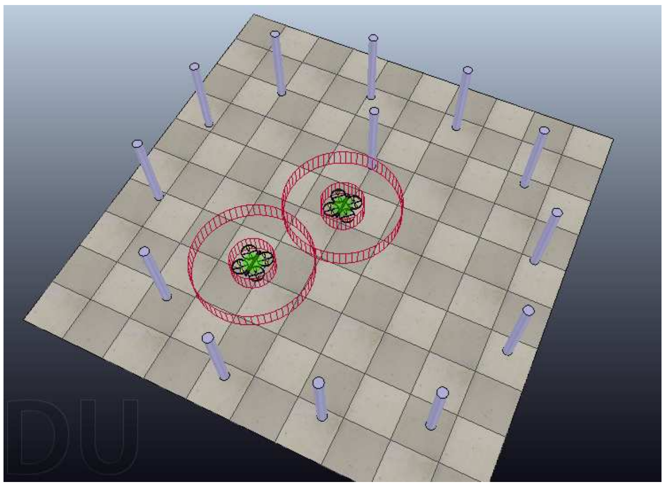
Figure 2.12: Environment with obstacles to test the avoidance algorithm. Video available at `http://youtu.be/W9QcrnLVI8Y`.

We tuned up the range of the sensor and the sensitivity of the reaction (how far the set point is placed) to avoid collisions without being too aggressive to achieve a smooth flight.

The robustness of the algorithm has been tested by measuring the minimal distance from the MAVs to each other and to the nearest obstacle (Figure 2.13 and Figure 2.14). In 10 minutes of simulation, the minimal distance recorded descended to about 0.57m for the distance between MAVs and 0.47m for the distance between the MAVs and the obstacles. This margin of security is sufficient as no collision occurred and the MAVs never got into too close of proximity to be in danger.

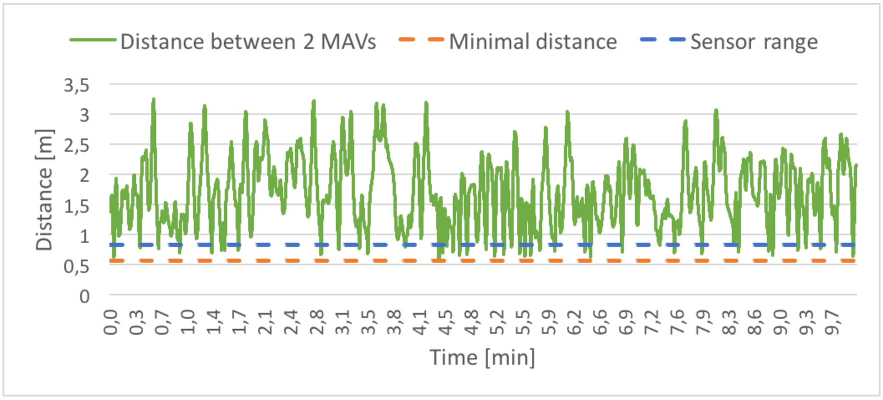

Figure 2.13: Distance between the MAVs

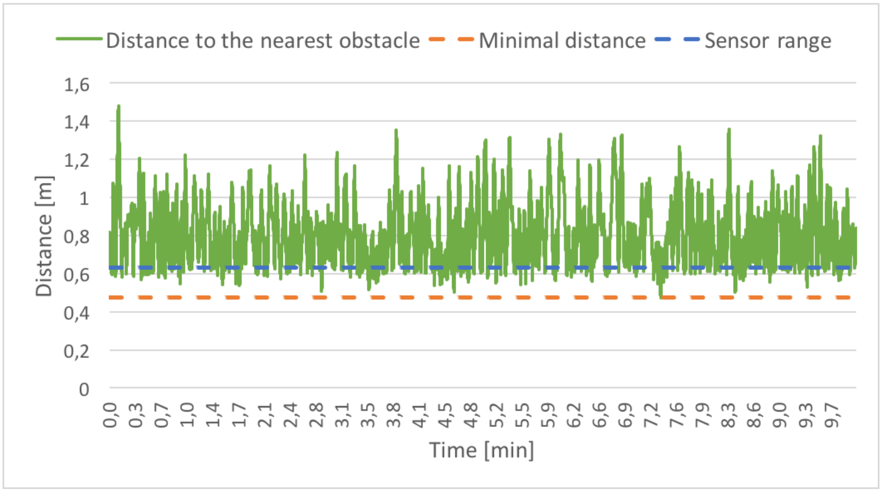

Figure 2.14: Distance between the MAVs and the nearest obstacle

2.3.3 Coherence state

A second proximity sensor is added on the MAV, similar to the one above but with a wider range (Figure 2.15).

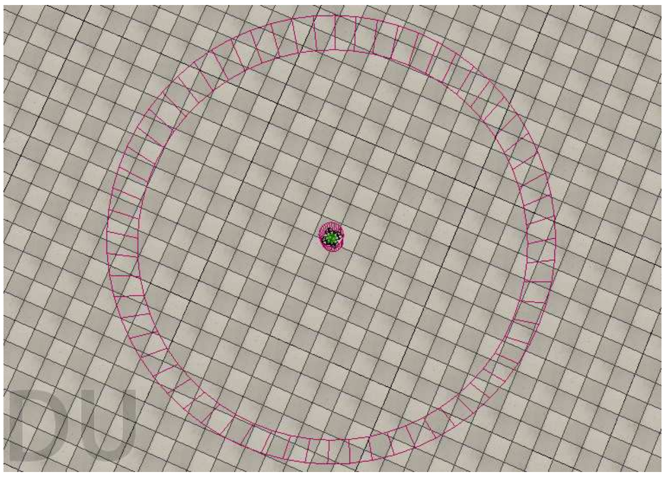

Figure 2.15: Coherence sensor

The task of this sensor is to count the number of MAVs within its range. Whenever this number diminishes (a neighbor is lost), the vehicle turns back and flies in the opposite direction. After a certain number of time steps, during which it does not react to any neighbor loss, it performs a random turn, deviating from $-\pi/4$ to $\pi/4$ from its trajectory, and continues in its rectilinear movement.

At this point, the cohesion of a smaller MAV swarm is achieved (video available at `http://youtu.be/t_JBYvmICqQ`). With more entities, the swarm becomes very compact as the robots react to the loss of every neighbor. This leads to a MAV performing collision avoidance and coherence manoeuvres more often than would be necessary, which decreases the overall stability of the swarm (as will be explained in Chapter 3).

2.3.4 Alpha algorithm

The above-described algorithm is modified according to [4]. A parameter *Alpha* is introduced that indicates to the robot how many neighbors it can let go before triggering its coherence maneuver. For example, for $\alpha = 5$, MAVs ignore the loss of their 6th neighbor, but react to the loss of their 5th one.
We have now an MAV swarm that successfully maintains coherence. Figure 2.16 shows the Alpha algorithm during one simulation. We are now ready to explore the patterns of behaviour emerging from this system.

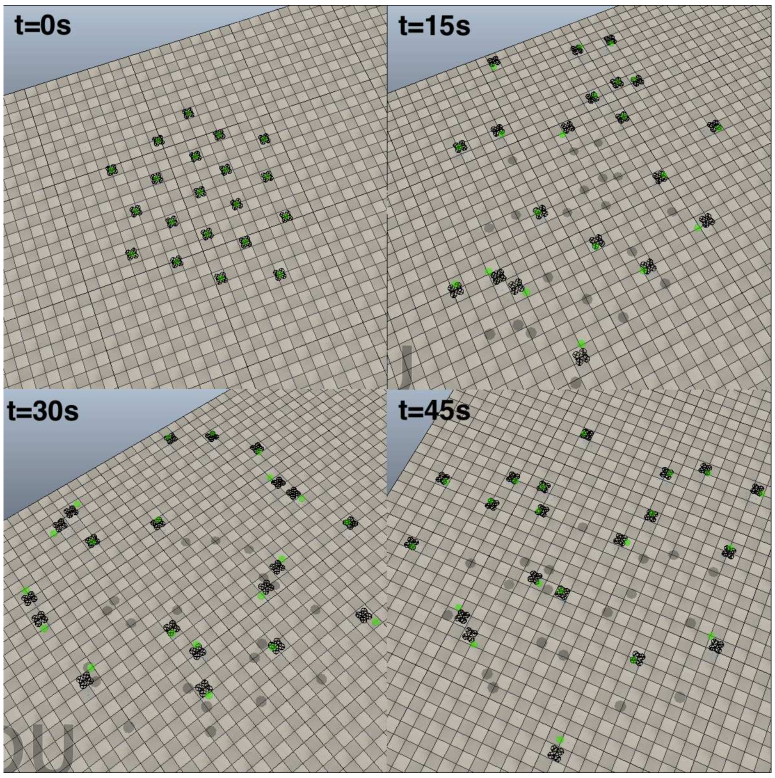

Figure 2.16: Swarm of 20 MAVs with $\alpha = 6$ during one simulation. Video available at `http://youtu.be/YCNWqypC2Po`.

Chapter 3

Parameter Analysis

Parameters of the algorithm will now be tuned up to observe their effect on the behaviour of the swarm and to find their optimal value. For every configuration of parameters 10 simulations of 10 minutes have been performed and the results averaged.

