MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Uzakov Timur** Personal ID number: **453574**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Perception-driven Multi-Drone Formation Control

Master’s thesis title in Czech:

Řízení formace více dronů řízené vnímáním

Guidelines:

The goal of this research is to propose a multi-robot formation control for UAVs driven by perception of one or multiple targets. The proposed approach must be implemented in c++ based software, with ROS/Gazebo simulations and experiments with real robots (UAVs). This approach must be based in optimal control theories with a good description of the theory used and relevant survey of the state-of-the-art is needed. Experiments with measurable data and consistent metrics must be provided in the final Thesis.
In resume, the student must consider the following tasks for the case of study (experiments):
1. Perceive one or multiple obstacle during flight and converge to optimally observe them.
2. Optimal controller based on minimizing covariance of observation.
3. Experiments with real robots must be performed.

Bibliography / sources:

[1] T. P. Nascimento, A. P. Moreira, and A. G. Scolari Conceição, “Multi-robot nonlinear model predictive formation control: Moving target and target absence,” Robotics and Autonomous Systems, vol. 61, no. 12, pp. 1502–1515, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889013001310
[2] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert and Martin Saska. The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles. Journal of Intelligent & Robotic Systems 102(26):1–28, May 2021
[3] Yurii Stasinchuk, Matous Vrba, M Petrlik, T Baca, Vojtech Spurny, Daniel Hert, D Zaitlik, Tiago Nascimento and M Saska.. A Multi-UAV System for Detection and Elimination of Multiple Targets. In 2021 IEEE International Conference on Robotics and Automation (ICRA). May 2021, 555–561

Name and workplace of master’s thesis supervisor:

Tiago Pereira Do Nascimento, Ph.D. Multi-robot Systems FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **03.02.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Tiago Pereira Do Nascimento, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

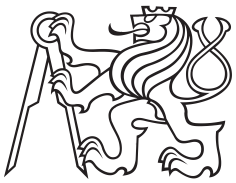
III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Perception-driven Multi-Drone Formation Control

Bc. Timur Uzakov

Supervisor: Prof. PhD. Eng. Tiago P. do Nascimento
May 2023

Acknowledgements

I want to thank many people who were involved in starting and continuing the project. They have contributed to my life such that the project did not stop and became successful. Mainly, I would like to thank Tomas Baca for teaching me how to reshape the formation of drones, Tomas Velecky for showing me how to operate trigger buttons in ROS, Tiago Pereira do Nascimento for providing me with the best optimization solver and guiding the entire work, Jorge Henriques de Araujo Junior for solving issues on the way of experiment, Lucas Nobrega for filming my experiments, Daniel Hert and Vaclav Pritzl for helping with MRS software system, Akash Chaudhary for showing me how to configure RealSense camera, safety pilots of Temeshvar camp.

I would like to thank everybody whom I met on the way toward completion of the thesis and wish them the best that they could have in their lives.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, May 11, 2023

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 11. května 2023

Abstract

In this work, a multi-robot system consisting of three drones is programmed to follow a geometrical shape during the flight and form it around objects recognized by a perception algorithm. In the thesis, formation control is proposed in a markerless, outdoor, GPS-enabled, decentralized manner. It combines robotics disciplines, such as computer vision, optimal control, and network dynamics. The network Kalman filter algorithm is proposed, and a state-of-the-art optimization solver is applied. The experiments were conducted in both simulation and real-world, where three drones were deployed to track colored plates on the grass.

Keywords: perception, drone, cooperative, control, formation, observation, multiple-object, detection

Supervisor: Prof. PhD. Eng. Tiago P. do Nascimento

Abstrakt

V této práci je multirobotní systém sestávající ze tří dronů naprogramován tak, aby sledoval geometrický tvar během letu a formoval jej kolem objektů rozpoznávaných algoritmem vnímání. V práci je navrženo řízení formace bezznačkovým, venkovním, decentralizovaným způsobem s podporou GPS. Kombinuje robotické disciplíny, jako je počítačové vidění, optimální řízení a dynamika sítě. Je navržen algoritmus síťového Kalmanova filtru a je použit nejmodernější optimalizační řešič. Experimenty byly prováděny jak v simulaci, tak v reálném světě, kde byly nasazeny tři drony ke sledování barevných talířů na trávě.

Klíčová slova: vnímání, dron, kooperativní, kontrola, formace, pozorování, více-objektový, detekce

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Problem Statement

Perception-driven formation control is a novel optimization-based algorithm. This approach allows drones to form a geometrical figure and follow a leader drone or a virtual leader. In our approach, the drones follow a leader based on using an RGBD camera. Furthermore, they observe the objects below and form a formation around them. This thesis' main topics are perception, sensor collaboration, and motion optimization.

1.2 Solution proposal

The work consists of two parts: perception and cooperative control. The perception part is about observing the leader drone (the first version) or objects below a drone (the second version) and using that measurement for further movement planning. The control part is about the optimization of the drone's motion. Experimentally, drones would follow a leader drone or a virtual point and form a formation around objects. The drones will communicate with each other via Wi-Fi. By sending each other their estimations of the leader drone position or estimated location of objects, drones will reach a consensus on a final value. The value is a collective measurement of the location of the leader drone(the first version) or the centroid(the second version). This measurement should converge to the same value for every drone in the formation.

Probabilistically, it is possible to address the circumstance of some drones observing the leader, whereas others do not. At least one drone should estimate the position of the leader. However, if one finds it, the other receives the message and adjusts their estimation to reach a consensus. The research consists of detecting objects with OpenCV using the blob detector. It is the starting milestone of the project. Furthermore, the cost function and non-linear convex optimization problem were formulated. By reaching the information about the control of multiple robots, it was possible to design a collaborative sensor fusion algorithm. The work contains the control theory of network control. The result of the applied idea is a relatively new modification

of the Kalman filtering approach with the introduction of network consensus.

1.3 Objectives and motivation

The motivation behind the thesis work is to use drones to help humans cooperatively. There are numerous applications of drone formation. For example, drones can assist electricians at power line towers or detect rubbish and clean the environment of trash. The benefit of this algorithm would be the autonomous control of drones and the absence of the need for multiple robot operators. Ideally, the goal is to achieve the ability to track objects in the environment and optimal motion control. There is a possibility of using the algorithm to remove waste.

1.4 Contributions

The novelty of the proposed algorithm in the thesis, compared to other scientists' results, is the following:

1) Every drone is considered a leader in the formation. In other words, algorithms and programs are identical on each drone. Every drone is equipped with the same number of nodes, and each can control through human interaction the whole formation. There is no central planning unit, which increases robustness. This algorithm feature is essential because the central planner creates a single point of failure. With improved communication schema between drones, the system can still accomplish its tasks even when some are faulty.

