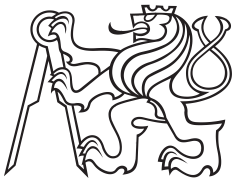
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Distributed Cohesive Control for Swarms of Micro Aerial Vehicles

Distribuované kohézní řízení autonomního roje helikoptér

Jan Charvát

Supervisor: Dr. ret. nat. Martin Saska
Supervisor–specialist: Tomáš Bača
Field of study: Cybernetics and Robotics
Subfield: Robotics
February 2017

Acknowledgements

I would like to thank my parents for their continued moral and financial support throughout my studies. Furthermore, I thank my supervisor Martin Saska for offering an interesting topic of investigation and his support. I would also like to thank Tomáš Bača, Vojtěch Vonásek and other people from IMR group for valuable advice.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date .....

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....

signature

Abstract

This thesis deals with distributed cohesive control of swarms of dimensionless particles and applicability of this approach for using with swarms of micro aerial vehicles. I have implemented a set of algorithms presented in [11] that leads to the cohesive swarm behavior of ground robots in a plane and extended these algorithms to 3D space for control of unmanned aerial vehicles. The behavior of original algorithms and their extensions was verified by simulations in V-REP and Gazebo robotic simulators.

Keywords: multi-robotic systems, swarm control, unmanned aerial vehicles

Supervisor: Dr. ret. nat. Martin Saska
Department of Cybernetics,
Karlovo náměstí,
Praha 2

Abstrakt

Tato práce se zabývá distribuovaným kohézním řízení rojů bezrozměrných částic a jeho použitelností pro roje bezpilotních letounů. Podle [11] jsem naimplementoval sadu algoritmů vedoucí ke kohéznímu chování rojů pozemních robotů v rovině a rozšířil tuto sadu algoritmů do 3D prostoru pro řízení bezpilotních letounů. Chování původních algoritmů a jejich rozšíření jsem ověřil simulacemi v robotických simulátorech V-REP a Gazebo.

Klíčová slova: multi robotické systémy, řízení rojů, bezpilotní letouny

Překlad názvu: Distribuované kohézní řízení autonomního roje helikoptér

Contents

Figures

Tables

Chapter 1

Introduction

Imagine a robotic swarm of autonomous mirco aerial vehicles (MAVs) in an exterior environment that needs to remain cohesive. The swarm is being influenced by external forces, which can easily stretch it until it disconnects. Robots are constrained by a limited range of communication and onboard sensors so that they have no knowledge of the swarm global topology. Furthermore, there is a demand to control global swarm motion effectively by controlling leaders through predefined trajectories or a human interface (i.e. joystick).

The unmanned aerial vehicles (UAVs) are nowadays very popular for their wide possibilities of usage and relatively low cost compared to manned aerial vehicles. Furthermore, these aerial robots are often able to handle same work as manned aerial vehicles in many areas (e.g. surveillance, dynamical sensor networks, film industry, photography, agriculture, package delivery or military) without risking human's life.

The UAV can be flown by a human operator or by an electronic system. The electronic control system might be either onboard or remote. The human control is usually preferred recently. In such a case the human control approach would require approximately the same number of "synchronized" people. In an ideal case, there should be a few persons flying leader UAVs and the remaining swarm members should be dynamically controlled by a distributed artificial intelligence algorithms leading to UAV's adaption to current environmental conditions and the determining moves of the leaders.

In Chapter 2, will be stated related literature, shown contribution of the work and where it goes beyond state-of-the-art. In Chapter 3, an adaptation of algorithms from [11] for using with UAV will be presented including integration of UAV model and low-level control and stabilization. Verification in V-REP robotic simulator [18], which enables the study of swarm behavior of a large number of individuals is presented in the same chapter. Extension into 3D space in two variants will be introduced in Chapter 4. Integration of designed method into the ROS system and subsequent verification in Gazebo simulator [1] is presented in Chapter 5.

Chapter 2

Related work

This work is based on a set of algorithms designed for ground robots presented in paper [11]. The authors present algorithms that achieve desired properties of swarm behavior such as connectivity, cohesiveness, scalability, and robustness. The authors also present a simulation with ground robots and the audited implementation of these algorithms for plane swarm control.

Swarm robotics is strongly inspired by collective intelligent behavior produced by swarms formed by simple and weak individuals [5]. The common source of inspiration for swarm robotics are social insects (e.g. ant, bees, wasps), but even wolfs or humans often use more sophisticated versions of distributed algorithms.

Distributed approach for swarm control is based on local control and communication observed in biological systems resulting in a collective intelligent behavior necessary for survival. Biological inspiration brought more interesting ideas like digital pheromone [16] as an analogy to pheromones, which are chemical substances used by ants and other similar insects to mark environment to inform other swarm members.

Iocchi et al. [8] has distinguished between distributed and centralized control approach. Centralized methods usually consist of an organization system having a leader that is in charge. Distributed approach is composed of robotic agents which are completely autonomous in the decision process with respect to each other so that in this class of systems a leader does not exist. By this definition, this thesis uses a combined approach.

Communication is necessary for autonomous agents to cooperate. In [14] other interesting research domains of swarm robotics are described areas often profiting from cooperation provided by communication: task allocation, object transportation and manipulation, mapping and localization or learning.

Motion coordination is important study field of the swarm robotics and includes the path planning problem, but only some papers consider path planning in a 3D space like [9]. Path planning algorithms operating in a 3D dynamical environment could be used for path planning for leaders employed to control swarm motion. Some research also focuses on the formation generation problem and papers can be divided by control approach:

central and distributed. The centralized approach requires control unit able to reach, supervise and command all swarm members, which is vulnerable because of possible central unit failure. The second group studies distributed algorithms for achieving a coordination. Varying distributed control strategies can be subdivided by the approach they use: behavior-based [3], potential field approach [22] and leader-follow approach [20]. Actually, the distributed control strategy combining potential field approach and leader-follow approach is used in the thesis. Compared to the centralized approach, distributed approach based on autonomous agents creating own decision with respect to others can be more interesting to observe and even more reliable, but the behavior of individuals can become easily unpredictable. Distributed approach is considered to be more reliable because of independent individuals resulting in partial node failure resistance of the whole swarm, but forming formations defined by desired shapes is hardly achievable with distributed approach because future positions of members are determined by onboard computation using data obtained from noisy sensors.

In our method we plan to use the system for relative localization presented in [25], which uses onboard visual relative localization of patterns (e.g. circular pattern) captured by onboard camera. At present, most of relative localization systems for UAV/multi-UGV uses visual localization based on pattern recognition like [28].

The basic distributed swarm motion coordination approach is based on Reynold's BOIDs model presented in [17]. BOIDs model is integrated into numerous systems for UAV motion control. In [23] BOIDs model is integrated for using with a system for visual relative localization for maintaining a stabilization of the swarm of UAVs.

In this work method presented in [15] based on potential fields will be used instead of Reynold's BOIDs model, which is based on heuristic rules. The basic attraction/repulsion controller will be used with distributed Leader follow algorithm from [11], which can balance the influence of multiple leaders based on distance sensitive weighting.

Chapter 3

Basic cohesive control algorithms

3.1 Introduction

In this chapter algorithms developed for control of unmanned ground vehicles (UGV) from [11] and other referred algorithms from [12], [13] and [15] are described. The designed system is based on proposed algorithms but uses them for control of UAV swarms.

3.2 Preliminaries

Let us assume a finite set of robots $\mathcal{R} = \{r_i \mid i = 1, 2, \ldots, m\}$, where $|\mathcal{R}| = m$ is number of robots forming a swarm. There is also subset $\mathcal{L} \subset \mathcal{R} = \{r_i \mid i = 1, 2, \ldots, l; l \ll m\}$ of leaders whose motion can be controlled remotely through a human interface or by a predefined trajectory. Each robot has unique ID and leader's IDs are easily known to the rest of the swarm. We denote position of robot r_i as $q_i \in \mathbb{R}^3$. Configuration of the swarm is defined by positions of all robots $q = col\,(q_1, q_2, \ldots, q_m)$. Neighbors of robot r_i are defined as $N_i = \{r_j \in \mathcal{R} \mid \|q_i - q_j\| < \Gamma; i \neq j;\}$, where $\Gamma > 0$ is interaction range, which is the maximal distance, in which the robots are able to communicate and relatively localize themselves.

We also consider the following dynamical model of robot r_i:

$$\dot{q}_i = p_i,$$

$$\dot{p}_i = u_i,$$

where p_i is velocity and u_i is acceleration.

Let us also assume graph associated with swarm configuration. The robots corresponds to vertices $\upsilon = 1, 2, \ldots, m$ of graph $G = (\upsilon, \epsilon)$. Set of edges between vertices of graph G consists of edges whose length is lower than interaction range r, i.e.: $\epsilon = \{(i, j) \mid i, j \in \upsilon; i \neq j; \|q_i - q_j\| < \Gamma\}$.

Pair (G, q) then defines overall swarm structure.

Let also define vector $r_{ij} = q_j - q_i$, which is considered to be a final output of the system for relative localization and the vector points to the neighbor r_j in the coordinate frame of r_i.

3.3 Method description

Not all presented algorithms are fully distributed, which means that global knowledge of the swarm topology or states of neighbors or other swarm members is required by some algorithms. Inputs of algorithms are mostly relative positions of neighbors thus this swarm control solution requires a system for relative localization of neighboring UAVs. For the final deployment of the system, the relative localization algorithm presented in [6] is considered. The only equipment demands placed on UAVs are these: system for relative localization and ability to communicate with neighbors in a predefined range. The communication is necessary because some of the algorithms need data, which can be obtained only by onboard processing of received messages and broadcast own computed data to the neighbors.

The presented set of distributed cohesive algorithms consists of three groups as divided in [11] containing one or more algorithms in each.

1. **Base behavior** provides a basic required swarm behavior that maintains a connected network. These algorithms should ensure forming of a regular lattice and boundary optimization. This strategy produces convex swarm boundary and should converge to a well-rounded shape.

    a. **Flocking algorithm I** presented in [15] uses a potential field to form regular α-lattice. Three flocking distributed algorithms are presented: *Algorithm I*, *Algorithm II*, *Algorithm III*, but only *Algorithm I* matches needs of this thesis.

    b. **Boundary detection** algorithm of [13] determines if the robot is on the boundary of the swarm. This information is necessary for the decision, which robot will be affected by the boundary tension force. The boundary detection also provides information if the robot lies on a convex boundary.

    c. **Boundary tension** algorithm of [12] straightens and minimizes the boundary of the swarm. The boundary tension force is applied only to boundary robots, which is determined by boundary detection algorithm. The boundary tension is simply performed by pushing in the middle of two adjacent boundary neighbors.

2. **Leader follow forces** improve the cohesiveness, which is vulnerable because of the following leader by remaining swarm members. Whereas the base behavior is not able to deal with a convex boundary leaders stretching the swarm, which fast leads to loosing the connectivity.

3. **Stability improvements** bring other advantages. These algorithms primarily affect the distribution of members leading to more homogeneous distribution and thickness, which is compressed similarly to the stockings. When using the Stability Improvement strategy (i.e. all presented algorithms) the swarm topology gives a generalization of a Euclidian Steiner tree [5]. Authors claim resulting swarm behavior performs close to the best-known approximation bound for a corresponding centralized static optimization problem.

   a. **Thickness contraction** algorithm works similarly to stockings. Contraction force grows linearly with thickness and is applied only on boundary robots.

   b. **Density algorithm** is based on the idea of attraction to a low-density neighborhood (area with the low density of neighboring robots) and repulsion from high-density neighborhood leading to maintaining the overall swarm's density at specific homogenous level.

3.3.1 Flocking algorithm

Flocking is a form of collective behavior of interacting agents that can be observed in nature (e.g. flock of birds or insect swarms).

Three heuristic rules that leads to flocking was introduced in [17]. These rules allowed first animations of flocking. The rules presented by Reynolds are

1. **Flock centering**: attraction to the center of a local flock;
2. **Collision avoidance**: repulsion from too close neighbors;
3. **Velocity matching**: adaption to the velocities of neighbors.