3.1 Factors considered

For each observed parameter, hypothesis are postulated about how it affects certain aspects of the swarm. After running and analysing the simulations, the proposed hypothesis will be evaluated. These are the factors considered in the analysis:

1. **Swarm stability.**
   - Whether or not a swarm is stable (i.e. no MAV is breaking away from the rest of the group) is not clearly definable. We decided to quantify it in the following way: stability is represented by the ratio of all time steps where all robots are not together in the formation to the total time steps elapsed. This means the swarm is considered as "unstable" from the first moment the MAV leaves the swarm.
   - Stability is the most important aspect to consider when designing a coherent swarming mechanism. Obviously it should always be maximized.
2. **Swarm width.**
   - The size of a swarm is also a variable not clearly defined. It has been defined here as the standard deviation of the position of all the MAVs in the swarm. This may not give us the exact absolute value of the size of the swarm, but in this case it does not matter as we are comparing the effects of various parameter on this size. The error induced is constant.
   - In most cases, the swarm width is maximized as it is efficient at covering a large area with a minimum of robots.

3. **State distribution.**
    - The last interesting property is the portion of time MAVs spend in the various states introduced. These values are easily obtainable as robots know in which state they are in.
    - In most cases, we seek to maximize the portion of time spent in the Forward state. In this state, the MAV is most efficient at performing any task given, since it does not have to perform any complex manoeuvre at the same time. Poorly negotiated manoeuvres can also lead to the robot breaking off from the swarm or even crashing.

3.2 *Alpha* parameter and robot number

Simulations are run with 5, 10 and 20 robots, each with multiple variations of the *Alpha* parameter. *Alpha* is the threshold indicating the number of neighbors each MAV tries to maintain.

3.2.1 Hypothesis

Six hypothesis about the behaviour of the swarm were tested:

1. Higher *Alpha*
   (a) makes the swarm more stable.
   (b) makes the swarm more compact.
   (c) increases the time spent in Avoidance and Coherence state.
2. A greater number of robots
   (a) increases the swarm stability.
   (b) increases the swarm width.
   (c) increases the time spent in Avoidance and Coherence state.

3.2.2 Simulation results and evaluation of the hypothesis

The data obtained in simulations are displayed on the following graphs (Figure 3.1, Figure 3.2 and Figure 3.3), showing the behaviour of different-sized swarms with different values of *Alpha*.

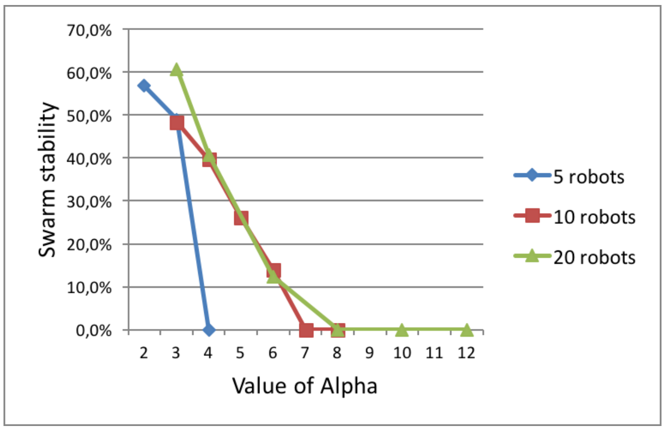

Figure 3.1: Swarm stability

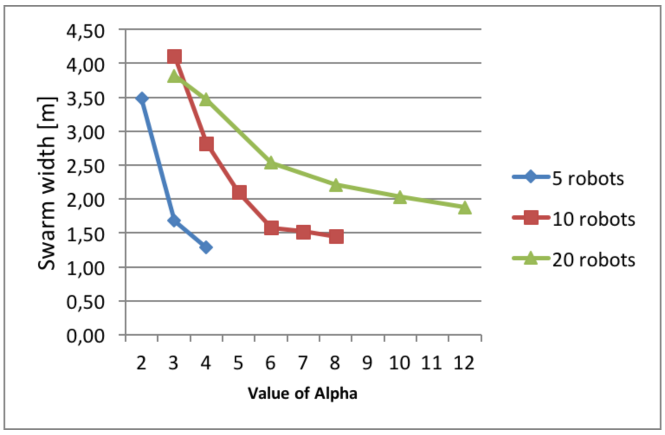

Figure 3.2: Swarm width

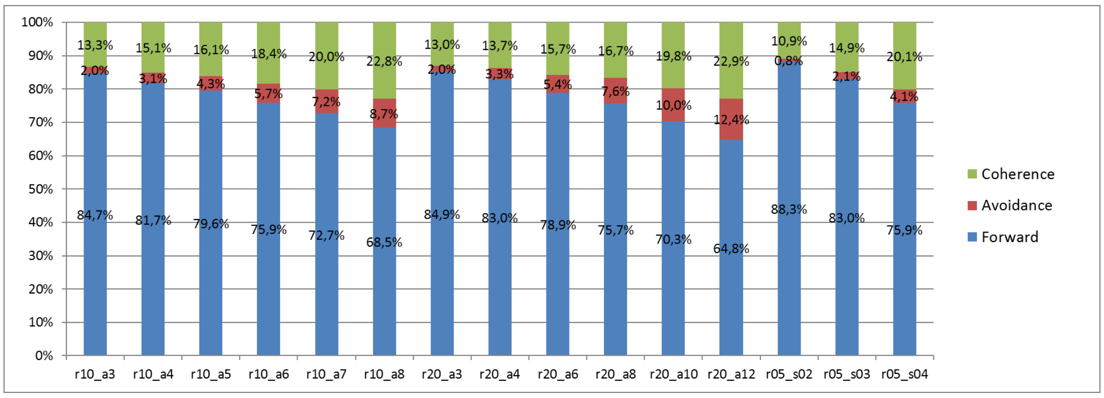

Figure 3.3: State distribution. The labels on the x-axis indicate the used configuration of the swarm (r05_a04 represents a swarm of 5 MAVs using an *Alpha* parameter of 4)

The previously presented hypothesis will now be evaluated.

1. Higher *Alpha*
    (a) indeed increases the stability of the swarm. For bigger swarms, the probability for the swarm to break apart decreases almost linearly, reaching 0% at $\alpha = 7$ and staying there for higher values of *Alpha*.
    (b) decreases the swarm width as we expected. The differences are more evident for smaller values of *Alpha*.
    (c) increases the time spent in the Avoidance (due to the increased compactness of the swarm) and the Coherence (caused by an increased sensibility to neighbor loss) state.
2. A greater number of MAVs
    (a) does not increase the swarm stability, contrary to our hypothesis. The results indicate there is an optimal value of *Alpha* (here 7 or 8) that is enough to guarantee the stability of the swarm. This value is not dependent on the number of robots, but on other parameters (such as the sensor range, size of the MAVs, their flight speed).
    (b) increases the swarm width, but this increase is not proportional.
    (c) does not increase the time spent in Avoidance and Coherence states. For a given *Alpha*, the state distribution is roughly similar for different swarm sizes. The behaviour of an individual MAV is influenced only by its local surrounding, not by the total number of robots.

In the following simulations, we will use a configuration of 10 robots with *Alpha* = 5. It is a sufficiently good configuration, but still results in some instability. In this way, the positive impact of other parameters on the swarm stability can be measured.

3.3 Sensor range

We will now show the impact of the range of the neighbor detecting sensor on the overall behaviour of the swarm. The basic range used in all previous experiments will be called R (R is 4 meters). A series of experiments will be executed where first the range of the sensor is doubled and halved ($2R$ and $R/2$), then the area covered by the sensor doubles and halves ($\sqrt{2}R$ and $R/\sqrt{2}$).

3.3.1 Hypothesis

1. Wider range
   - (a) increases the time spent in Forward state.
   - (b) increases stability.
   - (c) increases the swarm width.

3.3.2 Simulation results and evaluation of the hypothesis

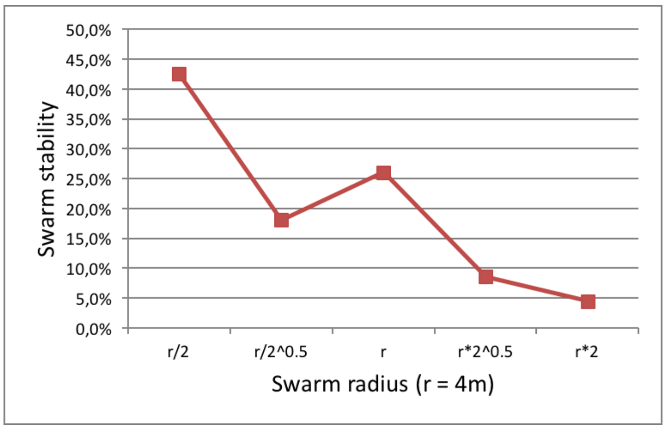

Figure 3.4: Swarm stability

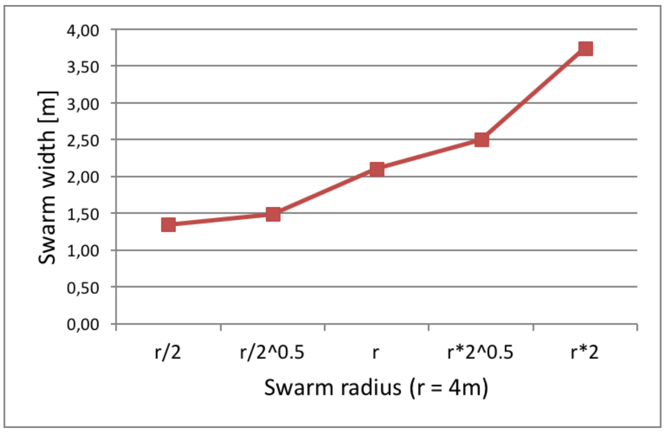

Figure 3.5: Swarm width

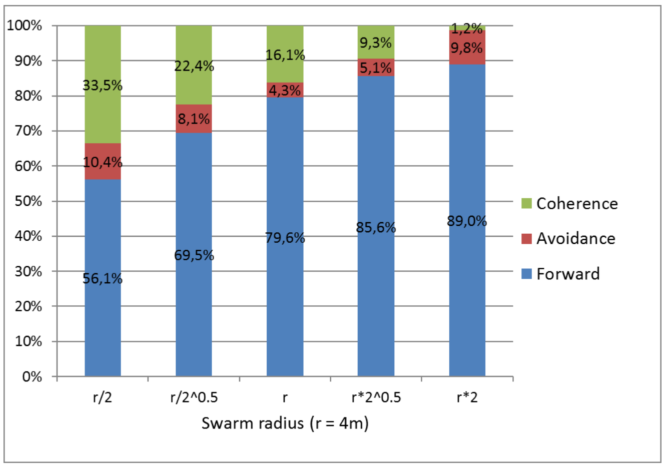

Figure 3.6: State distribution

Figure 3.4, Figure 3.5 and Figure 3.6 display the influence of the sensor range on the behaviour of the swarm. This influence is in accordance with our hypothesis:

1. Wider range
   - (a) increases the time spent in the Forward state, mainly to the expend of time in the Coherence state. The MAVs travel a longer distance until they have to turn back.
   - (b) increases stability. This is due to the reduced time spent in the Coherence state, where poorly managed manoeuvres can result in the MAV breaking apart from the rest.
   - (c) increases the swarm width. With a greater range individual MAVs will have more space between them.

A greater range produces a better solution, as it enhances stability, increases the size of the swarm and limits the time spend in the Coherence state. However, longer range sensors are also more demanding in terms of encumbrance, computational complexity and cost.

3.4 Noise in sensor reading

Up to now the system was considered to be noiseless. As losing and regaining contact between robots is crucial to our algorithm, adding noise might be a source of instability for the swarm. A uniform probability has been introduced indicating the chance for the sensor not to detect a neighbor in range.

3.4.1 Hypothesis

1. Noise in the sensor
   (a) increases the time spent in the Coherence state.
   (b) decreases stability.
   (c) decreases the swarm width slightly.

3.4.2 Simulation results and evaluation of the hypothesis

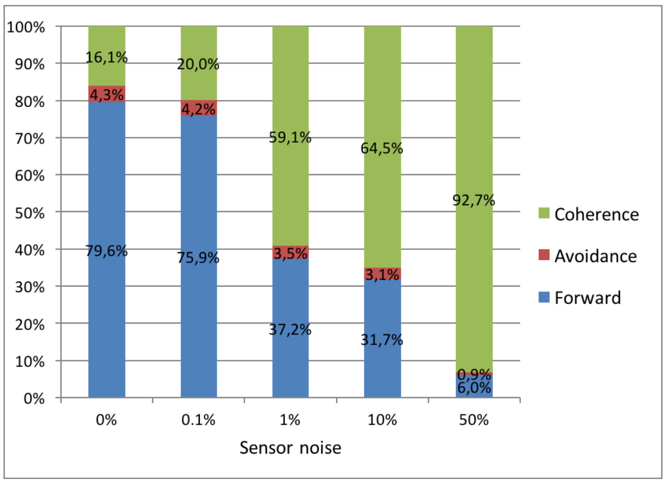

Figure 3.7: State distribution

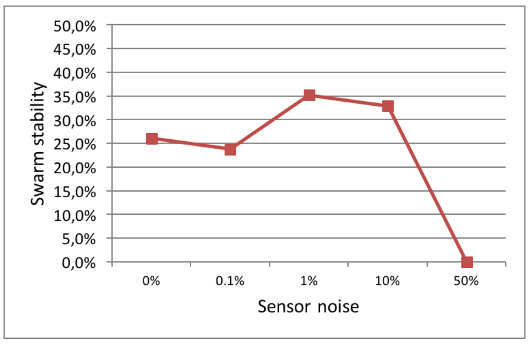

Figure 3.8: Swarm stability

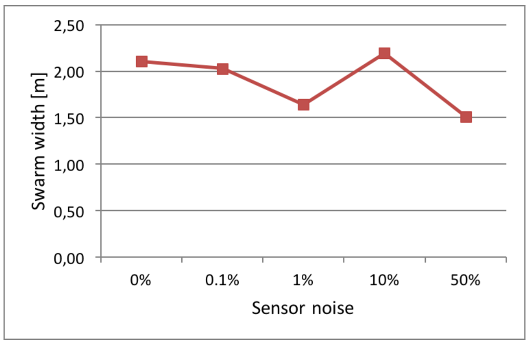

Figure 3.9: Swarm width

Figure 3.7 indicates the system can handle noise causing a loss of connection with a probability up to an order of magnitude of 0.1%. A higher noise level causes the MAVs to spend most of their time in the Coherence state without actually loosing any neighbor.

1. Noise in the sensor

   (a) drastically increases the time spent in the Coherence state, up to a point where it becomes the dominant state.

   (b) and (c): Stability and swarm width cannot be effectively measured to a swarm with such a high level of noise that the algorithm doesn't work. Individual MAVs are just flying back and forth, thinking they are losing connections.

If, during the implementation of this algorithm on real robots, noisy sensors are used, the probability of unsuccessful reading must be kept under 0.1% for the algorithm to work correctly. If the sensors are less reliable, filters could be introduced to increase the performance of the solution.

3.5 Sensor angle of view

The real MAV model described in [2] uses visual sensors to detect and identify nearby MAVs. Until now an omnidirectional sensor has been used in the implementation of the algorithm. If using visual sensors, there are several possibilities to obtain such a full sensor. Multiple cameras facing different directions to compose the entire field of view might be used. This approach can be expensive, encumbering and possibly computationally demanding. Another solution would be to extend the field of view of the sensor with lenses or mirrors. This, however, creates a distorted field, where the range, resolution, accuracy and robustness of the sensor is not equally distributed in all the directions. We claim the cohesion of the swarm can be maintained by equipping the MAVs with a single sensor with a reduced angle of view. We tested this hypothesis with sensors with reduced angle of view to 180 and 90 degrees. All robots are placed in their initial poses facing the center of the swarm as shown on Figure 3.10.

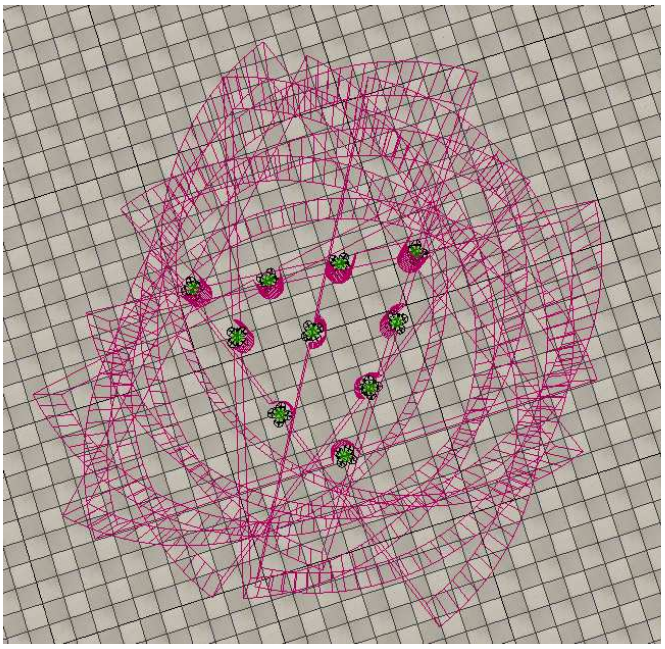

Figure 3.10: Swarm with the sensor with a limited angle of view. Video available `http://youtu.be/uHPyTaaaqqE`.

3.5.1 Hypothesis

1. Narrower angle of view
   (a) decreases stability.
   (b) decreases the swarm width slightly.
   (c) increases the time spent in the Coherence state.

3.5.2 Simulation results and evaluation of the hypothesis

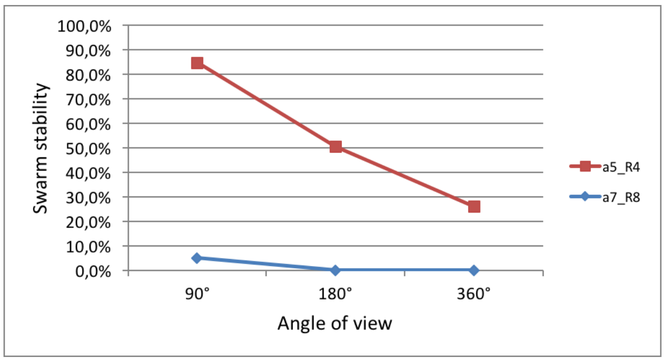

Figure 3.11: Swarm stability for different configurations of *Alpha* and sensor range (a5_R4 stands for $\alpha = 5$ and sensor range $R = 4m$)

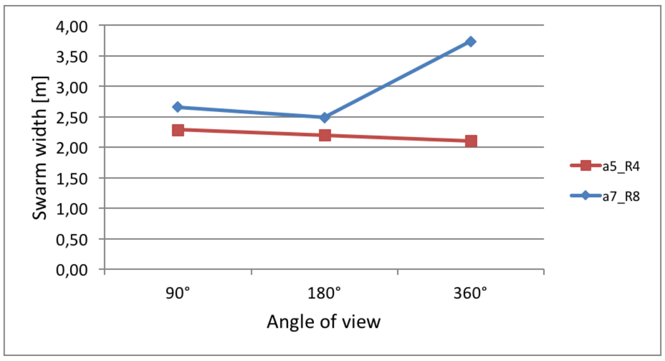

Figure 3.12: Swarm width for different configurations of *Alpha* and sensor range (a5_R4 stands for $\alpha = 5$ and sensor range $R = 4m$)