2) The virtual point that drones follow is the centroid of observed objects. Compared to previous work, the virtual point the drone follows is calculated based on observations of objects and taking their average. The calculation is accomplished in "form formation" mode. In "searching" mode, a user can control the centroid of formation or let it change in a circle. Ability to return to tracking observed objects by storing their centroid inside each robot's memory. The feature of the algorithm is to store not the positions of objects but their centroid and update it with time.

3) Ability to control formation when no objects are present through the switch from "searching" to "form formation" mode. When no objects are present, drones rely on their previous measurements; it is essential to start the formation with "searching" mode, detect objects, and then it is possible to return to them through mode switching.

Chapter 2

Literature Review

In recent years, scientists studied formation control. It has applications in territory surveillance, observation, search, and rescue scenarios. Formation control is one of the types of multi-robot motion that preserves a geometrical figure during the entire flight. Environment recognition benefits formation control, for example, collaborative object tracking. The problem is the ability to control the motion of drones in a geometrical shape with additional application of sensory data, which could bring more features and capabilities to the system of multiple robots. By observing objects, it would be possible to progress with the research of cleaning tasks by aerial robots and swarm-like monitoring of the environment. The requirements are decentralized schema, the ability to recognize objects in the environment, search in formation and form a geometrical shape around them. In the following sections, previous ideas and approaches of other scientists will be introduced and discussed with references and comparisons to our work.

2.1 Leader-following

Leader-following is a behavior that resembles the dependency of one position on another. According to paper [11], this pair of robots is a leader-follower couple. There can be multiple such pairs in the network of robots, and the agents create a formation. For example, the shape can be a triangle, two followers and one leader, a configuration of two leader-follower couples. The first utilization of such an approach was in the year 2002. Following those early works referenced by authors of paper [11], in 2007, the authors conveyed and proved theoretical aspects of formation control and validated it in experimental setups. The contributors presented their architecture of formation control. By a top-down approach, the algorithm consists of three levels: coordination, leader-follower control level, and entity control level. In such relationships of leaders and followers, the follower tracks a leader, whereas a leader is not dependent on the position of another robot. The coordination level calculates the paths of the whole formation or mutual trajectory. At the leader-follower level, there is a sensor feedback control loop. The follower tries to maintain its position and angle relative to the leader. The input to the tracking controller and actuators consists of high-

level coordination signals, data from a leader, and onboard sensor signals. Finally, the third level is responsible for role assignment to a robot. With the lead role, a robot follows a path and drives all other robots to a specified formation. When referenced as a follower, the task is alignment with a leader avoiding obstacles.

Interestingly, scientists studied centralized versions of formation control in 2019. The authors of the paper [12] have used a combination of centralized schema with a leader-follower approach. A module enables radio control of drones and communicates to a leader drone, which sends commands to the followers. In our work, some of the ideas of the work [12] resemble ours. Although decentralized, our approach keeps the ability to control formation to a human with a computer. In each drone, a node would enable a human operator to contact the drone and send the commands to the whole group. This approach has increased the safety of operations and, on the contrary, reduced autonomy. In 2019, there was represented yet another approach to leader-following formation control, which was decentralized. The authors have introduced a controller structure consisting of several layers. The architecture is composed of the application layer, formation layer, movement layer, and control layer. The first one defines the type of formation to be applied. The second is responsible for which drone would follow which leader. The third layer controls the goal position. The last layer is responsible for the speeds and acceleration of the drone. [1] In Fig.2.1, it is possible to see how the authors have implemented the approach. A V-like formation structure was used, and a global fixed-distance following between a follower and a leader was applied.

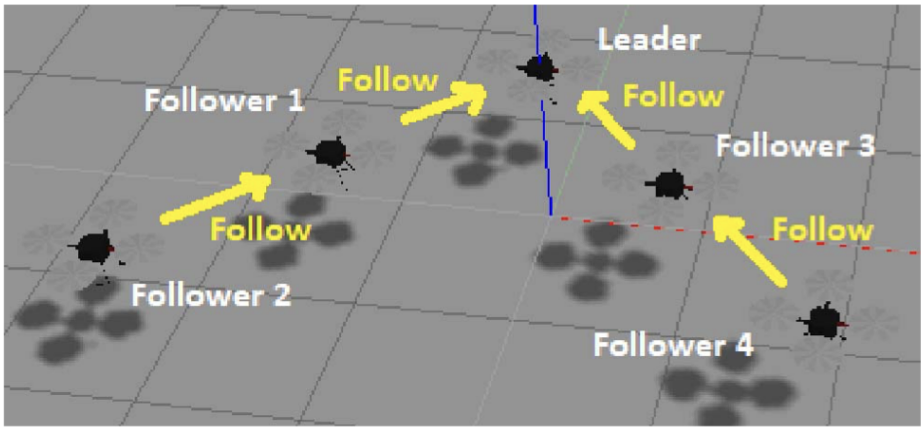


Figure 2.1: V-formation used in simulations [1]

In 2020, the multi-robot consensus was studied in a system with multiple leaders. The authors have proved that it is possible to reach an agreement between agents under formation control. Their graphs demonstrate that stability is guaranteed. Their contribution was to reduce network communication load by utilizing a unique algorithm for network graphs called Prim to cut

the OFTML(One follower connects to multiple leaders). They have reduced redundant connections in the system.[13] Another paper [14] published in 2020 has also proved the feasibility of formation control. The authors applied Lyapunov theory and delivered a series of equations to prove the system's stability. The convergence of agents' positions was depicted and is smooth. However, no practical, real-world experiments were provided. The experiment was done only in simulation. In another paper [2], which also has its experimental results in simulation, the authors proved that it is possible to follow a leader, even when no information about the leader's velocity is available. Distributed observers were used to find and follow the leader's trajectory. This approach leads to a more decentralized manner of formation control. Fig.2.2 depicts the obtained trajectories.

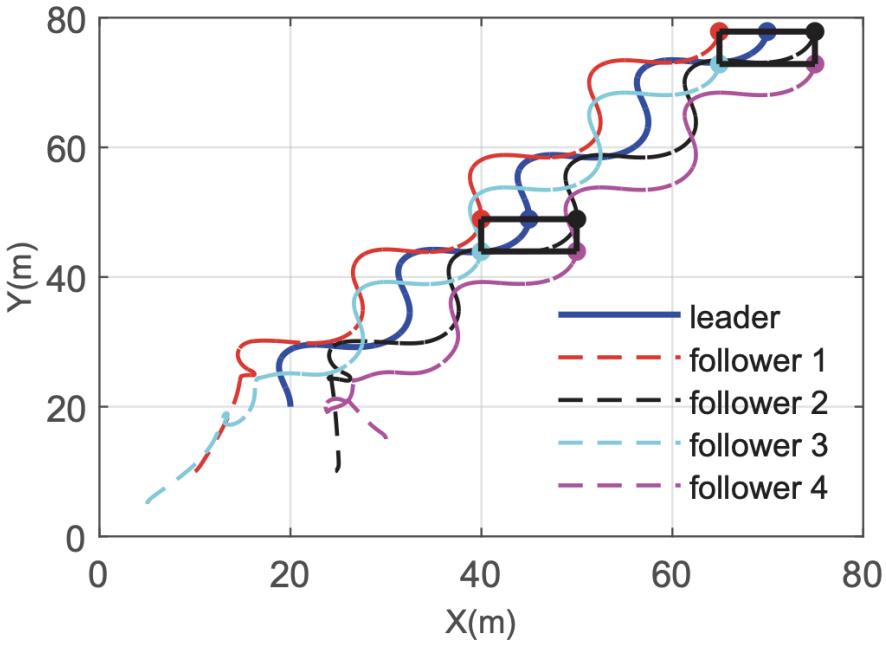


Figure 2.2: Trajectories of mobile robots [2]