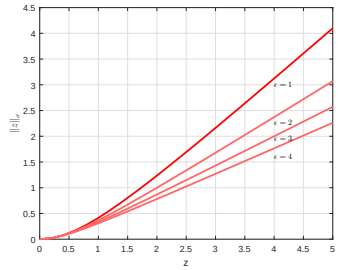
Reynolds rules are often called *cohesion*, *separation* and *alignment* in the literature. In [15] is demonstrated that *Flocking algorithm I* embodies Reynolds rules whose implementations can differ, because they are not mathematically described in [17].

The flocking algorithm is a main component of distributed swarm control. The algorithm is proven to converge and produces equilateral triangle grid called α-lattice or quasi-α-lattice with **configurable desired distance** $\gamma \in (0, \Gamma]$. In [15], γ is referred as *scale* and $\kappa = \gamma/\Gamma$ as *ratio*. It is recommended to set γ long enough, so that each robot has six neighbors, otherwise forming a clean grid would be impossible because of over skipping connections.

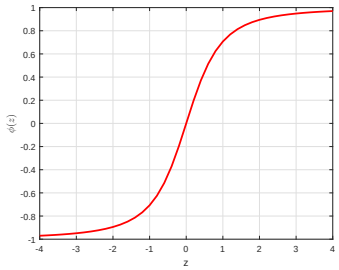
Algorithm I of Olfati-Saber:

$$\mathcal{R} \to \mathbb{R}^3, r \mapsto \underbrace{\sum_{n \in N(r)} \phi_\alpha \left(\|q_r(n)\|_\sigma \right) \mu \left(q_r(n) \right)}_{\text{Grid sum}} + \underbrace{\sum_{n \in N(r)} \nu \left(q_r(n) \right) p_r(n)}_{\text{Concensus sum}}$$

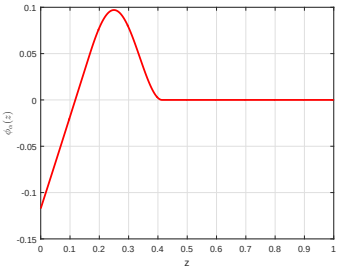
Unlike other algorithms from the thesis, output of *Algorithm I* is acceleration.



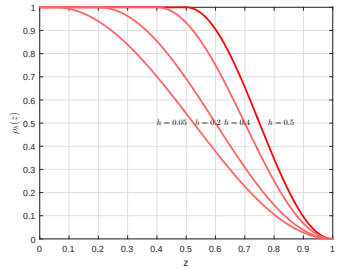
(a) : $\|z\|_\sigma$ for $z \in \mathbb{R}\epsilon \in 1, 2, 3, 4$



(b) : ϕ for $a = 1, b = 1, c = 0$



(c) : ϕ_α with $\Gamma = 1, \gamma = 0.5, \epsilon = 1$



(d) : ρ_h for $h \in 0.05, 0.02, 0.4, 0.5$

Figure 3.1: Plots for *Flocking Algorithm I*

The notation $\|\cdot\|_\sigma$ denotes special norm that is defined as

$$\|z\|_\sigma = \frac{1}{\epsilon}\left(\sqrt{1 + \epsilon \|z\|^2} - 1\right).$$

Norm has been defined because it is differentiable at $z = 0$, while $\|z\|$ is not. Adjustable constant $\epsilon > 0$ has been identified experimentally as 4.

The action function

$$\phi_\alpha(z) : \mathbb{R} \to \mathbb{R}, z \mapsto \rho\left(z / \|\Gamma\|_\sigma\right) \phi\left(z - \|\gamma\|\right)$$

introduces repulsion or attraction between two robots and deviation from the desired grid length.

$$\phi(z) : \mathbb{R} \to \mathbb{R}, z \mapsto \frac{1}{2}\left[(a+b)\frac{z+c}{\sqrt{1+(z+c)^2}} + (a-b)\right]$$

is a sigmoidal function used to smooth the attraction or repulsion force with tunable constants $a, b, c \in \mathbb{R}$. The strength of repulsion converges to $a > 0$ and strength of repulsion to $b > 0$ and $c = |a - b| / \sqrt{4ab}$. In thesis we set $a = 1$, $b = 1$ and $c = 0$.

Function

$$\rho_h(z) : \mathbb{R} \to \mathbb{R}, z \mapsto \begin{cases} 1, & z \in [0, h) \\ \frac{1}{2} * \left[1 + cos\left(\pi \frac{z-h}{1-h}\right)\right], & z \in [h, 1) \\ 0, & \text{otherwise} \end{cases}$$

is smooth between $\rho_h(0) = 1$ and $\rho_h(1) = 0$ with tunable constant $h \in (0, 1)$, which we set to 0.4 in this thesis.

Function

$$\mu : \mathbb{R}^3 \to \mathbb{R}^3, q \mapsto \frac{q}{\sqrt{1 + \epsilon \left\|q\right\|^2}}$$

determines the neighbor direction and the direction of the force to improve distance to the neighbor. Strength of the force is given by ϕ_α then.

Weighted adjacency function

$$\nu : \mathbb{R}^3 \to \mathbb{R}, q \mapsto \rho_h \left(\left\|q\right\|_\sigma / \left\|\Gamma\right\|_\sigma\right) \in [0, 1], i \neq j$$

is used for determining the influence of a neighbors based on its distance.

Despite the fact that algorithm produces a grid, *fragmentation phenomenon* linked to lack of cohesion in the swarm sometimes occurs. Note that flocking algorithm alone does not preserve **cohesiveness** necessarily. The formal definition of cohesiveness can be found in [15].

In paper [11] slightly modified version of *Algorithm I* has been introduced. Using consensus sum can produce a risky behavior, if motionless robot notices n neighbors moving in the same direction then accelerate n times. When using the averaged consensus instead of its sum, the following robot is never faster than robot it follows. In this thesis, we use both versions, because when moving in a smaller group with the single leader, averaged consensus could lead to losing connectivity because follower is never faster than a leader.

The modified version of Flocking algorithm I from [15] used in [11] is as follows:

$$\mathcal{R} \to \mathbb{R}^3, r \mapsto \underbrace{\sum_{n \in N(r)} \phi_\alpha \left(\left\|q_r\left(n\right)\right\|_\sigma\right) \mu\left(q_r\left(n\right)\right)}_{\text{Grid sum}} + \underbrace{\frac{1}{\left|N\left(r\right)\right|} \sum_{n \in N(r)} \nu\left(q_r\left(n\right)\right) p_r\left(n\right)}_{\text{Average concensus}}$$

In this thesis, we use both versions: average and sum consensus. In each Chapter is stated, which version has been used for simulations.

3.3.2 Boundary detection

Swarm boundary classification is fundamental for keeping a swarm cohesive and connected. We wish to classify boundaries as interior void or exterior boundary. Global boundary classification provides information for running additional algorithms filling interior holes or optimizing exterior boundary.

Cyclic shape algorithm presented in [13] is fully distributed and requires only local network geometry available to each robot as an output of the system for relative localization. The main component of the *cyclic shape* algorithm is a *find empty sectors* algorithm, which searches for empty sectors between adjacent neighbors.

Cyclic shape algorithm first sorts neighbors clockwise and then searches for empty sectors. The empty sector is found if and only if the angle between two adjacent neighbors is bigger than π or neighbors distance is not within the interaction range. If there has been at least one empty sector found robot is classified as the boundary, otherwise as non-boundary. If the single empty sector has been found and the angle is bigger than π, the robot is classified as convex boundary robot.

After the robot finished *local boundary classification* using the *cyclic shape* algorithm, *global boundary classification* can be performed by computing the exterior angle of boundary shape. An exterior angle of a polygon formed by a boundary robots will sum to 2π, while the interior will sum to -2π.

The cyclic shape algorithm is defined as follows:

$$CS(r) : \mathcal{R} \to \{0, 1\}, r \mapsto hasEmptySectors\,(r)\,.$$

Based on the *boundary detection algorithm* classification of each robot from $\mathcal{R}$ let's define subset $\mathcal{B} \subset \mathcal{R} = \{r_i \mid CS(r_i) = 1\}$ formed by boundary robots.

3.3.3 Boundary tension

Boundary tension algorithm presented in [12] optimizes the swarm boundary in terms of length and shapes and removes concave boundary regions. Authors also present *local articulation point identification algorithm* and *clustering algorithm*, which moves robots to the region, where local articulation point (LAP) was discovered, but this does not match needs of the thesis. As the LAP is classified a robot, whose removal (e.g. failure) disconnects the swarm in a local scope. In global scope removal of the LAP increases the routing distance between robots connected through the LAP or even disconnects the swarm globally. In a plane, robot r_i can be marked as the LAP, if the number of empty sectors is bigger than one, but this fails in a 3D space. To implement LAP detection invariant of space dimension, a different approach based on searching local communication graph by DFS or BFS had to be used.

In order to compute boundary force, a robot has to determine local boundary subgraph, which is done by running *boundary detection* algorithm and broadcasting its results to the neighbors. Then robot r_i has to generate vectors to its adjacent neighbors r_j and r_k using the system for relative localization. Then the boundary force vector can be computed as:

$$\mathcal{R} \to \mathbb{R}^2, r \mapsto w_b \left(\frac{r_{ij}}{\|r_{ij}\|} + \frac{r_{ik}}{\|r_{ik}\|} \right),$$

where w_b is weight used to adjust boundary tension force strength.

Note that applying boundary tension force on boundary robots also affects the overall swarm shape. With growing weight w_b, the density inside the swarm will get higher.

3.3.4 Leader follow algorithm

The algorithm is based on a smooth transition between two basic approaches for following the leader: matching leader's velocity and moving towards it. The combination of these two approaches is inspired by human (or more generally animal) behavior. Moving towards the leader is more efficient when the leader is far from the follower. Matching the leader's velocity is more suitable when the swarm member is closer to the leader.

Following the single leader or multiple leaders is the simplest and very effective control method, which allows one or only a few people to control the swarm of an arbitrary size.

We define pred(r) as a robot's predecessor in a minimum hop tree to the leader. In order to compute the leader forces, we need each swarm member to know three public variables for each leader: **leader velocity**, **leader direction** and **leader distance**, which is a minimum hop count to it. Leader velocity is one of the pred(r) in minimum hop tree to the leader. Leader direction is a normalized vector which is merged as follows: if pred(r) is a leader than the normalized direction to the leader is used otherwise each robot takes the direction of pred(r) and merges it with the normalized vector to pred(r) provided by the system for relative localization.

Swarm's global knowledge of these variables is done by periodical broadcasting messages in the limited group of neighbors. These messages are processed onboard by each UAV to compute these variables. Each robot needs to construct a minimum hop trees to all leaders by onboard processing, which is not trivial.

Leader force for robot r is computed as:

$$\mathcal{R} \to \mathbb{R}^2, r \mapsto w_l \sum_{l \in L} c_l\left(r\right) \frac{d_l\left(r\right)^{-1}}{\sum_{l' \in L} d_{l'}\left(r\right)^{-1}},$$

where w_l is a *leader follow force* weight, $d_l(r) : \mathcal{R} \to \mathbb{N}$ is a leader distance and $c_l(r) : \mathcal{R} \to \mathbb{R}^2$ is the force of leader l on robot r, which is computed by scaling leader direction to the length of the leader velocity.

3.3.5 Thickness contraction

The local thickness of robot r is defined as the radius of the largest hop circle containing the robot.

Thickness contraction force computation requires evaluation of three public variables for each robot: **thickness** $t(r)$, **boundary hop distance** $b(r)$ and **circle center distance** $h(r)$.

To avoid over skipping connections, which may distort relationship between geometric thickness and boundary hop distance, reduced neighborhood N_i' of r_i formed by neighbors $r_j \in N_i$ if edge between r_i a r_j meets the condition of the Gabriel graph. Considering a reduced neighborhood, where no other robot is allowed to be closer to the midpoint of the edge than robots connected by the edge, results in Gabriel Unit Disk communication graph (see [7] for formal definition).