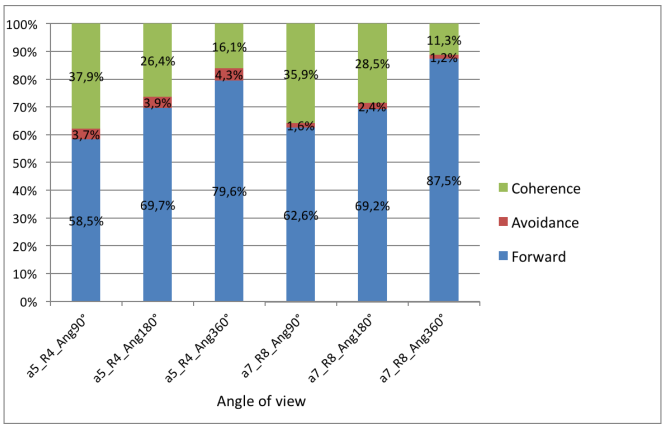

Figure 3.13: State distribution for different configurations of *Alpha* and sensor range (a5_R4_Ang90 stands for $\alpha = 5$ and sensor range $R = 4m$ and sensor angle 90 degrees)

Two parameter configurations were used in these simulations. The first configuration ($\alpha = 5$ and sensor range $R = 4m$) has an average performance with a full sensor. As shown on Figure 3.11, this swarm is highly unstable for reduced angles of view. We then chose a better *Alpha* with a greater sensor range ($\alpha = 7$ and $R = 8m$) to compensate the reduced angle and received much better results.

Based on these results, we can confirm our hypothesis on the impact of the reduced angle as follows.

1. Narrower angle of view
   (a) decreases stability. We must be careful to choose good parameters if we want to use a reduced angle. The setting with an average average performance with the full sensor becomes unusable while the good one holds on.
   (b) decreases the swarm width, although we might need more data to make a well-founded conclusion.
   (c) increases the time spent in the Coherence state, while the occurrence of the Avoidance state remains roughly the same.

As we have seen, the proposed algorithm works well even if the angle of view of the sensors is reduced. We must be careful however to use a well tuned *Alpha* value and invest a little of the resources saved when narrowing the angle into extending its range.

Chapter 4

Directed Swarming

A swarm stabilizing algorithm has been successfully implemented (in chapter 2) and tuned (in chapter 3). This algorithm will now be expended to allow the swarm to follow a target, enabling it to move in space.

4.1 Principle

Let us first imagine that the target is a simple light beacon. There are multiple approaches to create a target following algorithm that respects the restrictions set on the communication and sensor capacities of the MAVs.

One approach (the one discussed in [4]) is to equip the MAV with a binary light sensor that can recognise whether the target is in sight or not. By sharing this information with its neighbors, the whole group can have a vague idea of the target's location based on the fact that the vehicles that do not see the target are further away or behind obstacles. However this would necessitate more complex communication channels between individuals, and we want to limit that in our work.

The second approach would be to make a rough on-board estimate of the target's general direction. We use a couple of sensors (4 are used in this implementation) and deploy them around the robot. The position of the target is allocated in this way to the corresponding quadrant. Although it is a very inaccurate measurement, our argument is that a swarm can use this method to effectively drift towards the target.

We also chose this method having in mind the implementation of the proposed swarming algorithm on real MAVs. There, a set of 4 cameras pointing in different directions is used for neighbor detection as well as for obstacle avoidance [2]. The task of detecting the target can also be assigned to these cameras and no additional sensor is needed.

4.2 Implementation

To implement in V-REP the behaviour described above, the MAVs were equipped with four additional long range proximity sensors (Figure 4.1).

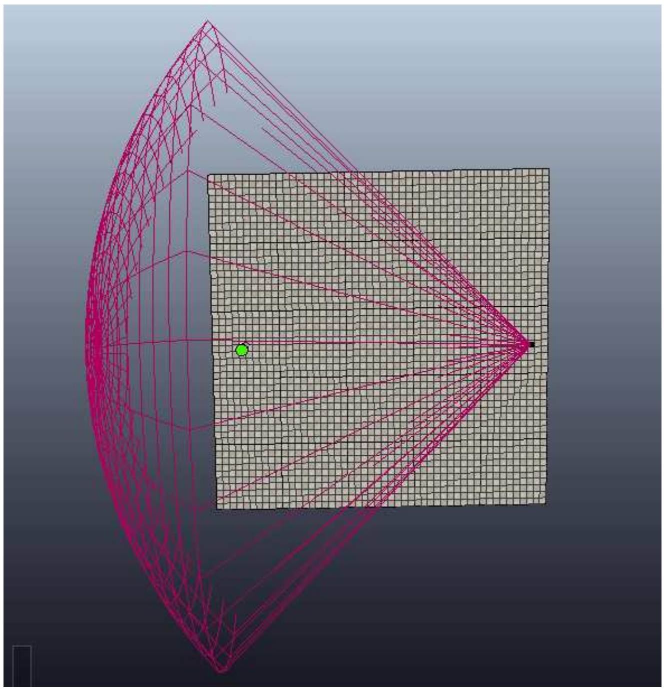

Figure 4.1: One of the target detecting sensors. The other 3 are hidden for clarity. The target is represented by the green object on the left.

We explained earlier how the robot performs a random turn when emerging fromthe Coherence state. Now the direction of the random turn is shifted towards the activated sensor. This is depicted on Figure 4.2 and Figure 4.3. The randomness is preserved, its center is just shifted accordingly.

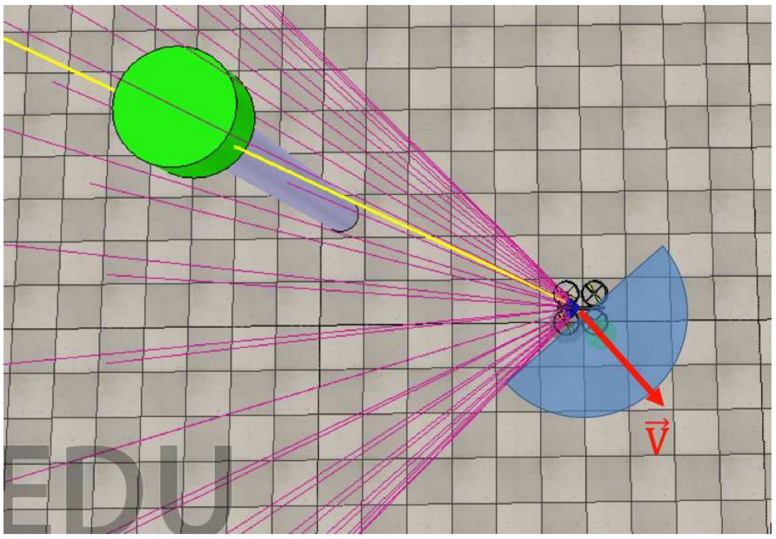

Figure 4.2: Original random turn. $\vec{V}$ is the velocity vector of the MAV. The blue area indicates the possible angle of the random turn.

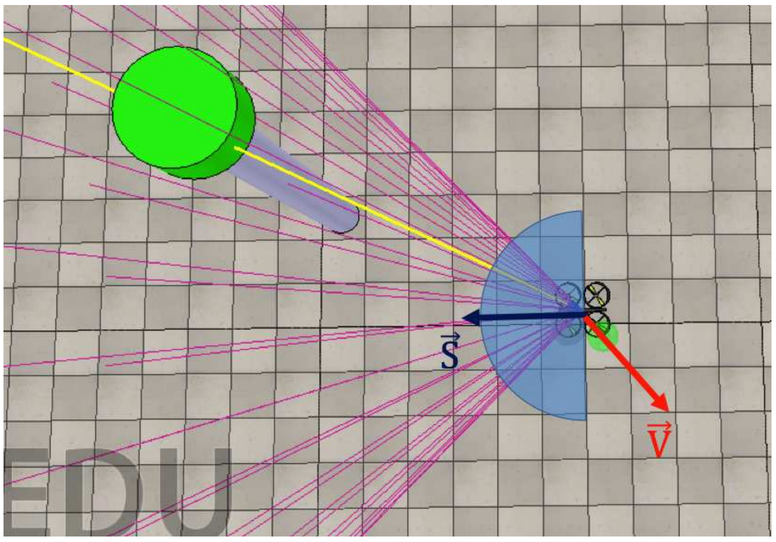

Figure 4.3: Random turn modified for the target following algorithm. $\vec{S}$ represents the direction the sensor detecting the target is headed. The blue area indicates the possible angle of the random turn.

The possibility of occlusion of the sensor by a neighboring MAV or an obstacle is also taken into account. This enables a realistic simulation of large swarms, where only the MAV in the front can see the target. It is also necessary when simulating directed swarming in environments with obstacles.

4.3 Results

A series of simulations will be performed to test and analyse the behaviour of the directed swarm. Figure 4.4 displays the movement of the swarm during one simulation. The position of the center of the swarm (the average position of all the MAVs - depicted in red on Figure 4.4) is tracked and used in the following comparisons. The initial distance between the swarm and the target is 20 meters.