The core idea of paper [15] is that the formation control introduced by authors is based on relative-position-velocity feedback between agents, and it was a requirement that at least one of the agents in the group could observe the leader. Other scientists have conducted similar research [16]. It is vital to mention that the authors have applied a leader-following approach combined with potential-filed obstacle avoidance. Moreover, adaptive control was implemented, and formation controllers were implemented in the first layer, and in the second, there was a potential field function optimizer. Lyapunov stability was proved. However, the results are only in simulation.

In 2021, a new concept in leader-following formation control was introduced and called event-triggered formation control. [3] Despite communication faults, the agents in the formation could follow the trajectory of a virtual leader.

A real platform of unmanned aerial vehicles (UAVs) technique has been proposed, and, despite degradation in the exchange of information, desired formation and following a virtual leader has been achieved. Fig.2.3 depicts more realistic trajectories of UAVs in a triangular formation.

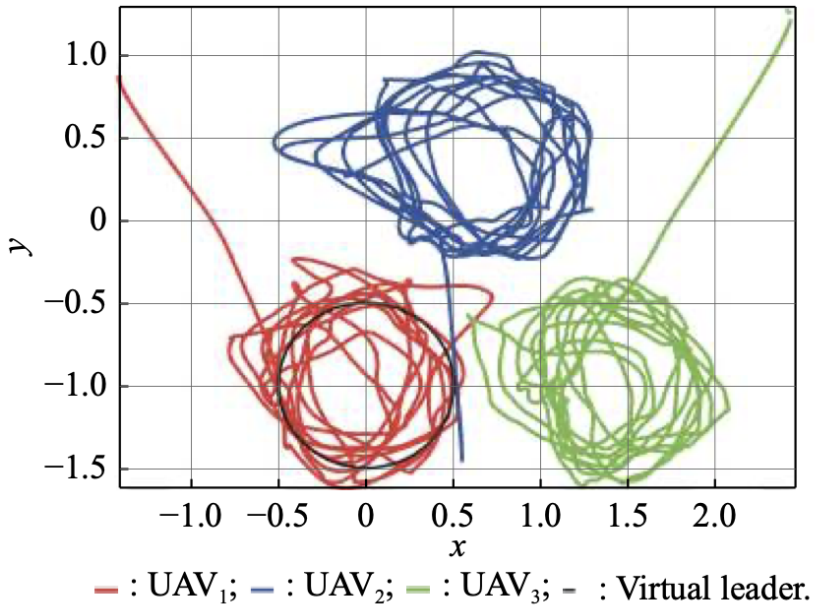


Figure 2.3: Trajectories of UAVs(proposed time-triggered approach) [3]

In 2022, there appeared research on formation control based on visual feedback. [4] The authors have used ground robots. However, followers could recognize a leader via a monocular camera. The vital part of this research is that the authors applied visual feedback in the control loop. However, they have tested the real-world experiment in the motion-capture system environment. The visual-feedback system is illustrated in Fig.2.4.

The most recent studying, 2023, was based on spacecraft research and the formation of space vehicles. The distributed control based on leader-following was presented. [17] The control theory was applied to satellites that recognized an asteroid as the leader, and the adaptive control law took into account the position of the space object. The research is bringing new frontiers in terms of robotics applications.

2.2 Model-predictive control approach

Early works in model-predictive control development were started in 2002 by Dunbar and Murray. The authors introduced the objective function, an optimization of non-linear constrained dynamics of robots. The utilization of Lyapunov analysis proves the stability of formation. [18] MPC advanced in 2005. A terminal state penalty applied to the objective function ensured control stability of formation. A terminal state region incorporated configured

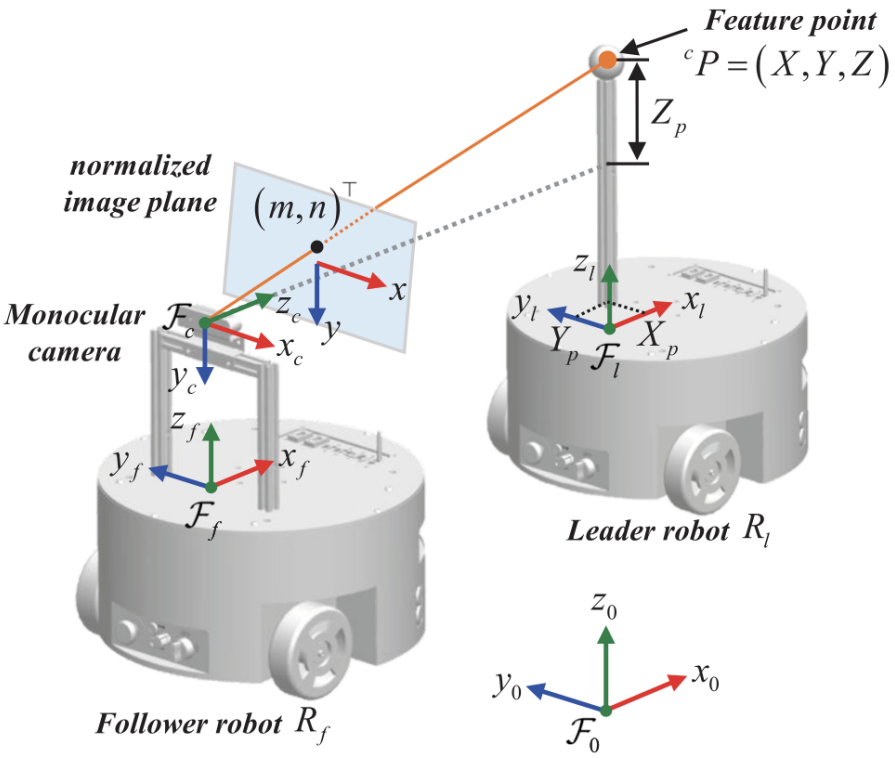


Figure 2.4: Leader-following system based on visual feedback. [4]

constraints.[19] In 2008, model-based and data-based model predictive control developed. The latter approach takes basis from input and output data, thus avoiding the first model identification step. The approach is a receding horizon, as a frame of optimal steps moves with time. Only the current optimal control signal reaches the system. [20] In 2011, the scientist applied the formation model predictive control to multiple drones. The authors have presented a hierarchical type of control based on linear model predictive control. The higher level of the controller is composed of shared measurements and position estimation. The approach is decentralized leader-based formation. [21] In 2013, scientists introduced a novel approach to model predictive control. The nonlinearity of the dynamic system of robots was addressed and investigated. The authors present a methodology to minimize a cost function, which is non-linear. They used resilient back-propagation and predictive state measurement, robots achieved the prescribed formation, and robot positions converged. The work incorporates two categories of tasks in robotics: formation-keeping and active target tracking. [22] In 2019, the authors conducted similar research to our first version formation control .[5]. The authors have used a convex-MPC formation controller to track and follow a human's position. The main finding of the research was that triangular formation reduces the uncertainty of measurement compared to linear or close-to-linear formation shapes. Fig.2.5 The same conclusion was found in our work, and the best tracking is when