Based on classification provided by the *boundary detection* algorithm we can compute the boundary hop distance as follows:

$$b(r_i) = \begin{cases} 0 & r_i \text{ on boundary} \\ min\left\{b(r_j) + 1 | r_j \in N_i'\right\} & \text{else} \end{cases}.$$

Then heuristic evaluation of thickness and circle center hop distance can be done as:

$$t(r_i) := max\left\{\left\{b(r_j)\right\} \cup \left\{t(r_j) | r_j \in N_i' \wedge t(r_j) + \lambda \geq h(r_j)\right\}\right\},$$

$$h(r_i) := min\left\{h(r_j) + 1 | r_j \in N_i' \wedge t(r_i) = t(r_j)\right\},$$

where $\lambda \in \mathbb{N}$ is a small constant. In thesis we use $\lambda = 2$.

The *thickness contraction force* grows linearly with thickness. Direction of the thickness contraction force is determined by position of the neighbor with the lowest center hop distance. Then the thickness force can be computed as:

$$\mathcal{R} \to \mathbb{R}^2, r_i \mapsto w_t t(r_i) r_{ij},$$

where w_t is a weight used to adjust thickness force strength, $t(r_i)$ is a thickness at robot r_i and r_{ij} is a vector pointing to the neighbor r_j, which is the neighbor of r_i with minimum circle center distance.

3.3.6 Density algorithm

Local density of robot r is defined as the number of visible neighbors divided by robot's observable area. The idea of this algorithm is based on attraction to low-density neighbors and repulsion from high-density neighbors leading to maintaining overall swarm's density at specific homogenous level. To improve the value because of problematic calculation of the observable area, the robot first calculates own density and then averages this origin value with origin values of the neighbors. Let denote this averaged density of robot r as $\rho(r)$.

Density force contribution for robot r_i is computed as:

Algorithm	Public variables
Flocking algorithm I	-
Local boundary detection	-
Global boundary detection	local boundary detection state
Boundary tension	-
Leader follow	leader direction, velocity and hop count
Density	optimal and averaged density
Thickness contraction	thickness,circle and boundary hop distance

Table 3.1: Algorithms used in thesis.

$$\mathcal{R} \to \mathbb{R}^2, r_i \mapsto w_d \sum_{r_j \in N_i} r_{ij}\phi(\rho(r_j) - \sigma),$$

where w_d is a density force weight used for adjusting its strength, $\phi(x) = x^3/\|x\|$, and r_{ij} is the direction from robot r_i to neighbor r_j.

Now when all algorithms have been presented lets define set of weights $\mathcal{W} = \{w_f, w_b, w_l, w_t, w_d\}$.

3.4 Implementation

V-REP robotic simulator was chosen for initial simulation implementation. V-REP provides a remote API available for many programming languages including Java, which has been chosen for the implementation. The parent project is maven project which consists of two nested maven projects:

1. **vrep** project built from files by Coppelia Robotics. The project contains custom Java data types and remote API files. This project is used by the second nested project to communicate with the simulator.
2. **QuadriDCA** is the main project formed by several packages, that includes algorithms.

3.4.1 Experiment controller application

I wrote the Experiment controller application, which allows a user to control experiments through GUI dynamically. The application provides activation/deactivation of each single algorithm, data logging, switching between dimension mode, change of weights of algorithms, a number of leaders, generate paths for leaders and much more.

List of relevant implemented commands available in the application (for full list type 'help' in the command line):

- run/stop simulation,

- load/save of experiment configuration files,
- take-off and land command,
- auto failure with defined failure rate per sec,
- change leader count,
- setting interaction range and desired distance,
- generating paths for leaders.

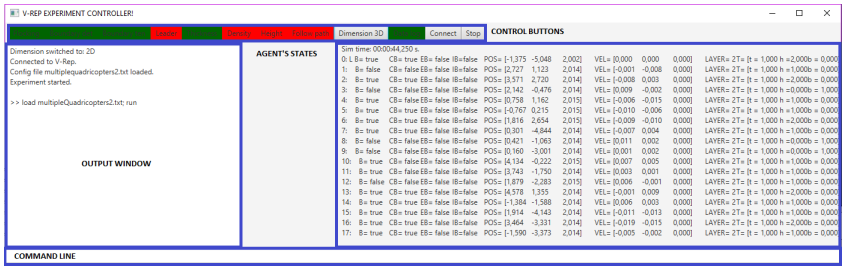


Figure 3.2: Screenshot of Experiment controller GUI with labeled elements

1. **Control buttons** allow a user to activate or deactivate each algorithm or switch between 2D and 3D modes. Data logging and export can be turned on or off as well. Connect and stop buttons can be used to start or stop the experiment.

2. **Agent's states** like position, velocity, boundary classification, thickness, circle center hop distance, boundary hop distance, averaged, origin and optimal density are displayed during the simulation.

3. **Command line** allows user dynamically change experiment parameters like *desired distance*, *communication range*, *number of leaders*, *algorithm parameters* and more.

4. **Output window** displays outputs of command interpreter after user commands are evaluated.

Button Switch 2D/3D automatically switches between 2D version and extended 3D versions of algorithms forming the **leader follow** strategy (i.e. flocking algorithm I, boundary detection, boundary tension, and leader follow algorithm) and LAP detection algorithm.

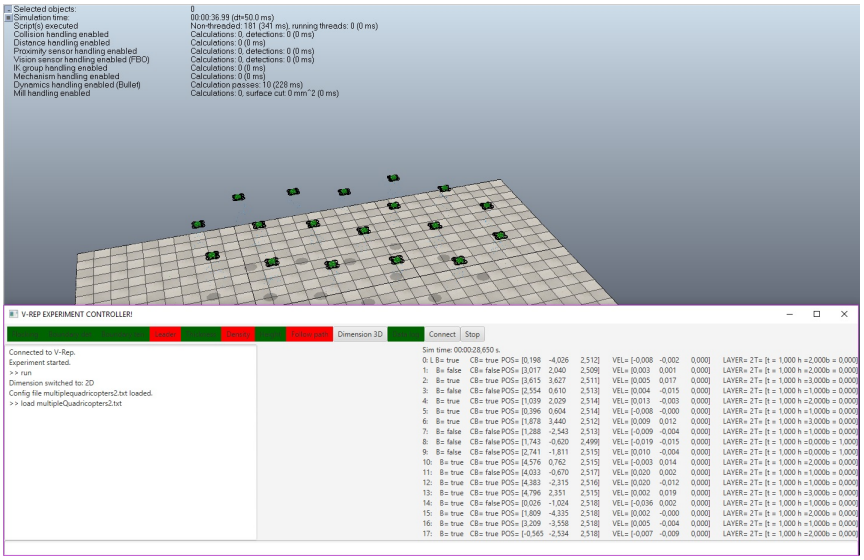


Figure 3.3: Screenshot of Experiment controller GUI and V-REP simulator during a simulation.

The source code of the application is available at *https://github.com/charvja2/Project1950* including Matlab scripts and functions used for plotting graphs from data exported from the application.

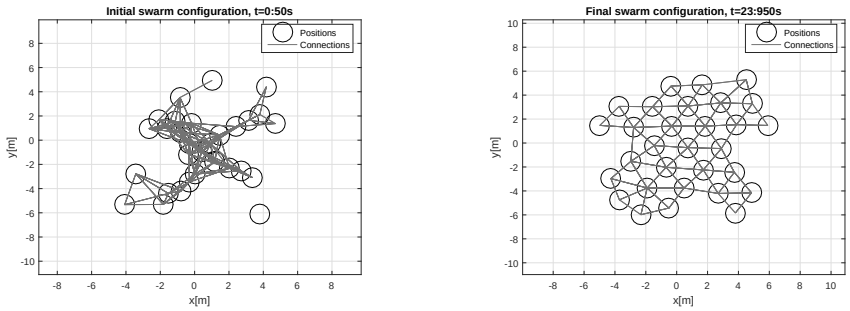
3.5 Verification

There were performed simulations containing many different scenarios. These tested scenarios were created to verify the behavior of the appropriate algorithm groups.

Experiment parameters used for all simulation look as follows: Robot communication range and range of relative localization system were set to 3 meters. Robot desired distance was set to 2 meters. Robot's maximal speed was limited by 0.5 ms^{-1}. Leader's maximal speed was limited by the range 0.25 – 0.3 ms^{-1}. The scenarios were tested for various combinations of numbers of leaders and robots.

3.5.1 Flocking verification

To verify the implementation of *Flocking algorithm I*, a simulation with 30 UAVs placed at random initial positions was performed. The simulation proved the ability of the algorithm to form almost equilateral α-lattice. Initial random positions and final positions of all robots are captured in 3.4.



(a) : Swarm initial configuration **(b) :** Swarm final configuration

Figure 3.4: Initial and final swarm configuration from the simulation. The final configuration is an example of quasi-α-lattice formed by *Flocking Algorithm I*.

3.5.2 Boundary detection

To verify the implementation of *Boundary detection*, *Boundary tension* and *LAP identification* algorithms, a simulation with 34 UAVs was performed. Video of the simulation is placed at https://youtu.be/qxsRAjdC-Ho. UAVs were placed in initial configuration containing LAPs, because during the forming of a α-lattice (see 3.4) LAPs quickly disappears.

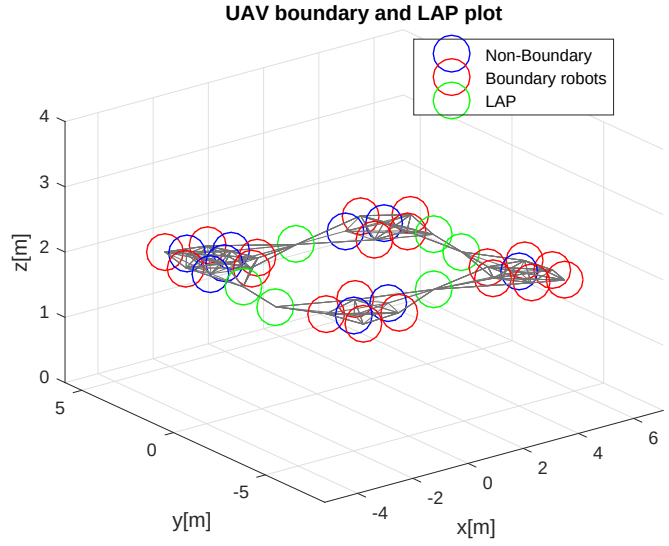


Figure 3.5: Swarm configuration from simulation verifying boundary detection algorithm and LAP classification in a plane based on a number of empty sectors.

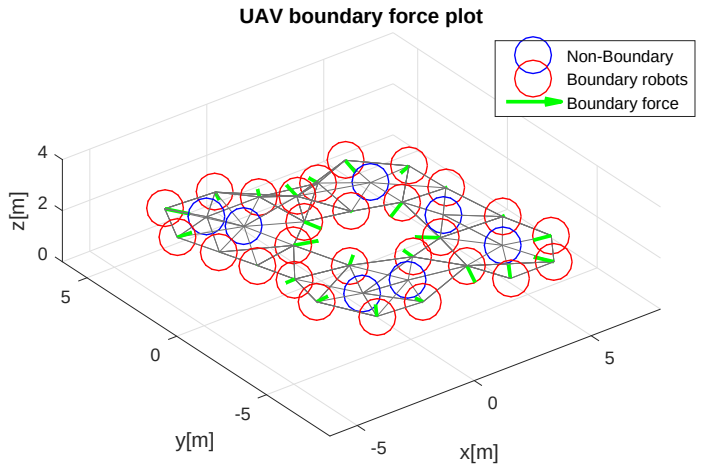


Figure 3.6: Swarm configuration from the same simulation as in 3.5 shows boundary tension force applied on boundary robots. Figure demonstrates principle of the boundary tension algorithm. In case of concave boundary robots, the direction of the force points outside the swarm. For convex robots, the direction points inside the swarm.