Swarms of different sizes (4, 9 and 18 MAVs) will be compared according to two factors:

1. The distance of the swarm to the target during the simulation.
2. The mean distance from the ideal (straight line) trajectory. This indicates how much the swarm deviates from the line defined by the position of the set point and the initial position of the center of the swarm.

Four hypothesis about the movement of the MAV swarm towards the target were analysed:

1. The distance to the target decreases linearly (speed is constant) at first, then it is stabilized around 0 as the target location is reached.
2. A smaller swarm reaches the target faster.
3. The swarm does not deviate much from its ideal trajectory - a maximum of 4 meters of deviation for 20 meters of total distance.
4. Bigger swarms to deviate more from their ideal trajectory.

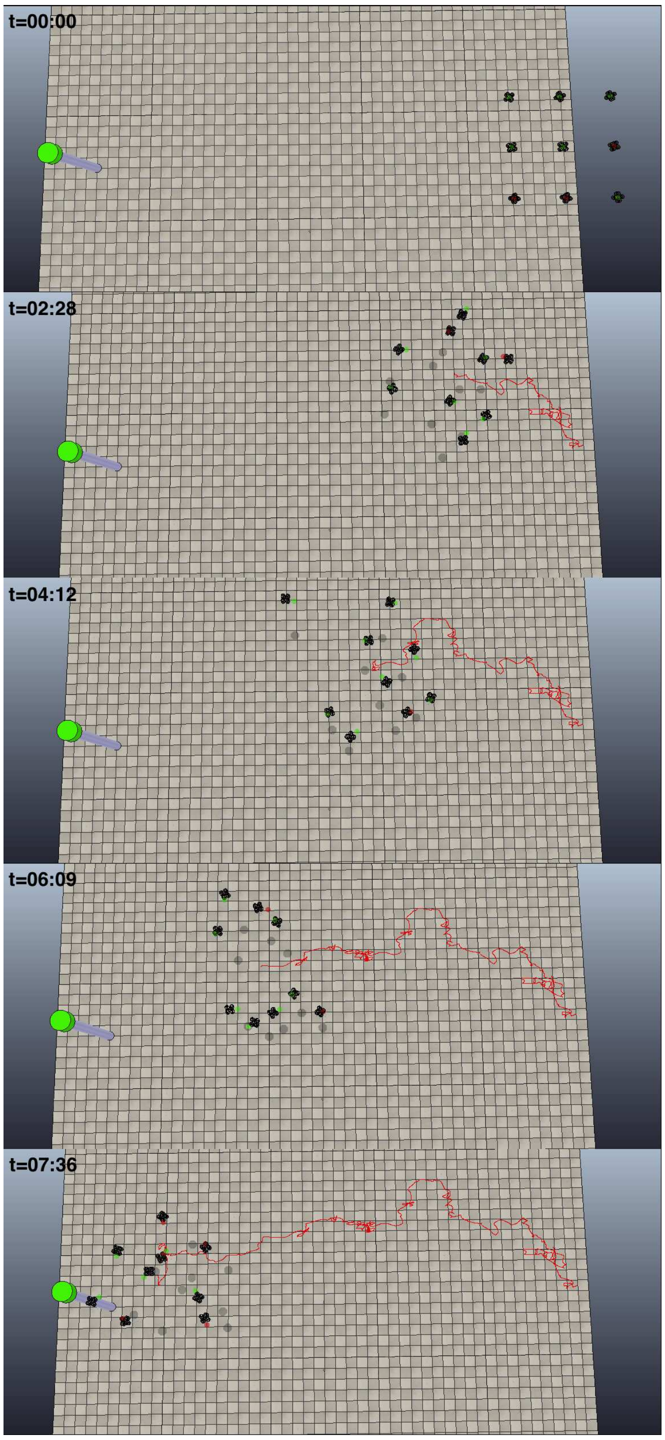

Figure 4.4: Swarm of 9 MAVs executing the directed swarming algorithm. The position of the center of the swarm is drawn in red. Video available at `http://youtu.be/_GurlF9shCg`.

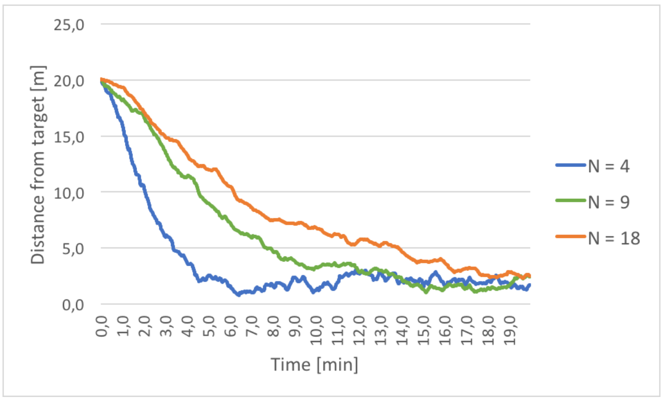

Figure 4.5: Distance of the center of the swarm to the target. N is the number of MAVs in the swarm.

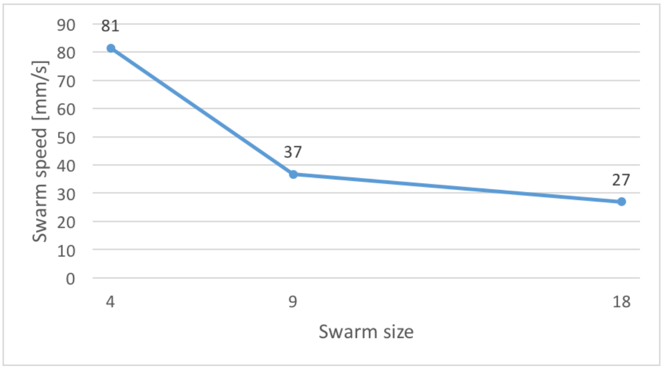

Figure 4.6: Speed of the swarm as a function of its size.

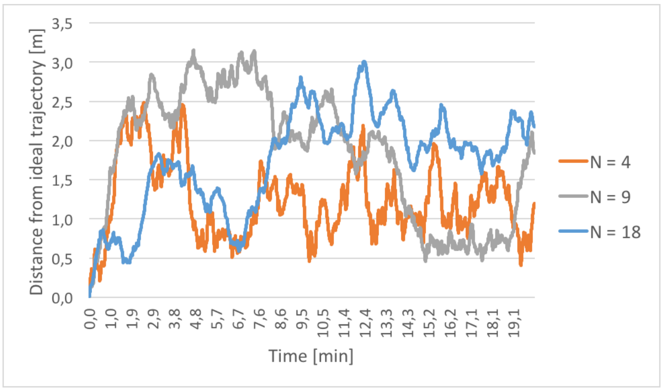

Figure 4.7: Distance of the swarm from its ideal trajectory through time. N is the number of MAVs in the swarm.

After running and averaging simulations with swarms of 4, 9 and 18 robots, the obtained results are displayed on Figure 4.5, Figure 4.6 and Figure 4.7. We will now evaluate the hypothesis we proposed.

1. As expected, the speed of the swarm is constant at first and starts to decrease at 5 to 7.5 meters from the target (depending on the size of the swarm). The swarms stabilise at a distance of about 1.8 meters from the target.

2. As we saw in the previous point, the speed of the swarm can be considered constant until the swarm gets too close to the target. We compared these speeds on Figure 4.6. According to our results, bigger swarms move slower. This is because the MAVs must react to more neighbour losses in bigger swarms and the vehicles in the the back are often occluded by those in the front.

3. The deviation for all three sets oscillates between 0.5 and 3 meters. This is considered this very good for a total distance of 20 meters.

4. We cannot discern clear patterns between the distinct swarms. Either there is no clear behaviour differences or the statistical set used in this work (10 simulations for each setting) is not sufficient.

A target following algorithm that extends the basic alpha swarming algorithm has been successfully implemented without adding any demands on the communication capacities of individual MAVs, requiring only minimal additional sensorial an computational capacities. The effectiveness of our approach, speed of the swarm, decreases with its size. For very large swarms, a different solution might be better suited.

4.4 Environment with obstacles

The presented target following algorithm is tested in a environment with different kinds of obstacles. The algorithm is well designed for environments with obstacles sparsely distributed in the space. For example it cannot, like other more explicitly controlled swarming algorithms, make the swarm fly through a narrow corridor by aligning individual MAVs in a row.

The different obstacles in the environment are displayed on Figure 4.8 and described on the following list:

- A: a window 4 meters wide (the diameter of the swarm is 10 meters)
- B: a long wall the swarm flies along
- C: an open space with columns
- D: a moving obstacle (column). The obstacle must be slower than the MAV's speed during it's evasion manoeuvre, so that it does not collide with them.