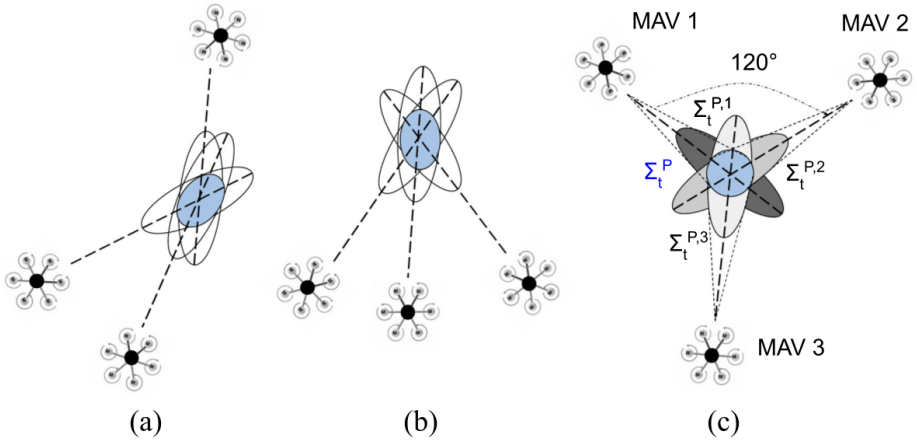


Figure 2.5: (a), (b) represent arbitrary configurations with large joint uncertainties. (c) is the optimal configuration for 3 MAVs used in this work. [5]

drones are separated by 90 degrees, meaning they can observe human motion in any direction. In 2021 [6], the authors have combined both ideas: leader-following based on model predictive control. Non-linear MPC has been used to control the formation of water surface vehicles. Their control approach consists of guidance and low-level control (other systems). The guidance layer is where global and local paths are computed. The other systems are navigation, control, and plant models. The NMPC proposed by the authors is also capable of addressing the disturbance. Fig.2.6 The robustness of MPC has been improved by authors of paper [23]. A switching MPC control was proposed for improvement in response to the noise of the controller. The hybrid approach is applied with an invented tube MPC and conventional MPC controllers. Essentially, tube MPC is the PID controller with augmented noise. In 2022, there was cutting-edge research on space robotics and the application of the model-predictive controller.[7] The authors used artificial potential functions similar to our cost-minimization strategy. The safety zones have been introduced with a specified radius for each satellite. This radius prevents other agents from the collision with the satellite. Additionally, collision zones were applied toward observed obstacles, and trajectories of the formation of two agents were depicted: Fig.2.7 In 2023, main combinations of MPCs, such as with DNN and PID, were presented by authors of the following papers: [24] [25]

2.3 Potential-field-based approach and obstacle avoidance

The artificial field-based control of robots is a well-known method in robotic planning. The cost function enables robots to reach their goals. The desired

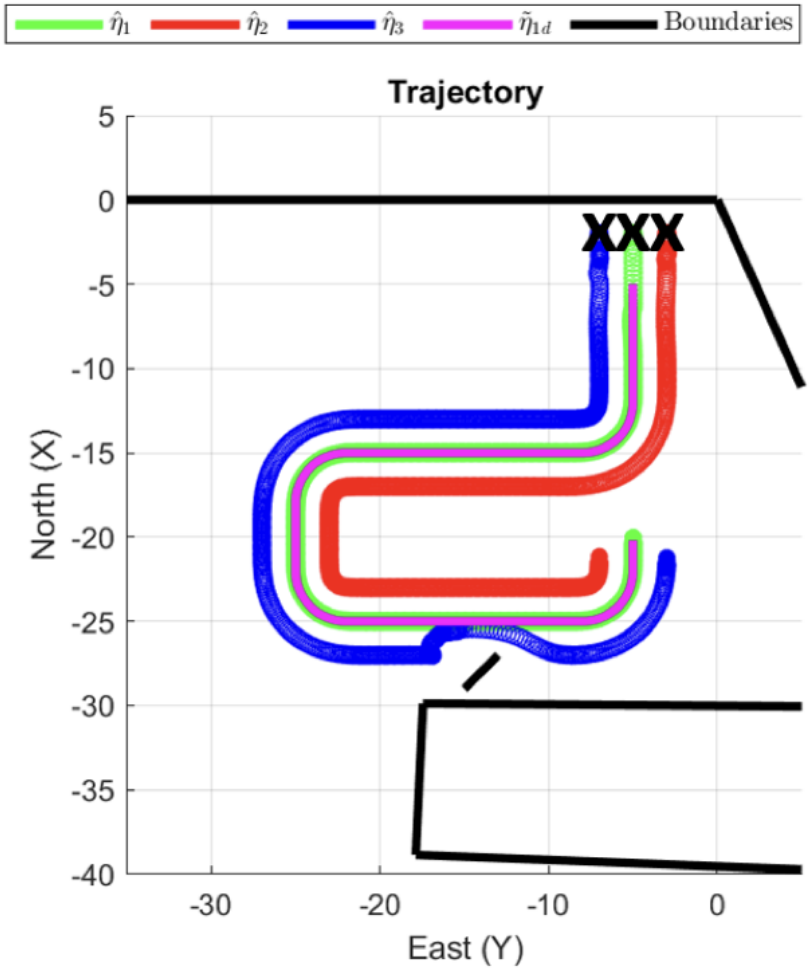


Figure 2.6: The sailed trajectory.(start at X) [6]

positions are programmed as attractive field points, whereas obstacles are sources of a repulsive field. The authors of paper [26] have implemented this technique to the formation task and solved related issues, such as trapping of robot in a local minimum. The formation structure consists of virtual nodes, which robots can occupy. In this case, it is V-shape and circular. The robot is assigned the node as a goal position and is programmed to minimize the distance between his state and goal.