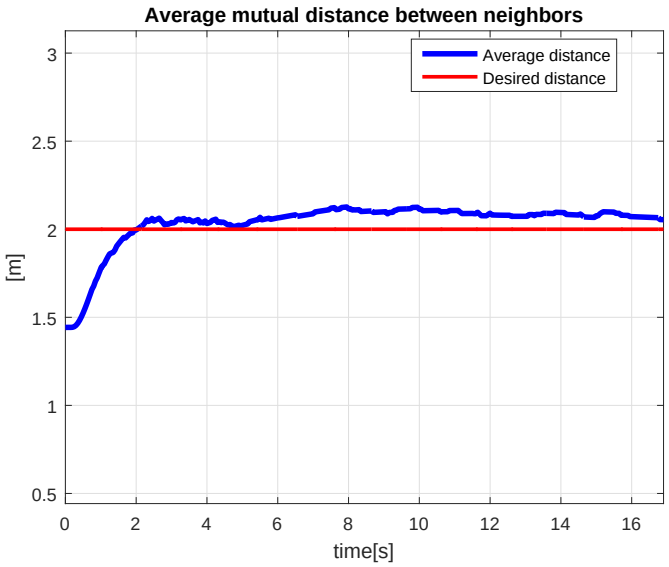


Figure 3.7: Averaged distance between neighbors and desired distance from the simulation captured in 3.5 and 3.6

Figure 3.7 shows that average distance between neighbors reached desired distance after 2 seconds.

3.5.3 Base swarm strategy

To verify behavior produced by base swarm strategy described in Section 3.3 following simulation was performed. UAVs were placed at very close initial positions to see if it is not a problem. Video of the simulation is placed at https://youtu.be/EaWENS-DW2w.

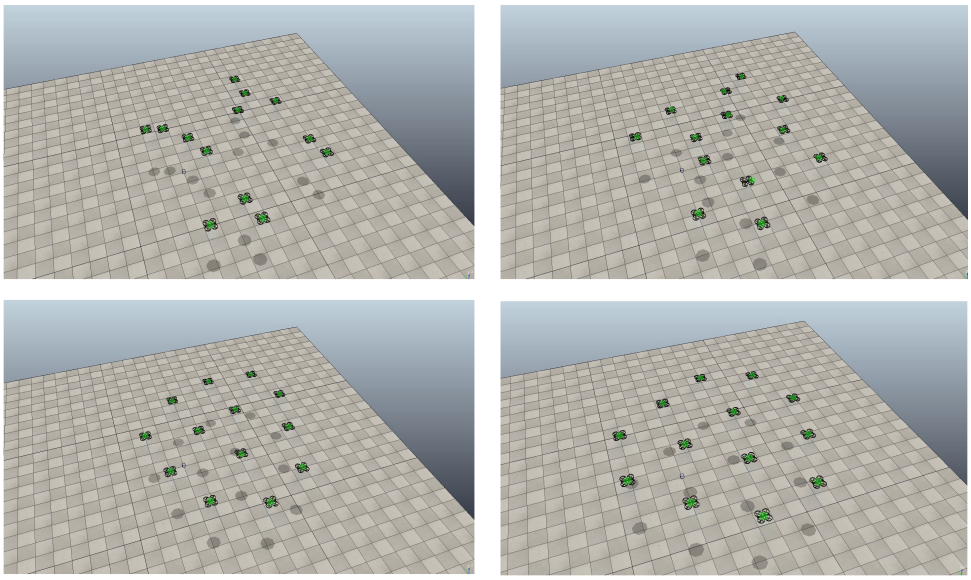


Figure 3.8: Snapshots from the simulation shows forming of a well-rounded formation using base swarm strategy

Base swarm strategy proofed ability to form well-rounded formation as can be seen in Figure 3.8 in less than 20 seconds.

3.5.4 Leader follow strategy

In this subsection, simulations including multiple leaders were performed. *Leader follow* algorithm has a significant impact on the swarm's topology and global behavior. Following single or multiple leaders is a very efficient but inaccurate way of global swarm motion control. Note that inaccuracy of this approach is caused by partial ability to influence the motion of the swarm members through the leader.

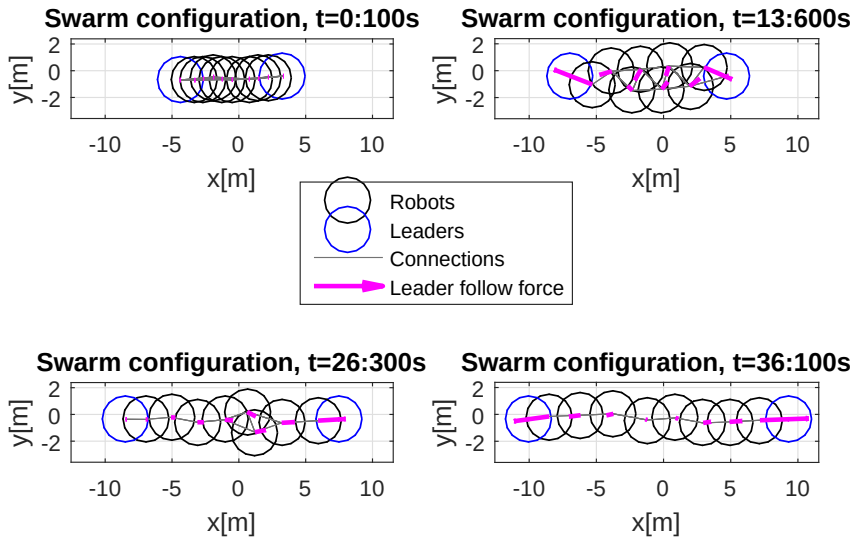


Figure 3.9: Graph shows principle of one dimensional case of leader stretching. Leaders move in opposite direction stretching the swarm until it disconnects. This approach is suitable for forming line formations. See the simulation at https://youtu.be/ybhOZF6EVO0.

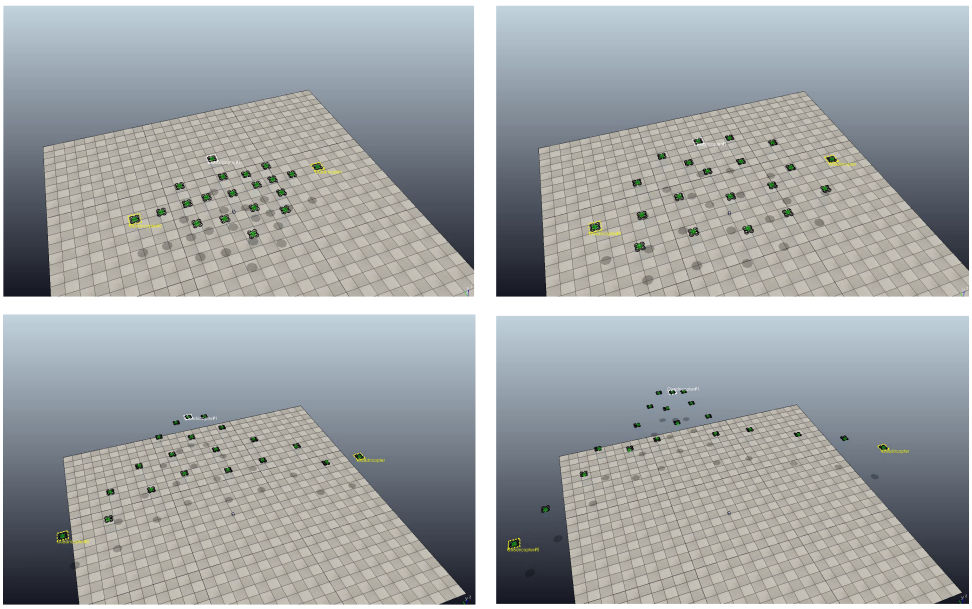


Figure 3.10: Snapshots from the simulation with 3 leaders moving in different directions until the swarm disconnects. Swarm members are balancing influence of 3 leaders based on hop count to each leader. See the simulation at https://youtu.be/5sDJH9QofyI.

3.5.5 Stability improvements

To verify the behavior of the algorithms forming *Stability improvements* strategy, two simulations were performed. Four leaders moving in a different directions scenario were tested with Leader follow strategy and then with Stability improvement strategy to see the difference.

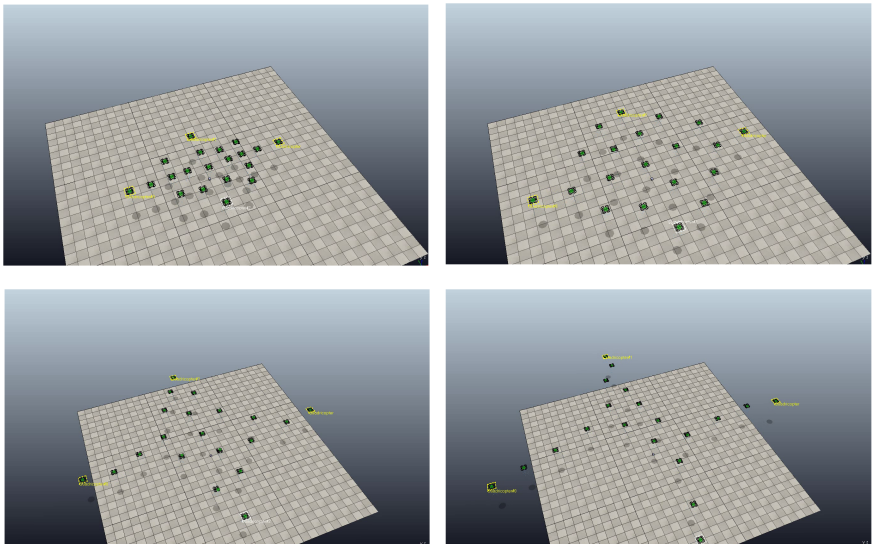


Figure 3.11: Snapshots from the video capturing the simulation available at https://youtu.be/WTMQzldlIRU using Stability improvement strategy

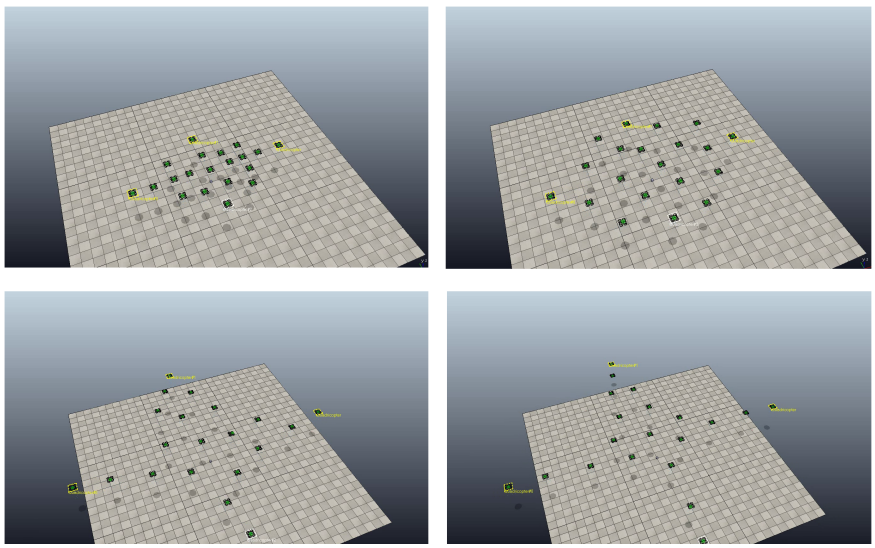


Figure 3.12: Snapshots from the video capturing the simulation available at https://youtu.be/86aunlMCZvw using Leader follow strategy

The difference between tested strategies can be seen by comparing the last pictures of Figures 3.11 and 3.12. When using Leader follow strategy

swarm formed hop circle (see right bottom corner of Figure 3.12). Stability improvement strategy compress thickness leading to better distribution along the edges between Steiner points. Compared to Leader follow strategy, Stability improvements strategy gives better performance in terms of duration before the swarm disconnects but applies only on swarms containing at least one hop circle.

3.6 Conclusion

In Section 3.3, algorithms originally designed for ground robots were analyzed for their possible usage with UAVs. Simulations to verify the implementation of the swarm controller were performed and presented in Section 3.5. During these simulations, I noticed that using a different configuration of $\mathcal{W}$ very diverse collective behavior can be achieved: from forming lattices, consensus flight, optimizing boundary, leader following or preserving homogenous distribution inside the swarm.