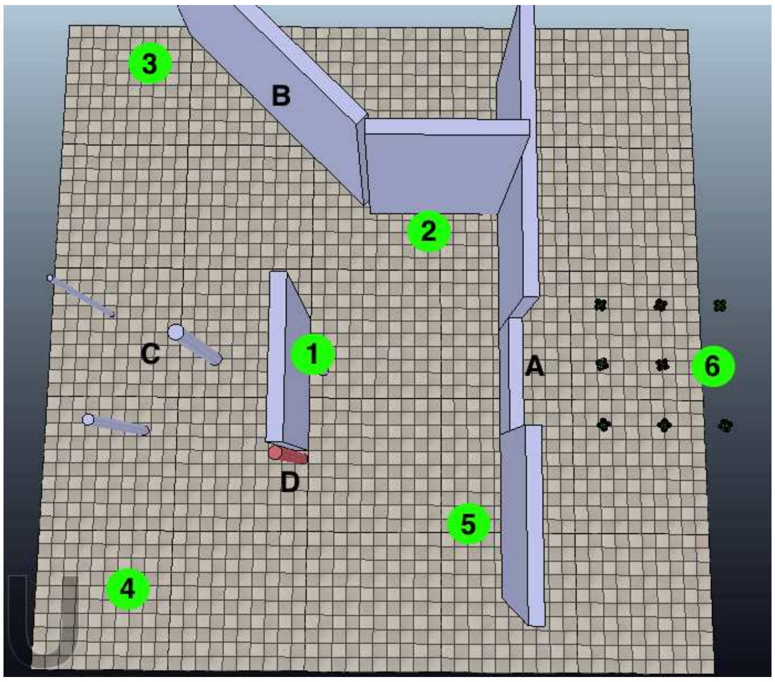

Figure 4.8: Environment with obstacles. The capital letters represent the different kinds of obstacles. The green numbered circles indicate the successive position of the target.

The obstacle avoidance algorithm presented in chapter 2 has been retuned to suit this new environment. The range of the sensor has been slightly reduced to enable better passing through tight spaces. The intensity of the avoidance manoeuvre has been increased (the set point is placed further from the MAV), in part to compensate for the shorter range of the sensor, partly to enable the MAV to dodge the dynamic obstacle.

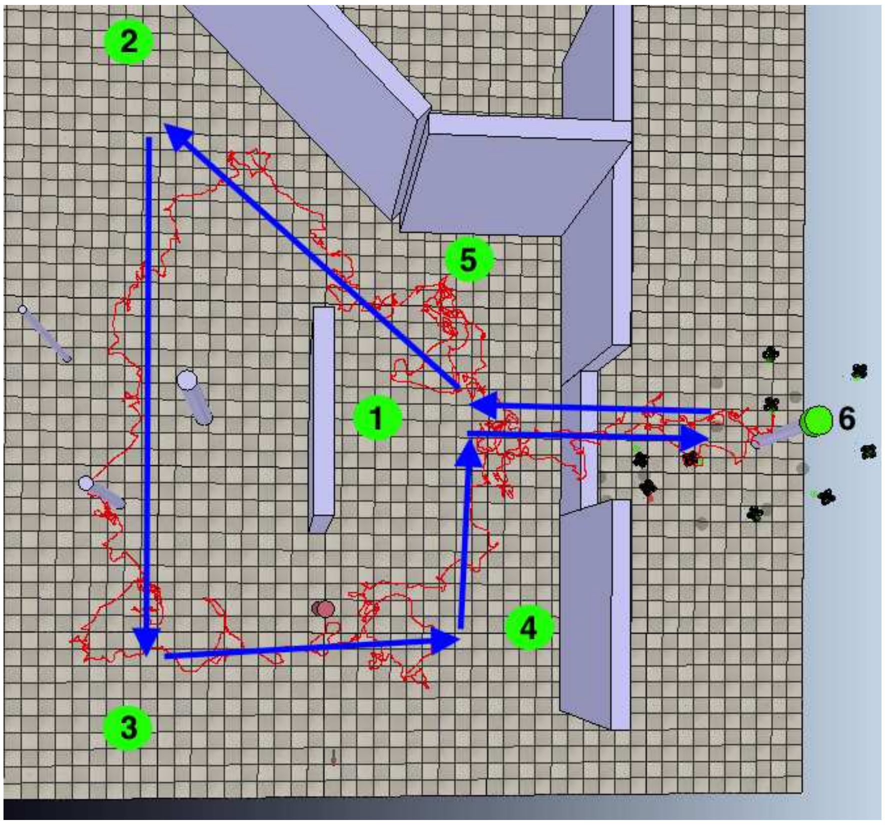

Figure 4.9: Trajectory of the swarm through obstacles. The blue arrows represent the intended trajectory, the red curve represents the actual trajectory of the swarm during one simulation.

After successfully overcoming one obstacle the target automatically moves to redirect the swarm towards another (Figure 4.8 and 4.9). The distance from the center of the swarm to the target was measured and the results displayed on Figure 4.10. It shows how the target changes it's position as the swarm approaches it. Figure 4.11 displays the swarm's trajectory during one simulation run. The algorithm proves to work well in this kind of environment.

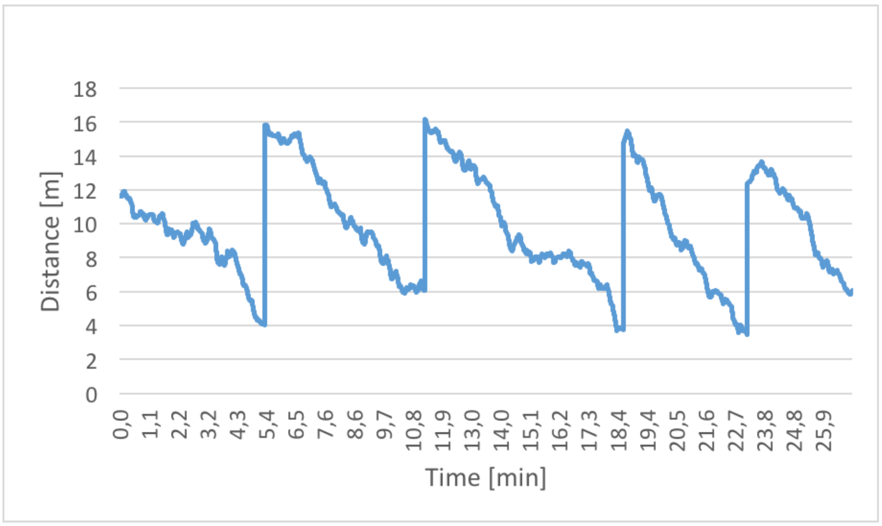

Figure 4.10: Distance of the center of the swarm to the target during one simulation.

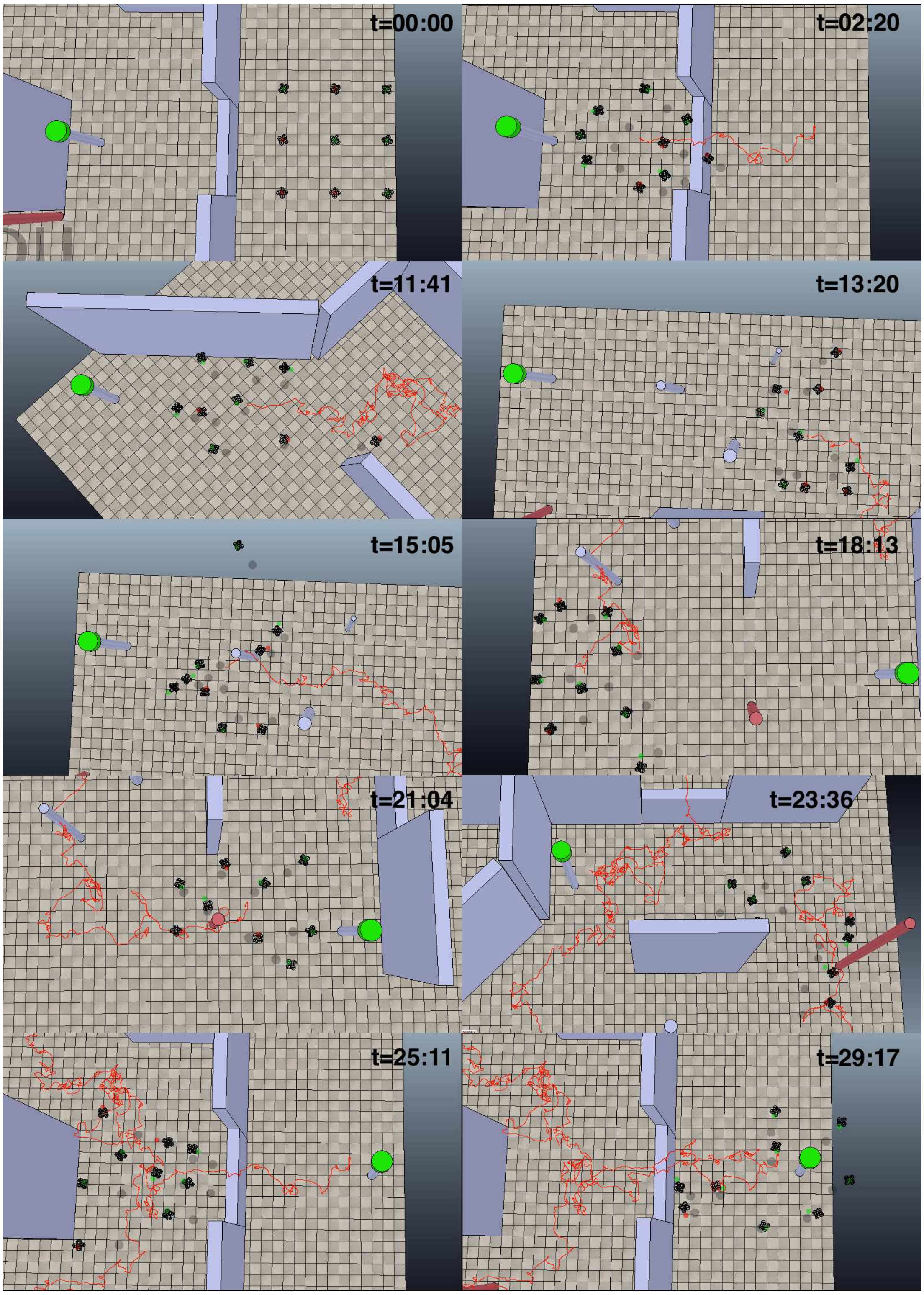

Figure 4.11: Simulation of the target following algorithm in a environment with obstacles. The trajectory of the center of the swarm is depicted in red. Video available at `https://youtu.be/gqFxtVEcEdc`.