The famous drone shows were introduced in 2020 by authors of the paper [8]. They have used potential fields, namely quadratic parabola functions, as the objective cost. Fig.2.8 The results have been verified in simulation and real-world experiments; the authors changed the formation from a dancing girl shape to a guitar shape. The path planning was applied to the 3D trajectory, meaning there was a multi-dimensional cost function. Although very successful, it is assumed by the aerial multi-robot system community

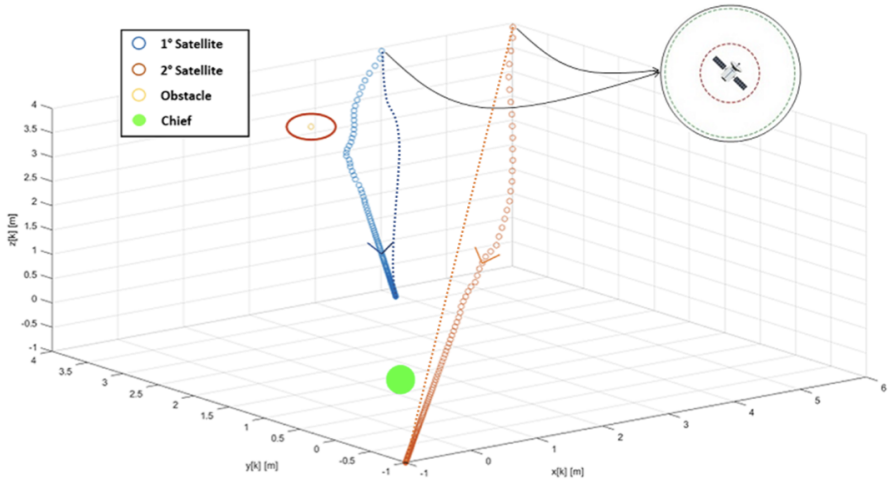


Figure 2.7: Simulation. Satellite trajectory. [7]

that this show is based on a centralized schema of control. Consequently, it is confirmed in paper [8] that the paths were loaded to each drone by Wi-Fi, and clock-synchronization using GPS-timing was applied. Nevertheless, it is an outstanding application of potential-field theory for multi-UAV planning. The authors of paper [10] have introduced yet another and improved approach to potential-field planning. The approach involves applying forces to each drone, such as attractive and repulsive. Fig.2.9 The technique was introduced at Czech Technical University as a swarming approach to UAV multi-robot systems. From experience, this approach works particularly well in simulation. However, it is not known if the algorithm is applicable in real-world scenarios. The authors' results (Fig.2.10) depict how drones fly with attractive and repulsive forces.

In 2021, authors of paper [27] used single path follower and multi-vehicle formation controllers to control unmanned underwater vehicles. The model of the vehicle was studied. The referenced authors in the paper previously transformed the problem from a formation control problem to a target tracking problem with a virtual pilot. They claim that introducing a virtual leader increases the system's stability and reduces problems related to local minima. Another improved version of the potential-field approach was introduced in 2022. [9] The control strategy is similar to the one in [10], but there are a lot of forces concerned. Interestingly, the authors applied potential-field planning through randomly selected balls, which were the obstacles, and successfully overcame them, at least in simulation. Fig.2.11 The main contribution and novelty of the algorithm is the rotating potential field around the drones. Consequently, the algorithm reduces the chances of robots' stagnation at a local minimum, claim the authors.

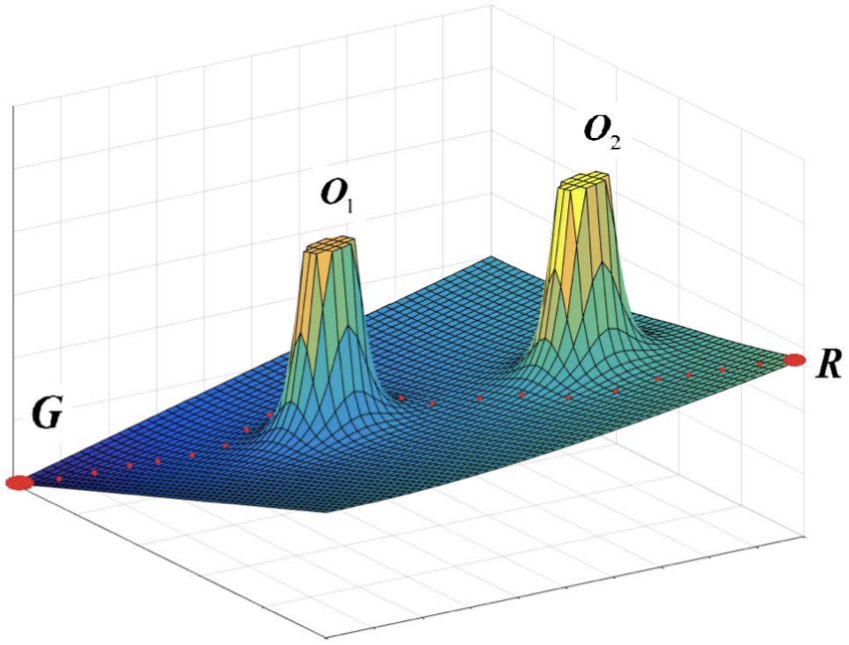


Figure 2.8: Schematic Diagram of Artificial Potential Field [8]

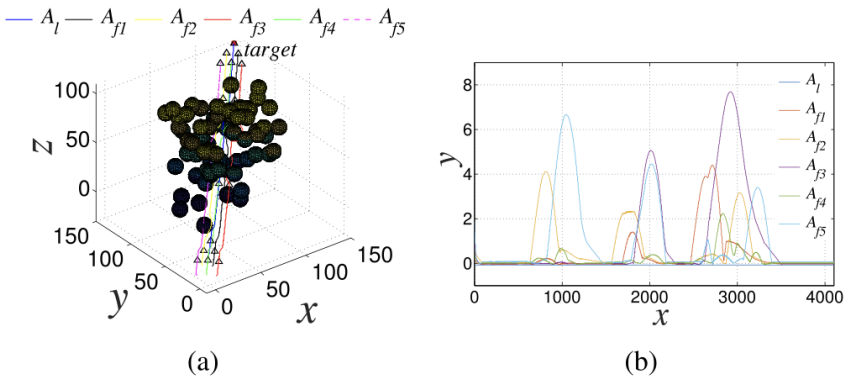


Figure 2.11: Case of random distribution obstacle environment: (a) Trajectories of six UAVs formation path planning and (b) trajectory tracking errors of follower UAVs. [9]

2.4 RPROP optimization solver

Resilient back-propagation optimization problem solver developed by Martin Riedmiller and Heinrich Braun in 1992. The solver can find an optimal weight in several steps of calculations. The calculation steps are programmed to reduce the cost function of the robot controller. The optimal weight enables a drone to reach a prescribed destination position. [28] Based on that algorithm and the addition of perception capability to robots, our first

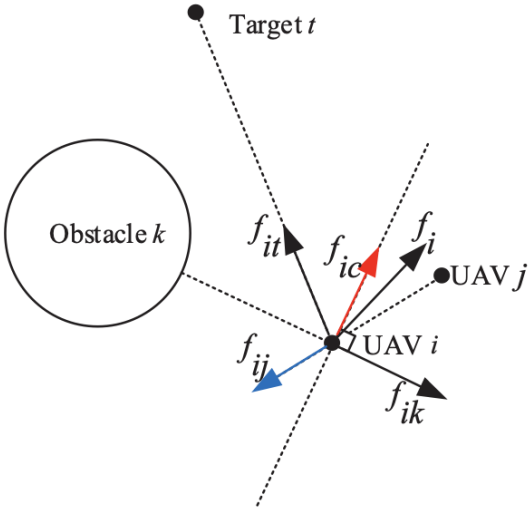


Figure 2.9: Schematic Diagram of Artificial Potential Field [10]

work was introduced in 2020 [29]. The perception was used to track humans in the environment, sensor fusion was applied, a consensus approach was implemented, and a better version of the RPROP algorithm was used to guide robots toward desired goals.