Chapter 4

Extension of the original algorithm for UAVs flying in 3D environment

4.1 Introduction

The goal of this chapter is to extend the referred methods into the 3D space for general using with micro aerial vehicles (MAVs). Two variants of the extension will be realized:

1. a general approach based on [15] for forming ball-shaped swarms using limited range of communication and onboard sensors,

2. an approach for MAV-swarm spreading driven by particular multi-robot applications (such as surveillance, active RFID localization, etc.) in the environment with different elevation.

4.2 Ball-shaped swarming

If we consider that the Base swarm behavior converges after some time to the shape similar to a water droplet, it's easy to imagine that extension of algorithms in the base swarm group will converge to a ball shape. Base swarm behavior group consists of three algorithms: Flocking algorithm, Boundary detection, and Boundary tension algorithm.

4.2.1 Flocking algorithm

There is no need for flocking algorithm to be extended. In a plane, Flocking algorithm I produces α-lattice or more often quasi α-lattice if we consider the presence of exterior forces. In a three-dimensional space, Flocking algorithm I produces crystal structures or quasi-crystal structures.

4.2.2 Boundary detection

According to our best conviction, there is no distributed three-dimensional boundary detection algorithm, so we had to design a new method. Designed boundary classification algorithm is inspired by the cyclic shape algorithm and the ray casting method often used for determining if a point lies inside a polygon. The point lies inside a polygon if a number of intersections between a ray and polygon's line segments is odd.

Each tested robot r_t first construct set of unique triples T_t formed by its mutually visible neighbors. Triplet $(r_i, r_j, r_k) \in T \Leftrightarrow \|r_{ij}\|, \|r_{ik}\|, \|r_{jk}\| < \Gamma \wedge i \neq j \neq i$. A set of triplets of neighbors forming triangles is mapped to the set of planes P in a following way: $r_{ti} + u(r_{tj} - r_{ri}) + v(r_{tk} - r_{ti})$, where $u, v \in (0, 1)$.

Let's define a half line equation $l = t\vec{v}$, where $t \in (0, \infty), \vec{v} \in \mathbb{R}^3$. Boundary detection algorithm searches for a ray that does not intersect any triangle defined by plane $p \in P$. If a ray that does not intersect any of triangles formed by neighbors triplets is found, the robot is classified as the boundary.

Boundary detection algorithm in 3D can be then defined as:

$$\mathcal{R} \to \{0, 1\}, r \mapsto \begin{cases} 1, & \text{otherwise} \\ 0, & \exists l \exists p \in P, l \cap p \neq 0 \end{cases}$$

4.2.3 Boundary tension

In a plane, two unit vectors pointing to the adjacent boundary neighbors are added and then multiplied by a parameter adjusting force magnitude. In 3D case, all vectors pointing to the boundary neighbors are added and normalized so that product is a unit vector.

Let's define boundary neighbors of robot r_i as $N_i^b = \{\mathcal{B} \cap N_i\}$. Boundary tension algorithm in 3D can be then defined as:

$$\mathcal{R} \to \mathbb{R}^3, r_i \mapsto w_b \frac{\sum\limits_{n_j \in N_i^b} r_{ij}}{\left\| \sum\limits_{n_j \in N_i^b} r_{ij} \right\|}$$

4.2.4 LAP detection

As mentioned in Subsection 3.3.3, LAP detection had to be extended for the 3D case. I have implemented distributed LAP detection using breath-first search (BFS) for the 3D case, where particles may operate.

General LAP detection algorithm searches for all LAPs in the graph. For our purpose, we need each robot to determine only its own LAP detection state,

so that the method could be simplified to be more suitable for distributed evaluation. Improved LAP detection invariant of dimension for robot r_i first selects random neighbor $r_j \in N_i$ as root of the tree. Then the local communication graph without edges connected to tested robot r_i is searched using BFS. After graph is constructed, the number of its nodes k is determined. The robot is classified as LAP if the number of its neighbors $|N_i|$ is not equal to the count of nodes forming a graph.

LAP detection algorithm is then defines as:

$$\mathcal{R} \to \{0,1\}, r_i \mapsto \begin{cases} 1, & |N_i| \neq k \\ 0, & \text{otherwise} \end{cases}$$

where k is a number of nodes in constructed tree.

4.2.5 Leader follow

There is no need to extend leader follow algorithm to the 3D in order to form a ball shape, but extension is beneficial and requires only formal change of definition of a leader follow force and leader direction from $\mathbb{R}^2$ to $\mathbb{R}^3$:

$$\mathcal{R} \to \mathbb{R}^3, r \mapsto w_l \sum_{l \in L} c_l(r) \frac{d_l(r)^{-1}}{\sum_{l' \in L} d_{l'}(r)^{-1}},$$

where w_l is a leader follow force weight, $d_l(r) : \mathcal{R} \to \mathbb{N}$ is a leader's distance and $c_l(r) : \mathcal{R} \to \mathbb{R}^3$ is the force of leader l on robot r.

Construction of a minimum hop tree to each leader is realized in the same way as in Subsection 3.3.4, i.e by broadcasting messages to the neighbors and their onboard processing.

4.2.6 Density algorithm

This subsection discusses an optional extension of density algorithm presented in Subsection 3.3.6. Similarly to the Leader follow algorithm, Density algorithm formula requires only the formal change of definition. Instead of projection from $\mathcal{R}$ to $\mathbb{R}^2$, mapping from $\mathcal{R}$ to $\mathbb{R}^3$ has to be used.

But we have to pay attention for computation of public variables required to compute the density force. Analytical calculation of observable area in 3D is very complicated, because of excluding exterior and interior areas. This performance demanding on board computation would probably gain only small advantage. From this reason, we leave this extension for future work.

4.3 Swarm spreading in the environment with different elevation

In this section, an approach for UAV-swarm spreading in an environment with different elevation is analyzed. This kind of spreading promises various real world applications like surveillance, border patrol or systematical search for survivors that are solved within our group in [25], [21], and [19].

In Chapter 3 has been verified that implementation of all swarming algorithms matches expected behavior so that this algorithm will be included for flying in a fixed height above the ground resulting in a swarm following the terrain. The major problem with swarm spreading in a terrain occurs, when UAV tries to fly over a slope with a high elevation. The maximal value of terrain elevation UAV is able to handle can be slightly improved by tuning maximal vertical speed constant defined in low-level MPC regulator [2], but there is still a limitation.

To particularly handle all slopes including the vertical one, the system would require an additional sensor integration (e.g. TeraRanger sensors inclined in different directions), which is the reason we leave this improvement for a future work.

4.3.1 Safe height

During the simulations in a 3D mode using Flocking algorithm I realized that algorithms inducing velocity growth can move UAV on a collision course with a ground surface. This is the reason I introduced parameter **safe height**. The periodically computed desired position passed as a reference to the UAV's MPC regulator is never allowed has the height lower than safe height.

4.4 Experimental results

4.4.1 Boundary detection 3D verification

In this subsection, two simulations to verify the behavior of *Boundary detection* algorithm in 3D with 11 and 44 UAVs were performed. Configuration from the simulation with 11 UAVs shows the principle of the algorithm. The single non-boundary robot inside the crystal structure cannot find half line, which does not intersect any of triangle between neighbors.

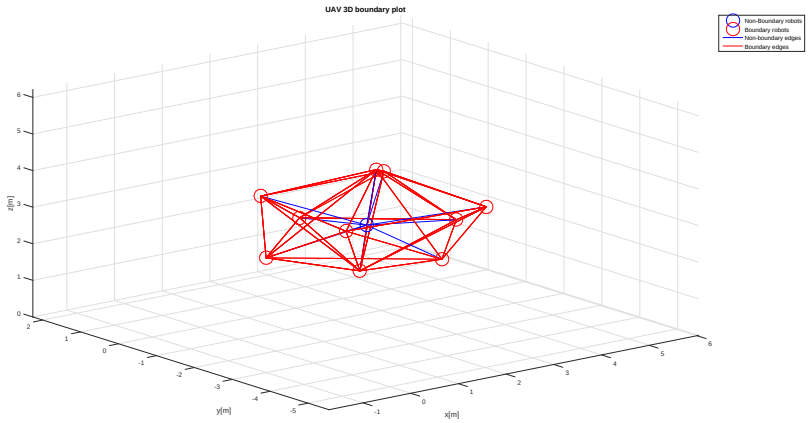


Figure 4.1: Swarm configuration with classified boundary robots is formed by 11 robots.

Figure 4.2 shows more complicated case with 44 robots forming the slightly deformed ball shaped swarm, but the principle is the same. Robots inside the swarm are surrounded by triangles formed by boundary robots.

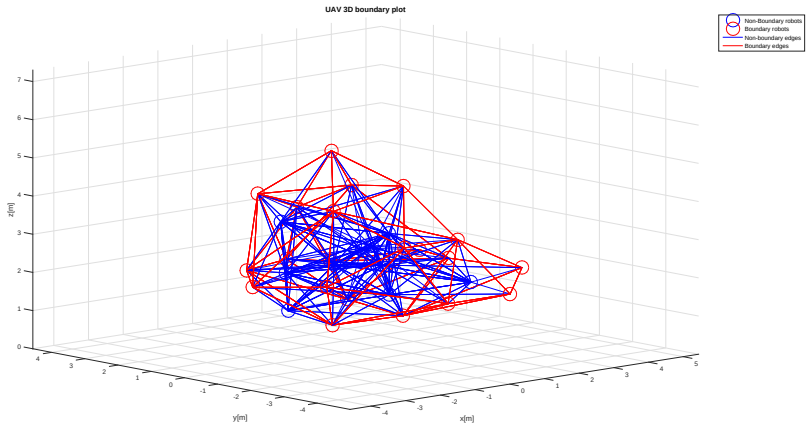
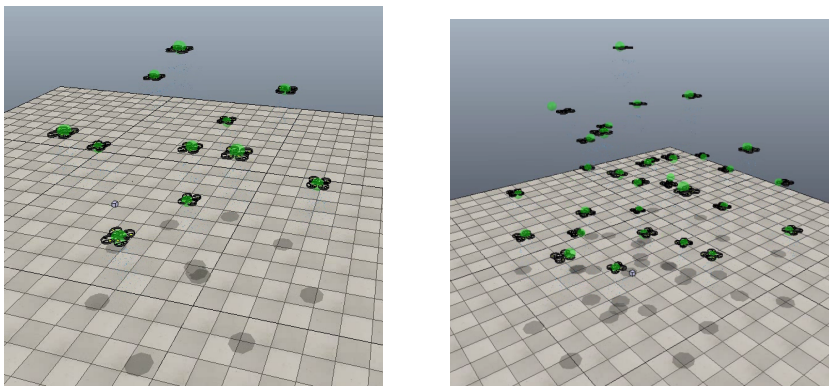


Figure 4.2: Swarm configuration with classified boundary robots is formed by 44 robots.



(a) : Same swarm configuration in V-REP as in Figure 4.1

(b) : Same swarm configuration in V-REP as in Figure 4.2

Figure 4.3: Screenshots from simulations performed to verify Boundary detection 3D.

4.4.2 LAP detection 3D

To verify the implementation of *LAP identification* algorithm in 3D UAVs were placed in the initial configuration containing LAPs.

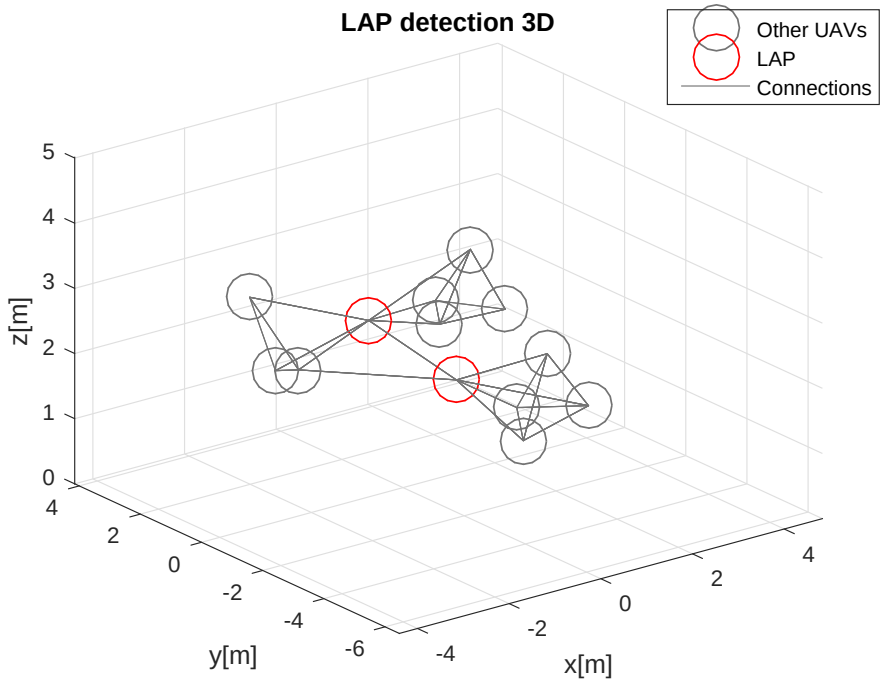


Figure 4.4: Configuration from the simulation verifying improved LAP detection invariant of dimension, where robots operate.