Chapter 5

Expansion into 3D

In the previous parts of this work, MAVs were made to move at a constant and commune altitude. Now will be described an extension of this algorithm to allow a general motion of the swarm in space.

5.1 Vertical air flow interference

As we mentioned earlier, propellers of the MAVs create an air flow that negatively affect neighbors in close proximity. In V-REP, this flow is represented by a cloud of bubbles emerging from the MAVs propellers (Figure 5.1). The generated air flow then exerts a force on nearby objects.

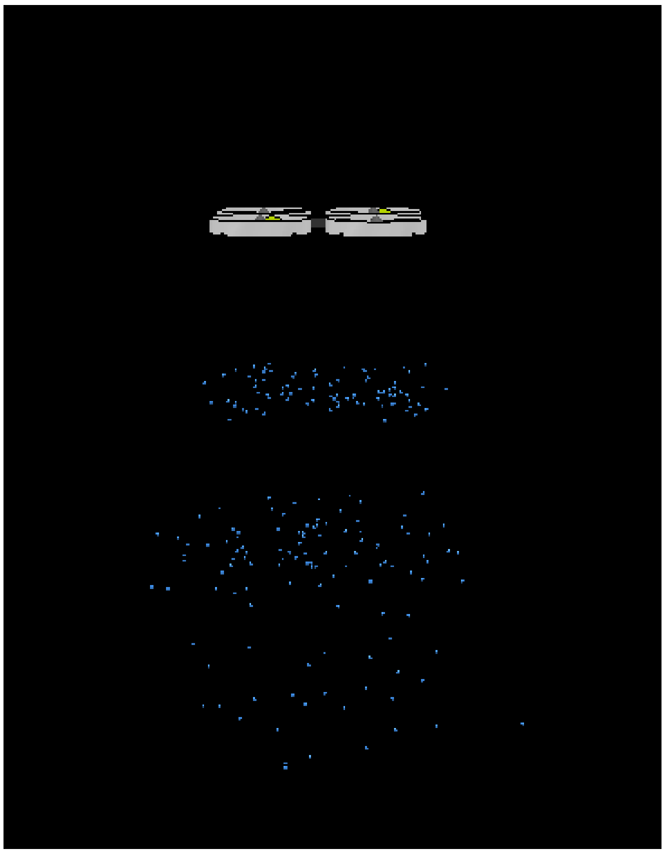

Figure 5.1: Representation of the air flow generated by the propellers in V-REP

Two properties of the air flow implemented in V-REP have been observed. Firstly, the flow is directed almost exclusively in the Z axis of the vehicle. This was something expected and therefore it was not necessary to include the problematic of air flow interference in the 2D approach, where MAVs fly at the same altitude. Secondly, the air flow affects only the area below the MAV and not above. Real propellers do produce much less flow in this direction and so V-REP designers decided to neglect the effect in this way. These two statements have been validated by simulations. To examine the effect of the air flow, a series of simulations has been conducted where two MAVs try to follow parallel trajectories on different altitude levels (Figure 5.2).

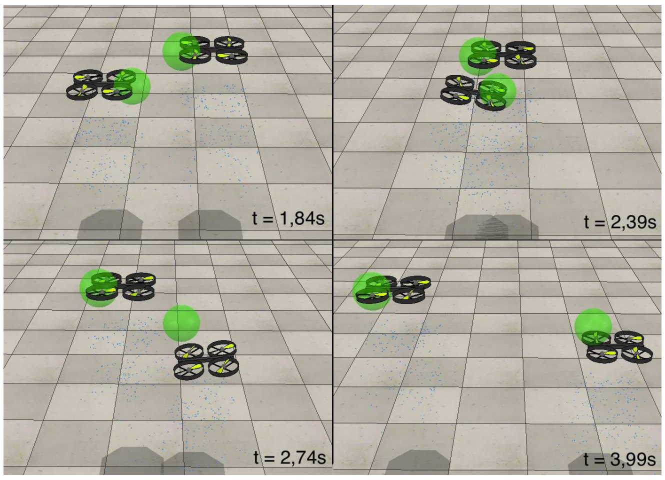

Figure 5.2: Sequence of images depicting the effect of interference by air flow. Video available at `http://youtu.be/aiuGY_pvLoo`.

We tracked the altitude of both vehicles for various altitude differences between them and displayed their trajectories on the following graph (Figure 5.3).

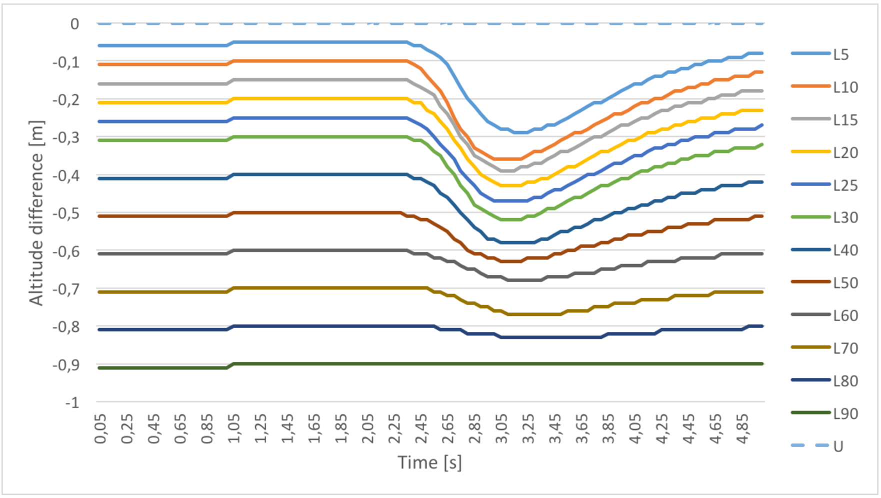

Figure 5.3: Relative altitudes of the lower MAV with respect to the upper one (the altitude of the upper MAV is constant, marked "u"). Different trajectories mark the different altitudes of the set point in cm (L5 is the trajectory of the lower MAV when its set point is 5 cm under the set point of the upper MAV).

As seen on the graph, the air flow generated by the upper MAV influences the lower to up to 80cm of altitude difference. The cylindrical proximity sensor used in obstacle detection was stretched to 1m on both sides to avoid situations of air flow interference (Figure 5.4).

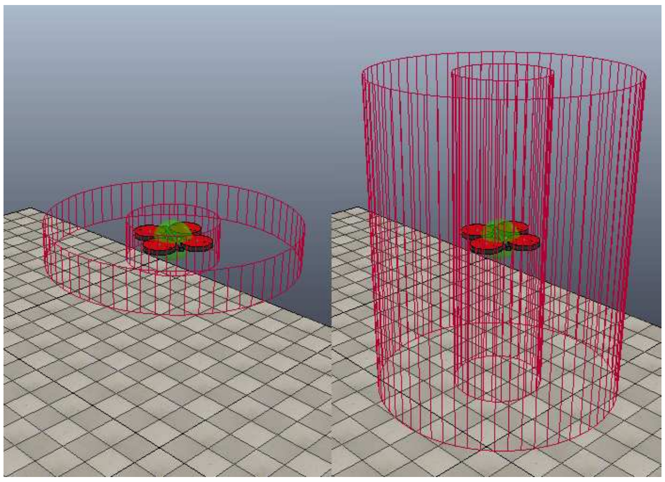

Figure 5.4: Comparison of the sizes of obstacle detection sensors for uses in 2D (left) and 3D (right).

5.2 Expanding the algorithm to 3D

The extension of the algorithm into 3D is fairly straightforward. A spherical proximity sensor must be used for the detection of neighbors instead of a cylindrical one (the sphere is the equivalent of the circle in 3D). The code must then be adjusted to enable the set point to be set in directions with different altitudes. Figure 5.5 previews an example of a simulation of such a swarm.

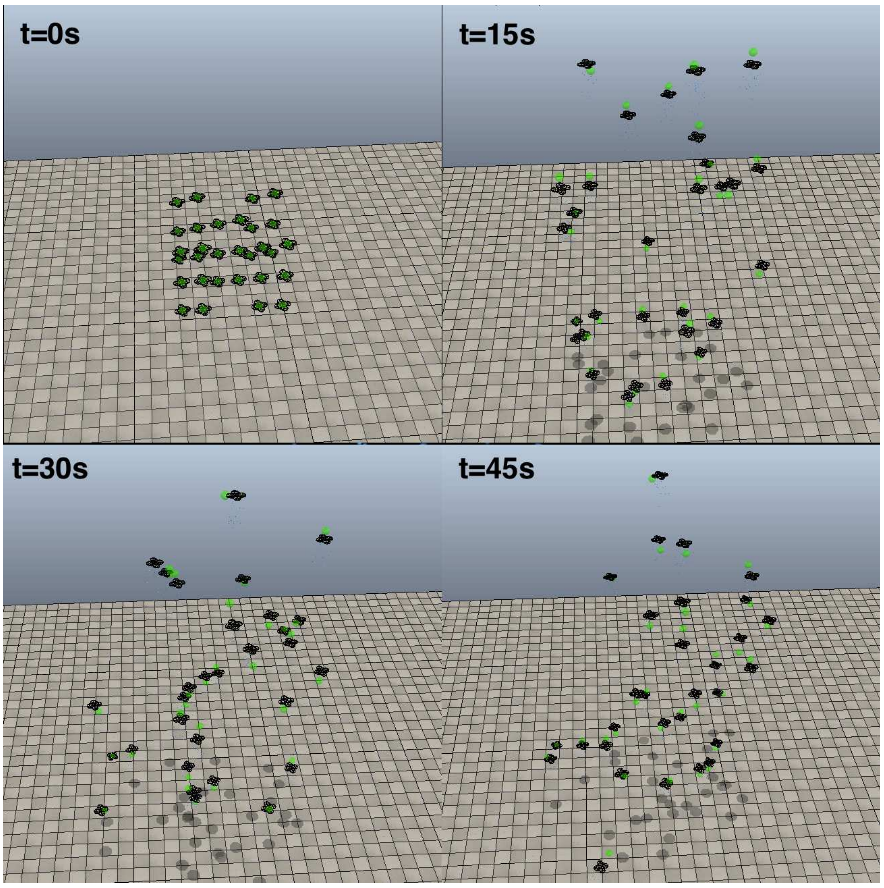

Figure 5.5: Swarm in 3D. Video available at `http://youtu.be/o0v5oe6ekVY`.