2.5 Latest formation control

In their paper, Visual Inertial Odometry Swarm: An Autonomous Swarm of Vision-Based Quadrotors [30], authors have introduced another possible solution to robotic swarms without the usage of GPS and motion capture systems. They applied perception algorithms. However, they have also designed the environment. They placed tags, observed by drones, in the area. In this approach, perception-driven formation control requires a specialized area. The proposed idea, although seemingly promising, is less versatile than drones using a navigation system (GPS). The advantage of the proposed research is the achievement of better, more precise navigation compared to GPS and motion capture approaches. Similarly to our work, the authors [30] have also used RGBD cameras for perception-based formation control. However, in our work, the goal was to achieve the ability to apply the algorithm to any environment, such as a field or beach.

In 2019, authors of papers [31], [32] demonstrated how to apply deep neural networks to ground robotics formations. The concept consists of two phases of the typical learning procedure: training and testing. The authors applied the Q-learning theory to control mobile robots. Although decentralized and the algorithm requires no communication means between robots, it still has some flaws. Using this approach, one has to introduce a training environment, rewards, and actions. Such a type of control is very applicable to the simulation environment but hard to achieve in the real-world scenario, where the operating conditions vary from a grass plane to a forest

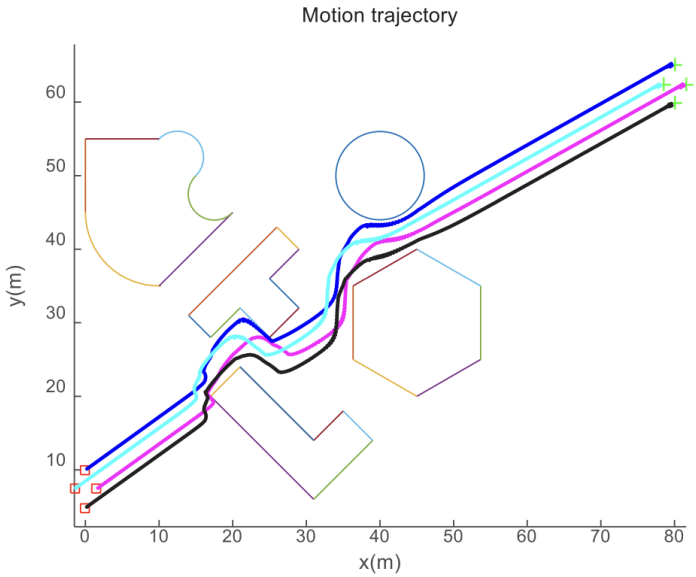


Figure 2.10: Schematic Diagram of Artificial Potential Field [10]

or a beach.

Authors of the paper [33] in 2021 studied ideas to fly drones based on the perception and recognition of other drones. They used DNN to perceive the drones. However, the authors also installed additional markers on the drones. This studying paves the way for further research and application of GPS-less, DNN-based visual formation control. In 2021, [34], the authors also applied a gradient-based control strategy and relied on perception algorithms. This approach could be a potential milestone for further research on the formation control of heterogeneous robots. In 2022 and 2023, scientists mainly studied robustness and aggressive formation tracking. The research is, at the moment, focused on improving the control part of the formation algorithm. [35] [36] [37]

2.6 Discussion

In conclusion, it is desirable to summarise the ongoing research in visual-based formation control of multiple UAVs via a similarity and dissimilarity table, which is provided here: 2.1

Paper	Similarity	Dissimilarity
[1]	Decentralized, formation and control layer	Number of layers
[2]	Tracking the leader through perception	In our case, a virtual centroid of observed objects
[3]	Robust to faults	Observation-based virtual leader
[4]	Visual-feedback	A motion capture system was used
[5]	Convex-optimization, agreement on triangular formation benefits	In our case, objects instead of human
[6]	Non-linearity	Model-based
[7]	Cost-minimization strategy	Satellites instead of drones
[8]	Potential-field	4D cost instead of 3D, centralized
[10]	Attraction and repulsion	Swarm technique
[9]	Attraction and repulsion	Rotating potential field
[11]	Following a leader	Drones are assigned a role; every drone is a leader
[12]	The ability of human control formation	Our work is decentralized and safe
[13]	Network consensus, analysis of stability	Network redundancy solved
[14]	Consensus, analysis of stability	Real-world experiments
[15]	At least one agent should observe the leader	Instead of relative distance feedback, attractive and repulsive filed were applied
[16]	Potential-field based	Adaptive control
[17]	Object recognition as a leader	Adaptive control
[18],[19] , [20],[21]	Non-linearity, optimal time-frame decision-making, decentralized	Model-based
[22]	Active target tracking	In our work, memory concept to retract lost objects
[23]	Mode switching	Rapid automatic switching instead of manual mode switching
[27]	Target tracking problem with a virtual point	Path following
[30]	Perception-based	Authors used tags instead of object recognition
[31], [32]	Decentralized	Deep neural-networks based
[33]	Perception-based	DNN-based with markers
[34]	Perception-based and a gradient-based control strategy	Heterogeneous formations
[35] [36] [37]	Robustness	Aggressive formation tracking
[26]	Goal-distance minimization, potential-field based	V-shape, or circular formation

Table 2.1: The comparison of the work of the thesis to the accomplishments of other scientists.

Chapter 3

UAV Perception

3.1 Detection

The first and foremost reason for applying a blob detector instead of other sophisticated algorithms, such as Deep-Neural-Networks, is because a blob detector is more robust and can avoid errors that most algorithms have. For example, students experienced that DNN would falsely classify a goal but not precisely. However, the blob detector does not need to consider the features of an object. It computes the mask of an object within a specific range of color and, thus, is not likely to mismatch the thing. Another reason for using the blob detector is that it is easy to configure. In other words, it is easier to set up the blob detector and tune it than to retrain the neural network and adjust the learning rate. Finally, in a natural environment, the shape of the image of the leader drone would be slightly different from the simulation image, which could result in false detection. In contrast, the blob detector would be the same, identifying the most prominent color contour within a range configured to suit the real-world environment.