4.4.3 Boundary tension 3D verification

To verify the behavior of *Boundary tension* algorithm in 3D simulation with a small ball shaped configuration formed by 11 UAVs was performed.

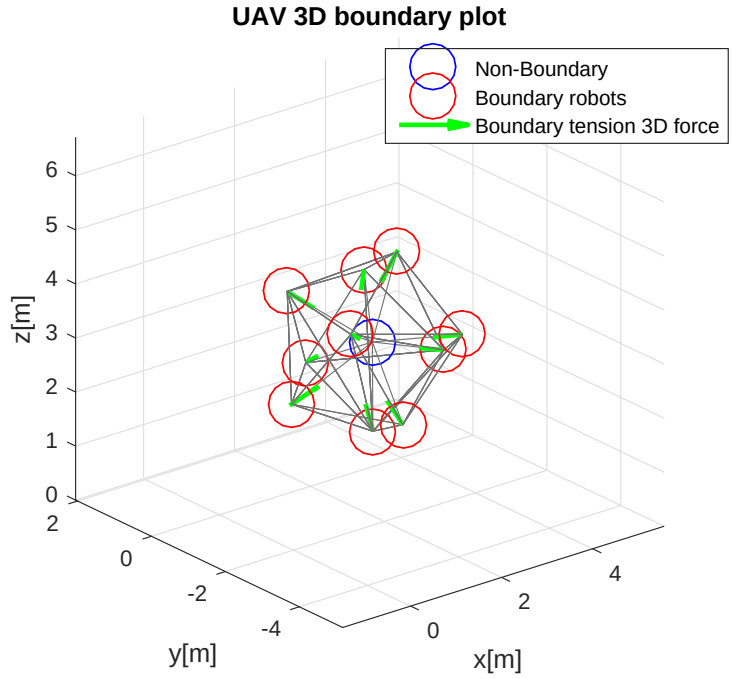


Figure 4.5: Graph of the simple swarm configuration with 11 robots shows principle of extended Boundary tension algorithm. Analogically to the 2D version, the direction of the induced force points inside the swarm in case of a convex robots.

Figure 4.5 shows classified boundary robots affected by Boundary tension force, which points inside the swarm leading to higher cohesiveness. Same as in 2D case, applying Boundary tension force affects the overall shape of the swarm and leads to higher density inside the swarm.

4.5 Conclusion

In this Chapter, the proposed 3D extension of the swarming algorithm has been analyzed. *Boundary 3D detection* and *boundary 3D tension* algorithms have been presented and mathematically stated in Subsections 4.2.2 and 4.2.3. Optional extension of the *LAP detection algorithm* has been explained and stated as well. The behavior of the introduced algorithms has been experimentally verified and presented by swarm topology graphs capturing states detected by algorithms in Section 4.4.

Boundary detection 3D algorithm gives acceptable results with occasional classification errors (see Figure 4.2, where classification error occurs in left bottom corner). Boundary tension algorithm has been extended as a 3D analogy to the original algorithm so that they share some properties such as direction (i.e. inside/outside swarm) of boundary force (see Figure 4.5).

Chapter 5

Implementation and integration into the ROS environment

5.1 Introduction

The goal of this chapter is to integrate the designed algorithms into the Robot Operating System (ROS), verify its behavior with MAV models in Gazebo robotic simulator and adapt the system for using with relatively localized multi-MAV system [25] of Multi-Robot Systems group at CTU in Prague.

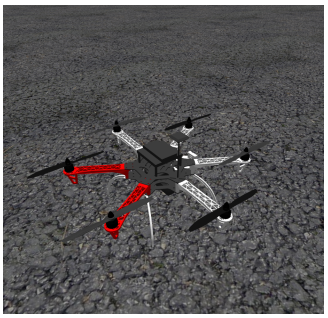


Figure 5.1: Model of MAV in Gazeboo simulator used in simulations

5.2 ROS overview

The Robot Operating System (ROS) is a robotic software framework and provides a collection of tools, libraries, and conventions. Framework simplifies the process of creating complex and robust robotic application and also allows collaborative development. ROS provides client library in Python and C++, which has been used for the system integration.

ROS is a network based system and provides anonymous and asynchronous messaging system between nodes using *publishers* and *subscribers* writing or reading from *topics*, which are strongly typed by message format definition allowing custom messages to be defined.

Multi-Robot System group at CTU in Prague uses MAVROS node to connect ROS directly with **PixHawk autopilot**. UAVs are equipped with **model predictive control** node, so that the computed reference position can be passed to the MPC regulator through the ROS messages using *publisher*.

5.3 Implementation

The designed system has been integrated into the ROS in two ROS nodes.

1. **Swarm controller** node contains definitions and implementations of proposed algorithms in a separated header and source files so that the code is reusable. The node represents agent's abstraction and is run on each UAV individually. Each UAV use publishers for broadcasting custom boundary classification and leader follow messages and subscribers for receiving odometry of its neighbors.

2. **Swarm dynamic reconfigure** provides dynamical reconfiguration of UAVs controlled by Swarm controller node through the service server. Similarly to the V-REP implementation this feature allows users to control important parameters (e.g. **desired distance**, **interaction range**, **dimension mode**) during the simulation.

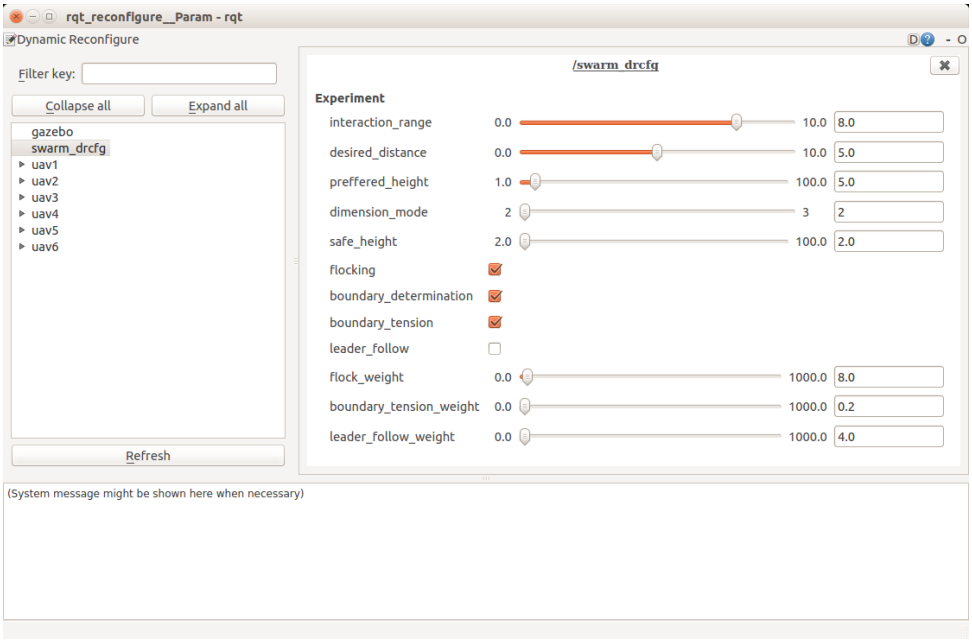


Figure 5.2: Screenshot of GUI used for dynamic reconfiguration of UAVs during the simulation.

The source code of the application is available at *https://github.com/charvja2/mbzirc*.

5.3.1 System for a relative localization adaption

Whole framework works with vectors of relative localization r_{ij} pointing to robot r_j in the coordinate frame of robot r_i. Thesis works with a presumption that the system for relative localization produces a localization vector in the cartesian or spherical coordinates. Node internally processes cartesian vectors, but transformations between conventional coordinates are trivial.

The thesis also considers a system for a relative localization to be able to cover a whole spherical area around the robot without blind spots so that the system has a spherical range. Violation of this condition could lead to neighbors getting closer without mutual notifying resulting in a crash.

Global position system (GPS) can be used as well, but it is redundant. Despite that fact, for some real world applications using GPS and emulating outputs of a system for a relative localization from obtained GPS data could reveal less demanding than integrating a complex system for a relative localization. It is worth considering, which localization system is more suitable because classical GPS could not be able to satisfy required accuracy for flight with low desired distance of UAVs.

5.3.2 Digital elevation model

As stated in Chapter 4, the second 3D extension aims to simulate swarm spreading in a terrain with a different elevation. Digital elevation model (DEM) is a 3D height map representation of a terrain's surface, which is often used in geology, geomorphology, geography and hydrology. DEM can be used as a model in Gazebo world to build an authentic environment with a different elevation.

DEM of the volcano, which promises challenging environment, has been used to build the world for simulations performed in Section 5.4.

5.4 Experimental results

Tests presented in this chapter focus on small swarms (i.e robot count $n < 8$), because of huge performance demands rising from including firmware of MAVs in the simulations and aim for a future real world experiment limited by a number of available MAVs. Larger swarms have been tested in the V-REP simulator. The aim of the experiments in Gazebo is to verify the possibility of using the designed algorithms for control of small swarm and mainly to test the possibility of using the MPC control system designed in our group for MAV low-level stabilization.

Robot communication range and range of relative localization system were set for simulations in this chapter to 8 meters. Robot desired distance was set to 5 meters. Robot preferred height used in 2D mode was set to 5 meters

and safe height used in 3D mode to 2 meters. Leader's maximal speed was limited by the limit 0.5 ms^{-1}. For tests performed in this Section original version of *Flocking algorithm I* using sum concensus was used.

5.5 Ball-shaped swarming

All figures in this section are related to the simulation at https://youtu.be/7gwgZpVSv8g. UAVs were initially placed into the approximately same height and formed a crystal like structure using algorithms of the extended Base swarm strategy. Figure 5.4 captures initial and final swarm configuration from the accelerated animation of the swarm topology at https://youtu.be/h7bha66l9kk related to the same simulation.