The position of the center of the swarm has been tracked on Figure 5.6. The avoidance manoeuvre performed by the MAVs to avoid collision is more aggressive in the vertical direction because the MAVs must keep a larger margin of security along the Z axis (due to the air flow interference discussed above). This causes the altitude of the swarm to oscillate more than the positions along the two horizontal axis.

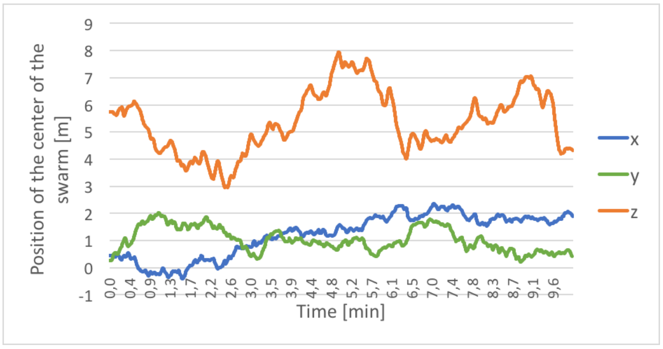

Figure 5.6: Position of the swarm in space.

5.3 Issues of the 3D algorithm

The 3D algorithm might have been fairly easy to implement once we have the 2D one, but for the following reasons we used the 2D model in the previous parts of our work:

1. It is relatively easier to develop a new behaviour in 2D and then convert it into 3D (as we did above for the basic coherence behaviour) than to develop it directly in 3D.
2. The current control system of MAVs which is expected to be used in real deployment of the system (described in [2]) does not support very well this kind of random flying in 3D.
   - (a) On real MAVs, the air flow generated by propellers is much more significant than it is in V-REP simulations. In a system where the emphasis is given to random movements, an accident can very soon happen as MAVs fly at different altitude levels.
   - (b) The vehicle controller uses optical flow to measure the relative velocity of the ground below with a down facing camera. If this camera is occluded by another MAV, it can disturb flight stability.

Chapter 6

Conclusion

The Minimalist Coherent Swarming algorithm presented by Nembrini et al [4] has been successfully implemented and extended for using with MAVs. After introducing the algorithm itself, the implementation process was described step by step, as well as the simulation environment we worked in, V-REP. The behaviour of the swarm was analysed by changing parameters of the algorithm, or by introducing restraints such as noisy sensor readings or limited angles of view. A target following mechanism was designed, implemented and verified, enabling the swarm to move in a environment with obstacles into a given location and requiring only minimal sensorial and computational resources. Finally, the swarming algorithm was expended into 3D, respecting the constrains of the MAV.

Finally let us conclude that all the mandatory points of the thesis assignment were fulfilled. After consultation with the thesis supervisor, it has been decided not to include in this work the implementation of this algorithm on real MAVs, which will be the topic of a subsequent common research. The overall system was verified in numerous simulations in realistic robotic simulator V-REP. The swarming approach was adapted and integrated into the system being developed by the Multi-robot Systems group for relative stabilization of MAVs, which enables its experimental deployment in real world scenarios in a similar way as it was done for testing swarm behaviour in [13, 8, 12] and stabilization of MAV formations in [9, 14, 10, 11].

Bibliography

[1] Gerardo Beni. From swarm intelligence to swarm robotics. In Erol Şahin and WilliamM. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9. Springer Berlin Heidelberg, 2005.

[2] Tomáš Báča. Model predictive control of micro aerial vehicle using onboard microcontroller. Master's thesis, Czech technical university in Prague, 2015.

[3] Yongnan Jia and Long Wang. Experimental implementation of distributed flocking algorithm for multiple robotic fish. *Control Engineering Practice*, 30:1 – 11, 2014.

[4] Julien Nembrini, Alan Winfield, and Chris Melhuish. Minimalist coherent swarming of wireless networked autonomous mobile robots. In *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior on From Animals to Animats*, ICSAB, pages 373–382, Cambridge, MA, USA, 2002. MIT Press.

[5] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987.

[6] R.Andrew Russell. Heat trails as short-lived navigational markers for mobile robots. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 3534–3539 vol.4, Apr 1997.

[7] Young-Sun Ryuh, Gi-Hun Yang, Jindong Liu, and Huosheng Hu. A school of robotic fish for mariculture monitoring in the sea coast. *Journal of Bionic Engineering*, 12(1):37 – 46, 2015.

[8] M. Saska, J. Chudoba, L. Preucil, J. Thomas, G. Loianno, A. Tresnak, V. Vonasek, and V. Kumar. Autonomous Deployment of Swarms of Micro-Aerial Vehicles in Cooperative Surveillance. In *Proceedings of 2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, volume 1, pages 584–595, Danvers, 2014. IEEE Computer society.

[9] M. Saska, Z. Kasl, and L. Preucil. Motion Planning and Control of Formations of Micro Aerial Vehicles. In *Proceedings of The 19th World Congress of the International Federation of Automatic Control*, pages 1228–1233, Pretoria, 2014. IFAC.

[10] M. Saska, T. Krajnik, V. Vonasek, Z. Kasl, V. Spurny, and L. Preucil. Fault-Tolerant Formation Driving Mechanism Designed for Heterogeneous MAVs-UGVs Groups. *Journal of Intelligent and Robotic Systems*, 73(1-4):603–622, January 2014.

[11] M. Saska, T. Krajnik, V. Vonasek, P. Vanek, and L. Preucil. Navigation, Localization and Stabilization of Formations of Unmanned Aerial and Ground Vehicles. In *Proceedings of 2013 International Conference on Unmanned Aircraft Systems*, pages 831–840, New York, 2013. Springer.

[12] M. Saska, J. Langr, and L. Preucil. Plume Tracking by a Self-stabilized Group of Micro Aerial Vehicles. In *Modelling and Simulation for Autonomous Systems*, volume 1, pages 44–55, Cham, 2014. Springer.

[13] M. Saska, J. Vakula, and L. Preucil. Swarms of Micro Aerial Vehicles Stabilized Under a Visual Relative Localization. In *ICRA2014: Proceedings of 2014 IEEE International Conference on Robotics and Automation*, pages 3570–3575, Piscataway, 2014. IEEE.

[14] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil. Coordination and Navigation of Heterogeneous MAV-UGV Formations Localized by a hawk-eye-like Approach Under a Model Predictive Control Scheme. *International Journal of Robotics Research*, 33(10):1393–1412, September 2014.

List of Figures

Appendices

Appendix A: Content of the enclosed CD

- `/`
  - `bachelor_thesis.pdf` ... This thesis in PDF format
  - `Scenes` ... V-REP scenes used in this thesis
    - `readme.txt` ... Instruction on how to open scene files
    - `3D.ttt` ... Scene with Alpha algorithm expended into 3D (5.2)
    - `AlphaAlgorithm.ttt` ... Scene with the Alpha algorithm (2.3.4)
    - `EnvironmentWithObstacles.ttt` ... Scene with the target following algorithm in and environment with obstacles (4.4)
    - `ReducedAngle.ttt` ... Scene with the Alpha algorithm with a reduced sensor angle (3.5)
    - `TargetFollowing.ttt` Scene with the target following algorithm (3.5)
  - `Videos`
    - `3D.avi` ... Simulation of the expention of the Alpha algorithm into 3D (5.2)
    - `AlphaAlgorithm-10.avi` ... Simulation of the Alpha algorithm for 10 MAVs (2.3.4)
    - `AlphaAlgorithm-20.avi` ... Simulation of the Alpha algorithm for 20 MAVs (2.3.4)
    - `AltitudeTest.avi` . Simulation of the effect of air flow interference on the MAVs (5.1)
    - `AvoidanceTest.avi` ... Simulation of the testing of the collision avoidance algorithm (2.3.2)
    - `BasicAlgorithm.avi` ... Simulation of the basic coherence algorithm (2.3.3)
    - `EnvironmentWithObstacles-FastVersion.wmv` ... Simulation of the target following algorithm in and environment with obstacles (4.4) - 10x speed up
    - `EnvironmentWithObstacles.avi` ... Simulation of the target following algorithm in and environment with obstacles (4.4)
    - `ReducedAngle.avi` . Simulation of the Alpha algorithm with a reduced sensor angle (3.5)
    - `TargetFollowing-FastVersion.wmv` ... Simulation of the target following algorithm (3.5) - 5x speed up
    - `TargetFollowing.avi` ... Simulation of the target following algorithm (3.5)