3.2 Coordinate extraction

3.2.1 Extracting a global position

After the recognition step of the work, it was possible to obtain the center of the blob in the image frame and its coordinates in the image. The coordinates in the photo start from the left upper corner and stretch to the right lower corner. Since the values are in pixels and are too large, they are not a realistic representation of the position of a detected object. It is necessary to find the actual global place of the thing and use it for motion optimization afterward. The global position of a leader drone is the drone's position in a world frame which is common to all other drones, such that the origin is at the same place for each robot. Using a global frame of reference to synchronize robots' motion is necessary for mutual observations in the joint coordinate frame.

The global position of the leader drone is essential for collaborative following. The followers would receive and compare their measurements with other

measurements via communication. Sending position in the image frame would result in the wrong estimation of the leader drone, thus breaking the whole algorithm. The transfer from image frame to drone frame is essential because, without it, there is no possibility of synchronization of drones in one observed point.

3.2.2 From image coordinates to drone coordinates

The image coordinates obtained from the blob detector are for the origin, located in the upper left corner concerning the drone. The x-axis stretches to the right, and the y-axis stretches to the bottom. Fig.3.1 The problem is that the coordinate system should align with the center point of the drone or the camera position. Therefore, with this frame of coordinates, the detection position is relative to the upper corner. That is why it is necessary to shift coordinate values. Fig.3.2

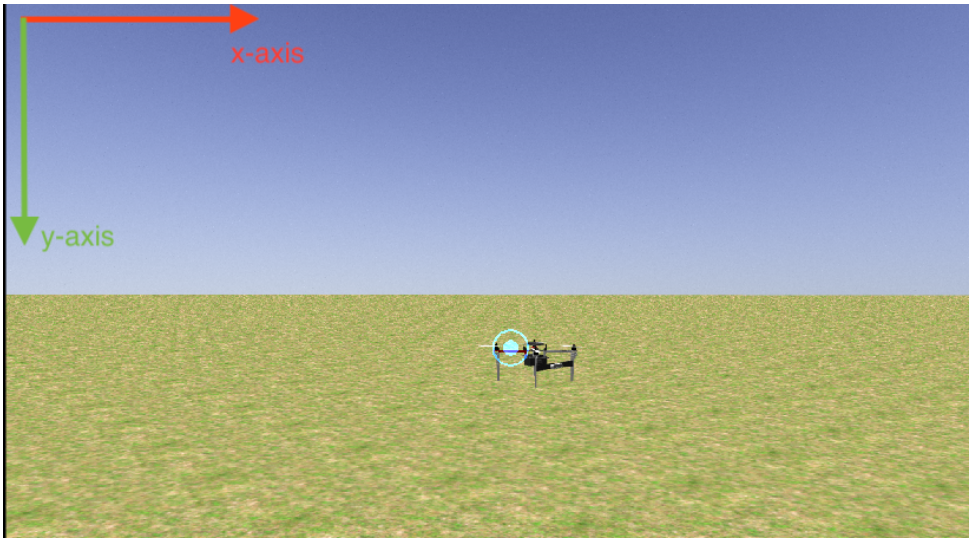


Figure 3.1: Image representing the coordinate axes on camera image

The approach is to consider image sizes: width and height. By taking half the values of the image dimensions, it is possible to define how many pixels must be removed for the current origin (upper left corner) to align with the center of the image. Subtracting the two calculated values from the obtained measurement in the upper-left corner system is necessary. The coordinates calculated are in pixels, but it is necessary to know their meter representation. There are two approaches to this task: to measure pixels and distance or to look for technical parameters of the RealSense camera. The programmer can do the first method by using a cube with a side length equal to 1 m and measuring the number of pixels in the image. However, it is far better to look for the technical parameters of the camera. Finally, a scale of 0.005 works well with conversion.

Because now, the y-axis points and stretches downwards, it is necessary to flip the axis or to put minus in front of the y-axis scaling factor. In the

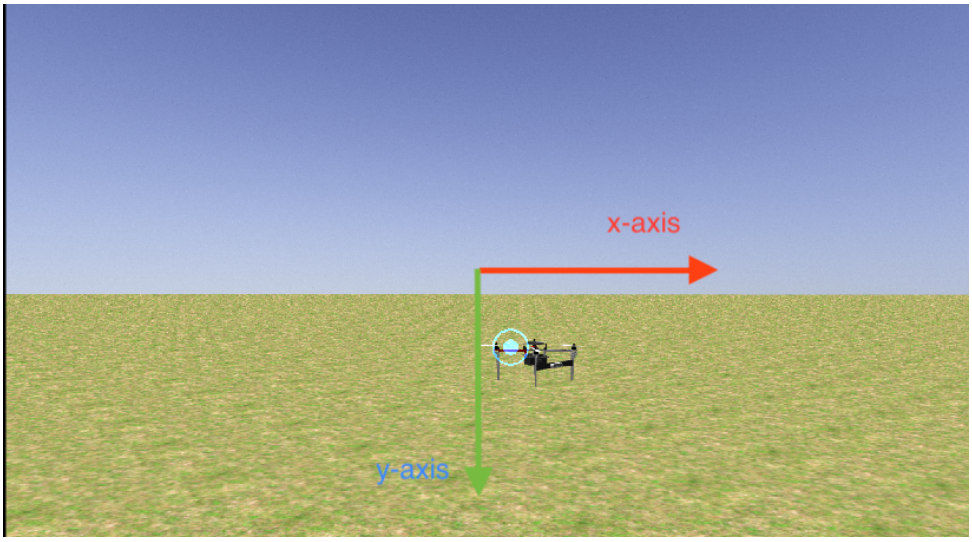


Figure 3.2: Image with coordinates axes shifted from the corner to the center of camera image

Gazebo simulator, the XY plane of global origin is located on the floor or grass, whereas the drone's XY plane is on the captured image plane, which needs changes. Since the y-axis of the drone points upwards, it coincides with the origin's z-coordinate, which is also upward. The difference was to multiply by the following matrix:

$$RotationMatrix = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.1}$$

Furthermore, new y now coincides with old z, from drone to the front. However, in global origin coordinates, this stands for the X-axis. Therefore, the current drone's x and y axis should flip by the following matrix:

$$RotationMatrix2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Finally, there is a last misalignment. That is new y-axis does not coincide with the global y-axis. It is in the opposite direction. So the following matrix is applied:

$$RotationMatrix3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Now, the drone's detection is in its coordinate frame. However, it depends upon the value of the drone's yaw or its viewpoint. Therefore, it is essential to compensate for the effect of rotation around the z-axis. Initially, the robot subscribed to the odometry topic and extracted yaw values from there.