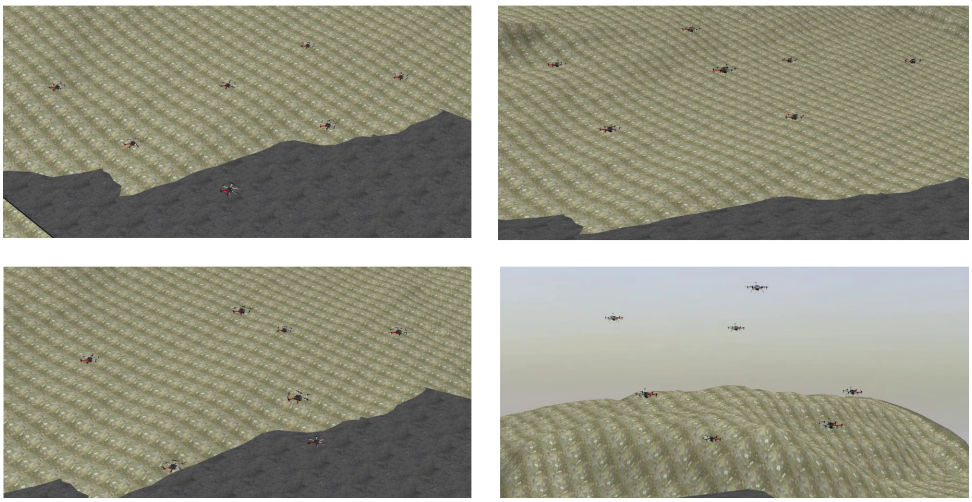
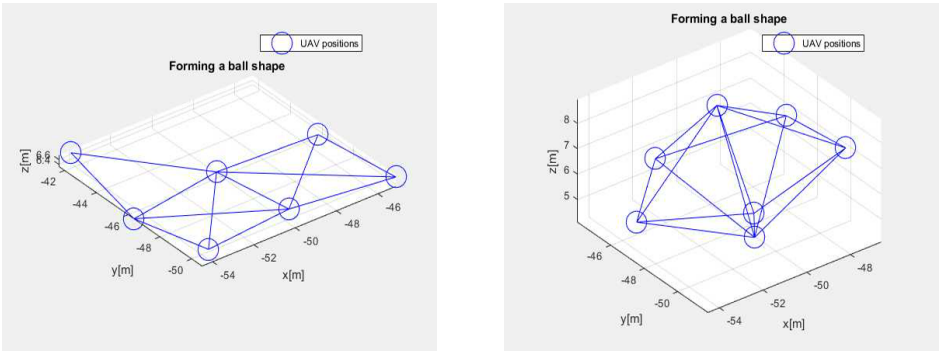


Figure 5.3: Snapshots from video at https://youtu.be/7gwgZpVSv8g showing forming of a ball shaped swarm using extended base swarm strategy



(a) : Swarm initial configuration **(b) :** Swarm final configuration

Figure 5.4: Initial and final swarm configuration from animation of the swarm topology at https://youtu.be/h7bha66l9kk.

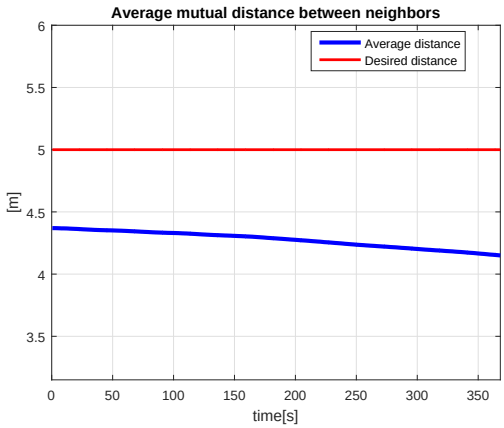


Figure 5.5: Graph shows averaged and desired distance between robots. Averaged measured distance never reaches desired distance, because of boundary tension force.

Figure 5.5 shows the average distance between neighboring UAVs, which is lower than desired distance, when using extended Base swarm strategy. Desired distance is reflected only by *Flocking Algorithm I*. Average distance decreases because of *Boundary tension 3D* algorithm causing higher density inside the swarm to the point where boundary tension force is suppressed by repulsion produced by *Flocking Algorithm I* and value of the average distance between neighbors stabilizes.

Figure 5.6 captures the evolution of heights of UAVs during the forming of a ball shaped formation.

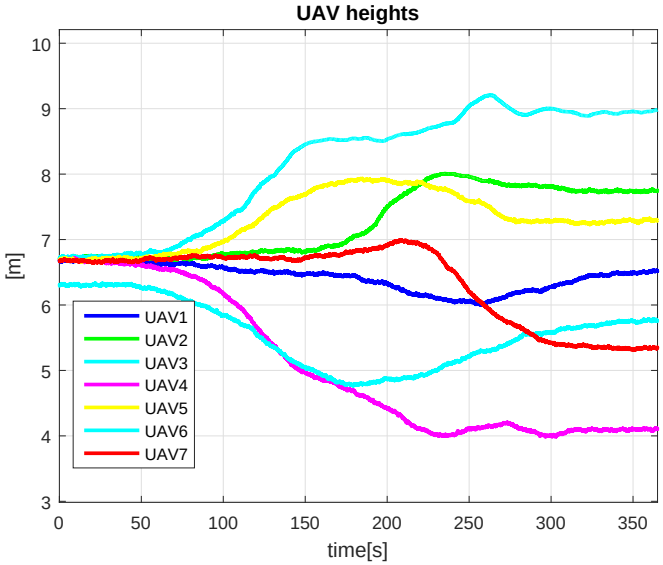


Figure 5.6: Graph shows heights of the UAVs. Graph proves that forming a ball shaped formations is possible even the robots are initially in the approximately same height.

5.6 Spreading in terrain with different elevation verification

5.6.1 2D mode test

Experiment in this section is motivated by many applications such as systematical searching for survivors or environment mapping. Leader follow strategy presented in Subsection 3.3.4 was used for moving the swarm over the hill. See captured simulation verifying swarm spreading in a 2D mode https://youtu.be/a7Nnf4xgchg. Animation showing the swarm topology during this simulation can be seen at https://youtu.be/Zu9sEeg5CPE.

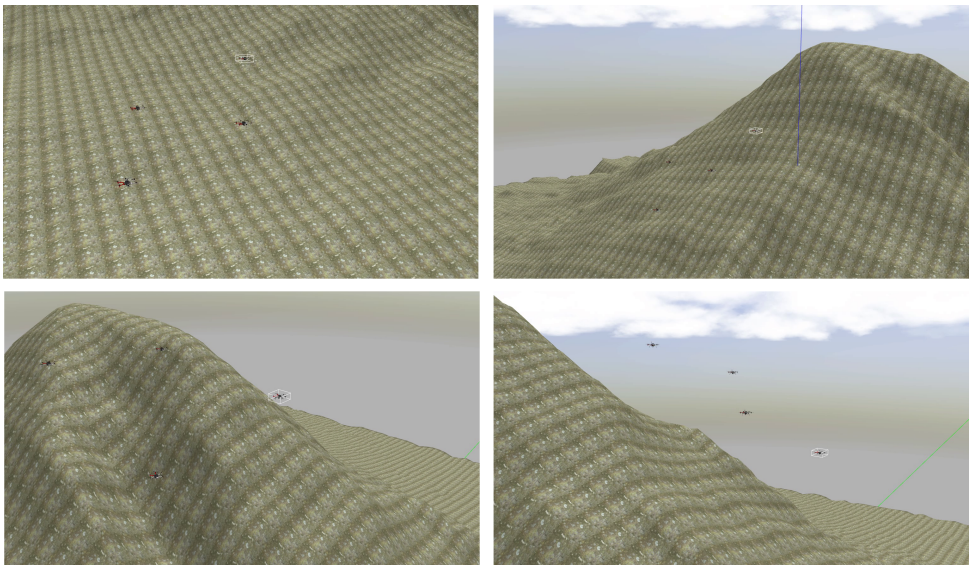


Figure 5.7: Snapshots from video at https://youtu.be/a7Nnf4xgchg show the swarm spreading in the terrain with different elevation using leader follow strategy in 2D.

See Figure 5.8 capturing UAV heights relative to the terrain during the simulation. All UAVs fly the desired height above the ground, which is the definition of the 2D mode. Note that deviation of measured height from preferred height is negative if UAV moves uphill and positive when downhill. Compare to Figure 5.9, where heights of UAVs measured relatively to the height of the home position are captured.

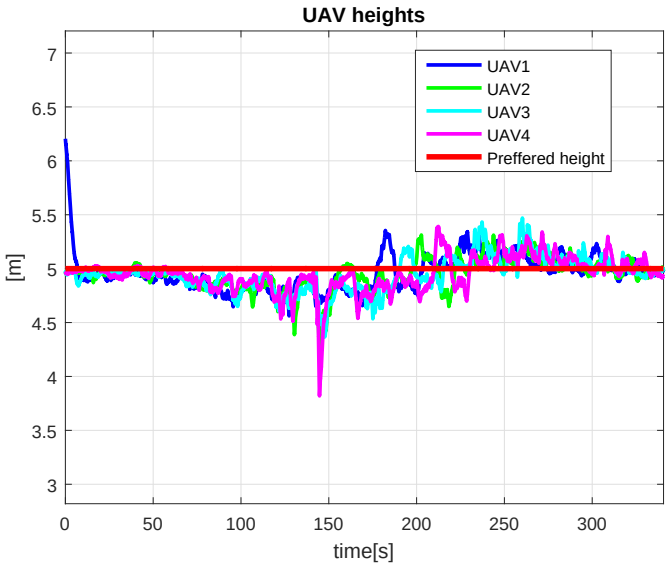


Figure 5.8: Plot of UAV heights shows the measured and desired heights during the simulation in 2D mode. All UAVs are supposed to fly the preferred height. Observed deviations could be caused by noise from altimeter sensor or by external forces.

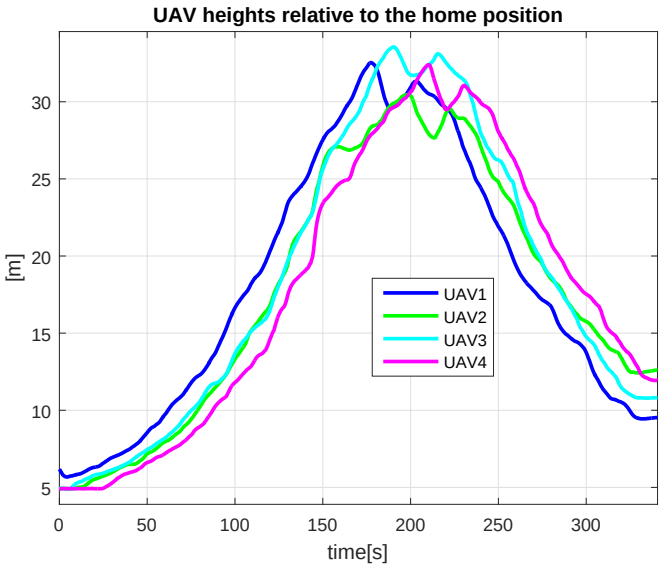


Figure 5.9: UAV heights measured relative to the height of the home position. First UAV is the leader determining the swarm motion.

5.6.2 3D mode test

For simulations verifying extended Leader follow strategy presented in Subsection 4.2.5 see https://youtu.be/xJ4QgddaeDM. See also animation related to the same simulation showing the swarm topology https://youtu.be/itdnY5rCaGI.

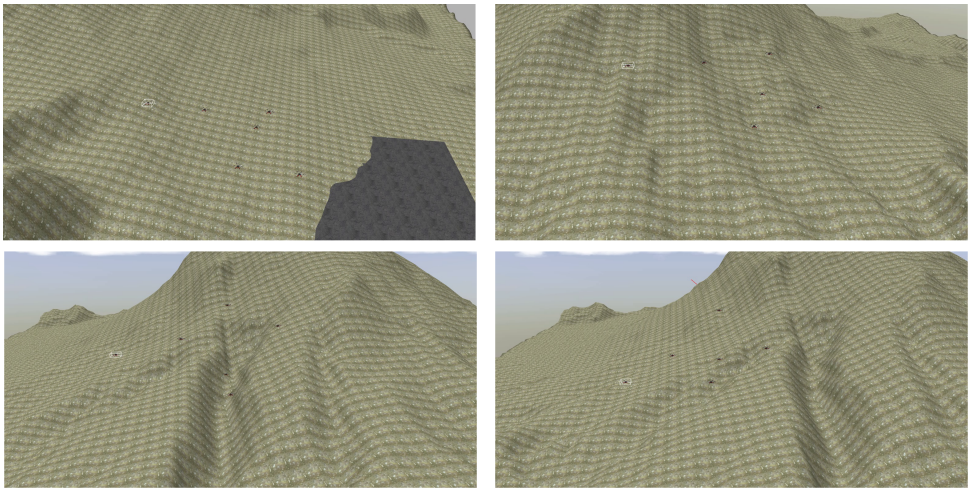


Figure 5.10: Snapshots from video at https://youtu.be/xJ4QgddaeDM show swarm spreading around the volcano in 3D mode. Swarm was moved by single leader flying the predefined trajectory in preferred height.

The simulation demonstrated the swarm spreading in an environment with a different elevation in a general formation. The Leader follow strategy in 3D can be used for moving the ball-shaped swarms formed by extended Base swarm strategy presented in Section 5.5. See Figure 5.11 for UAV heights during the experiment and compare to UAV heights during the experiment in the 2D captured in Figure 5.8 to see the main difference between 2D and 3D modes.

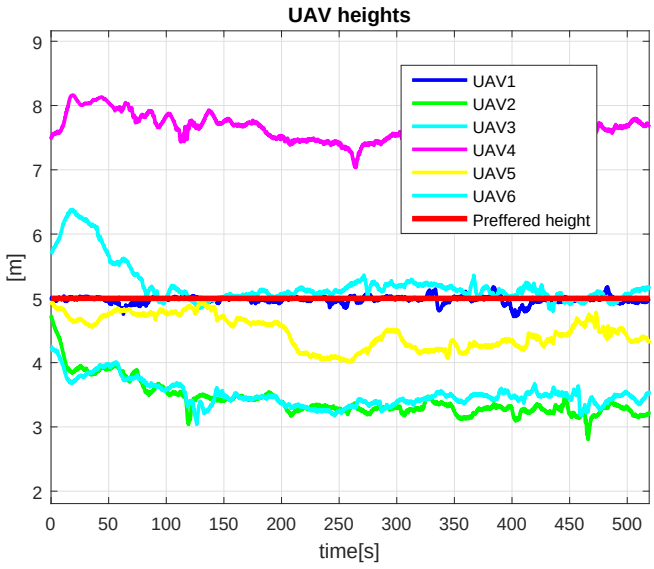


Figure 5.11: Plot of UAV heights related to the simulation shows main difference between flying in 2D and 3D mode (see Figure 5.8). UAVs are not encouraged to fly the preferred height so that the swarm can migrate in general formation, e.g. ball shaped formation presented in 5.5. First UAV flying in the preferred height is a leader following the predefined trajectory.

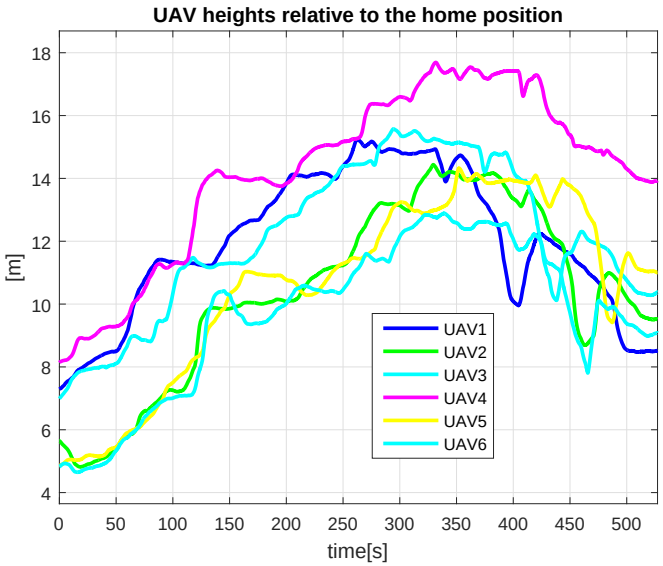


Figure 5.12: Plot of UAV heights relative to the height of the home position.

5.7 Conclusion

I integrated designed distributed swarm controller into the ROS using ROScpp. Forming of a ball shaped swarm is presented in Section 5.5. Designed system has been used to form a crystal structure, which approximates the ball shape (see Figure 5.4). Implementation has also been successfully used for swarm migration in a terrain in both modes. Figures 5.8 and 5.11 in Section 5.6 captures the difference between flying in a 2D mode and 3D mode.

Chapter 6

Conclusions

The contribution of this thesis is mainly extension of algorithms used to control ground robots in a plane for using with MAV. In Chapter 3, the original algorithms were adapted for MAVs, analyzed for usability with MAVs, explained and experimentally verified. Thesis also contributed by introducing mathematically stated algorithms for 3D mode: Boundary detection 3D and Boundary tension 3D in Chapter 4. In Chapter 5, integration of the complex distributed motion controller into the ROS was presented. Integrated motion controller can be used in the 2D or 3D mode for swarm migration in a terrain with a different elevation and allows an operator to dynamically control important parameters leading to comfortable MAV swarm control.

First assignment, which aims to understand and experimentally verify method presented in [11] and the referred algorithms in [12], [13] and [15], was accomplished in Sections 3.3 and 3.5. The second goal of this thesis was to extend the method into 3D for using with MAVs in two variants. The general approach for forming a ball-shaped swarm is designed in Section 4.2 and experimentally verified in Section 5.5. The second approach for swarm spreading in an environment with a different elevation is discussed in Section 4.3 and verified in Section 5.6. The final goal of the thesis was to integrate the designed system into the ROS and verify its behavior with MAV models in Gazebo, which is accomplished in Chapter 5.

The experiments in V-REP simulator verified that the designed algorithm enables to stabilize large groups of UAVs and to control them fully autonomously or using a minimum number of human operators that control the leading robots. In comparison with previous work of our group in the field of swarm control [27], [19], [22], [24], the presented solution enables to achieve the desired behavior more precisely and to control the swarm in a much more reliable way. The integration of the system into the ROS environment and the experimental verification of the system in the Gazebo environment enable its deployment in real-world scenarios. The credibility of the tests is ensured by the inclusion of the same PixHawk firmware that is used in HW platform of our group into the simulator (for HW description, see [25]). We plan to experimentally test the presented swarming approach using the onboard relative [10], [6] and global [4],[26] localization of UAVs in a similar

way as it was done in the previous studies of compact UGV-UAV formations [23], [26].

Abbreviation	Meaning
ROS	Robotic operating system
UAV	Unmanned aerial vehicle
MAV	Micro aerial vehicle
UGV	Unmanned ground vehicle
LAP	Local articulation point
DEM	Digital elevation model
GPS	Global position system
DFS	Depth-first search
BFS	Breath-first search

Table 6.1: Abbrevations used in thesis.

Directory name	Description
thesis	Bachelor's thesis in pdf format
thesis sources	latex source codes
sources	software source codes
videos	videos of simulations

Table 6.2: CD content.

Appendix A

Bibliography

[1] Gazebo, 2017. [Online; accessed 9-February-2017].

[2] Tomas Baca, Giuseppe Loianno, and Martin Saska. Embedded model predictive control of unmanned micro aerial vehicles. In *Methods and Models in Automation and Robotics (MMAR), 2016 21st International Conference on*, pages 992–997. IEEE, 2016.

[3] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6):926–939, 1998.

[4] Jan Chudoba, Miroslav Kulich, Martin Saska, Tomáš Báča, and Libor Přeučil. Exploration and mapping technique suited for visual-features based localization of mavs. *Journal of Intelligent & Robotic Systems*, pages 1–19, 2016.

[5] Prasun Dutta, S Pratik Khastgir, and Anushree Roy. Steiner trees and spanning trees in six-pin soap films. *American Journal of Physics*, 78(2):215–221, 2010.

[6] J. Faigl, T. Krajnik, J. Chudoba, L. Preucil, and M. Saska. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *ICRA2013: Proceedings of 2013 IEEE International Conference on Robotics and Automation*, pages 985–990, Piscataway, 2013. IEEE.

[7] K Ruben Gabriel and Robert R Sokal. A new statistical approach to geographic variation analysis. *Systematic Biology*, 18(3):259–278, 1969.

[8] Luca Iocchi, Daniele Nardi, and Massimiliano Salerno. Reactivity and deliberation: a survey on multi-robot systems. In *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 9–32. Springer, 2000.

[9] Yoshifumi Kitamura, Takaaki Tanaka, Fumio Kishino, and Masahiko Yachida. 3-d path planning in a dynamic environment using an octree and an artificial potential field. In *Intelligent Robots and Systems 95.'Human*

Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, volume 2, pages 474–481. IEEE, 1995.

[10] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562, 2014.

[11] D. Krupke, M. Ernestus, M. Hemmer, and S.P. Fekete. Distributed cohesive control for robot swarms: Maintaining good connectivity in the presence of exterior forces. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[12] S. K. Lee and J. McLurkin. Distributed cohesive configuration control for swarm robots with boundary information and network sensing. *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS). IEEE*, pages 1161–1167, 2014.

[13] J. McLurkin and E. D. Demaine. A distributed boundary detection algorithm for multi-robot systems. *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS). IEEE*, pages 4791–4798, 2009.

[14] Yogeswaran Mohan and SG Ponnambalam. An extensive review of research in swarm robotics. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 140–145. IEEE, 2009.

[15] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51:401–420, 2006.

[16] Anies Hannawati Purnamadjaja, Johan Iskandar, and R Andrew Russell. Pheromone communication simulation for mobile robots using java 3d. In *6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)*, pages 261–266. IEEE, 2007.

[17] C.W. Reynold. Flocks, herds and schoold: A distributed behavioral model. *ACM Siggraph Computer Graphics*, 21:25–34, March 1987.

[18] Coppelia robotics. V-rep, 2017. [Online; accessed 9-February-2017].

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil. Coordination and Navigation of Heterogeneous UAVs-UGVs Teams Localized by a Hawk-Eye Approach. In *Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 2166–2171, Piscataway, 2012. IEEE.

[24] Martin Saska. Mav-swarms: Unmanned aerial vehicles stabilized along a given path using onboard relative localization. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 894–903. IEEE, 2015.

[25] Martin Saska, Tomas Baca, Justin Thomas, Jan Chudoba, Libor Preucil, Tomas Krajnik, Jan Faigl, Giuseppe Loianno, and Vijay Kumar. System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization. *Autonomous Robots*, pages 1–26, 2016.

[26] Martin Saska, Tomáš Krajník, Vojtěch Vonásek, Zdeněk Kasl, Vojtěch Spurnỳ, and Libor Přeučil. Fault-tolerant formation driving mechanism designed for heterogeneous mavs-ugvs groups. *Journal of Intelligent & Robotic Systems*, 73(1-4):603–622, 2014.

[27] Martin Saska, Vojtěch Vonásek, Jan Chudoba, Justin Thomas, Giuseppe Loianno, and Vijay Kumar. Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent & Robotic Systems*, pages 1–24, 2016.

[28] Paolo Stegagno, Marco Cognetti, Lorenzo Rosa, Pietro Peliti, and Giuseppe Oriolo. Relative localization and identification in a heterogeneous multi-robot system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1857–1864. IEEE, 2013.

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Jan C h a r v á t

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Distributed Cohesive Control for Swarms of Micro Aerial Vehicles

Guidelines:

The aim of the thesis is to understand, re-implement and experimentally verify in numerical simulations method [1] designed for dimensionless particles operating in a plane and extend the algorithm for using with a swarm of micro aerial vehicles. Work plan:

- To understand method in [1] and the referred algorithms in [2-4] and to verify the presented results in V-REP robotic simulator.
- To extend the method into 3D for using with micro aerial vehicles (MAVs). Two variants of the extension will be realized. A) A general approach based on [4] for forming a ball-shaped swarms using limited range of communication and onboard sensors. B) An approach for MAV-swarm spreading driven by particular multi-robot applications (such as surveillance, active RFID localization, etc.) in environment with different elevation.
- To integrate the designed system into the Robot Operating System (ROS), verify its behavior with MAV models in Gazebo, and adapt the system for using with the relatively localized MAV system of Multi Robot Systems group at CTU in Prague [5].

Bibliography/Sources:

[1] D. Krupke, M. Ernestus, M. Hemmer, S.P. Fekete: Distributed Cohesive Control for Robot Swarms: Maintaining Good Connectivity in the Presence of Exterior Forces. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015.
[2] S. K. Lee and J. McLurkin, “Distributed cohesive configuration control for swarm robots with boundary information and network sensing,” in Proc. of Int. Conf. on Intelligent Robots and Systems (IROS). IEEE, 2014, pp. 1161–1167.
[3] J. McLurkin and E. D. Demaine, “A distributed boundary detection algorithm for multi-robot systems,” in Proc. of Int. Conf. on Intelligent Robots and Systems (IROS). IEEE, 2009, pp. 4791–4798.
[4] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: Algorithms and theory,” IEEE Transactions on Automatic Control, vol. 51, no. 3, pp. 401–420, 2006.
[5] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno and V. Kumar. System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization. Autonomous Robots. First online. 2016.

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

Valid until: the end of the winter semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, September 21, 2016