This approach turned out to be false because mistakes started to appear. Eventually, it was found that yaw values from the odometry sensor were from -1 to 0 and 0 to 1, to the left and right, respectively. Those values are not suitable for the computation of trigonometric functions. Finally, by subscribing to EstimatedState message from MRS messages, it was possible to obtain angle value in radians, which was suitable for final computation and multiplication by the following matrix:

$$RotationMatrix4 = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

,where ψ is heading

$$CameraOffset = \begin{bmatrix} \alpha * cos(yaw\_value)) & \alpha * sin(yaw\_value)) & 0 \end{bmatrix} \tag{3.5}$$

, where α is the distance from the center of a drone to the camera

3.3 Filtering

The coordinates extracted from blob detection are in the raw format and require further processing. The processing is essential for several reasons, such as noise and rough estimations. Noise can contribute to the decision process that the drone makes, for example, for its movement. The coordinates, extracted from the blob detector algorithm, will be further used to optimize how the drone moves. Therefore, provided that the drone measures the rough and raw coordinates of the detected object, the drone would constantly estimate the jumping values due to noise. The process will lead to non-smooth estimations of the optimization algorithm's position and result in the robot's abrupt movements. Another reason is that it is necessary to have some means of comparison between measurements to determine whether they are good or bad. The measurements can be compared using probabilities and especially the matrix, known as the co-variance matrix. This matrix defines the probability distribution of the detected object by measuring the variance of each coordinate. Initially, a programmer can set it to identity and later process it by Kalman filtering. For example, the co-variance can also be used in the optimization phase to minimize it. Therefore, it is essential to filter the measurements.

To conclude, raw measurements will be disadvantageous for further usage in the algorithm that governs the drone's motion. Abrupt jumping values result in chaotic movements. Hence, filtering would improve the observations and smooth the values for the optimization algorithm, producing finer and better-quality motion.

3.3.1 Matrices and Model

For the model, the fundamental equation of motion was applied. The equation states that the position in frame k+1 is equal to the one in frame k and the integration of velocity at k. It is the following:

$$\mathrm{x}_{k+1} = x_k + v_k * t \quad (3.6)$$

Where x is a state vector, v is the velocity vector, k is the time frame, and t is the length of the one-time frame. A more convenient description of the change of states is the following:

$$\mathrm{x}_{k+1} = A * x_k + B * u_k \quad (3.7)$$

The state of the system is composed of position x, y, z and velocities v_x, v_y, v_z.

The transition matrix is the following:

$$A = \begin{bmatrix} 1 & 0 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.8)$$

And its simple version is the following:

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.9)$$

Initially, lecture slides from Czech Technical University represented the Kalman Filter algorithm. However, recently it was found that in OpenCV, there exists an implementation of KF already. Therefore, the library was used, leading to shorter code and better understanding. The algorithm computes position value by first making a prediction, then transferring the forecast with

transition matrix A as in the model, the covariant equation is instantiated to some initial value and then also multiplied by matrix A from both sides, co-variance of measurement noise is also predefined and then calculated through Kalman Filter. The algorithm adjusts the gain towards the value that is most probable. Finally, the programmer can obtain the resulting position together with overall co-variance. It is essential to pre-configure some parameters of the Kalman Filter, such as measurement and process noise co-variance, because they play a significant role in how the filter will behave. For example, changing measurement noise co-variance to some minimal values would result in reliance upon measured value, whereas minimizing process co-variance would favor modeled value. From samples from the library, the importance of co-variance matrices is set to some small weights, while the matrices are diagonal. Initially, the co-variance matrix values should be significant, and 10 was chosen for both matrices. Finally, the simple model of object tracking works well enough to measure the position. The implementation of KF in OpenCV is good enough and clear enough, for example, to set parameters and configure the transition matrix. The performance in C++ has proved that the filtering is effective, as the user can observe the smooth following of the object position in the camera image.

3.4 Cooperative detection

To be able to sense an environment with multiple agents raises both benefits and challenges. Mutual observation can bring about many opportunities, such as better object detection, better localization of an object, ability to sense the thing while others cannot see it. However, addressing several situations to implement such an idea is essential. Despite the challenges that come with the creation of the algorithm, the advantages far outweigh the disadvantages and provide new possibilities. Cooperative detection in terms of perception-driven formation control is mutual sensing and observance of an object in the environment and collective analysis of its position, leading robots to reach a common conclusion about the object's location. The methodology proposed in this work is that each drone senses the thing and transfers data to the other drones over a wireless network. The drone that does not see the object would still acquire an estimation of the object's position and take a prescribed place.

The technique implemented in this work is also known as sensor fusion and applies to other industries. The main application of the sensor fusion algorithm is in the automotive industry. Programmers use sensor fusion for a combination of multiple cameras in a SLAM task, simultaneous localization, and mapping. Apart from that, the idea of using redundancy in sensors is quite popular in fault-tolerant scenarios, where the system would work even though a part of it can break, for example, if a sensor fails. The potential of this algorithm is that it would enable each drone to obtain the position of an object, even though it does not observe it. Moreover, having mutual observance, the drones would be able to reach a consensus regarding the

position of the object, enabling them to converge to a single value of the place. The location would further guide drones to their configured site in the environment.

Having a concept of collaborative sensing, it is first necessary to discover the possibilities of implementing the algorithm. The theory exists on dynamics of networks of agents that facilitate understanding of the processes, such as convergence into one value and consensus. However, in simple terms, the agreement to a single value is about averaging the measurements obtained by each drone. This so-called averaging acts as a model for predicting the value of each coordinate the object would take. In collaboration, each drone would sense an object and send its location to the network to cause every other drone to know where the thing is. Since there are multiple measurement cases, processing them in each scenario is necessary. For example, only one drone sees, or only the neighbor sees, both see, or both do not see. This example is for a network of two drones. The scenarios increase exponentially with each added drone. In other words, if there are two drones, there are four scenarios. If there are three drones, there are 2 to power 3, thus eight scenarios. Finally, even though there may be complications during the development of this idea, it is still worth doing it because of the advantages.

3.4.1 Kalman filtering modified for sensor fusion

How to sense collaboratively? The idea behind the mutual discovery of objects is the work of Kalman, the Kalman filter. Since a thing is tracked in some way, there is an obvious need for a measurement mechanism, such as a filter. After a long thought, it was conceived that it is possible to use the filter in the same manner in network and single sensing. The result of applying the Kalman Filter to network dynamics is an extension, which does the exact prediction and correction steps, but for the mutual observations of each drone in the robot network. Each drone obtains multiple measurements, and thus it is necessary to process them by some mechanism. The mechanism is a co-variance of observation. The drone shall get tracked and co-variance values and process them so that the computer uses the most probable value for motion optimization. Therefore, with each obtained measurement comes calculated co-variance of the detection, and each has the same form for each drone. The most straightforward way is to average the points and their co-variance to obtain the single "golden" value. However, the magic is that the computer uses this averaging as a model parameter of the KF, such that the prediction comes from not only the model of a single drone but from models of the network of drones. Eventually, there is the necessity for values to converge to a common observation with a single measurement and, again, a single co-variance.

3.4.2 Consensus approach

According to the Cambridge Dictionary, a consensus is a "generally accepted opinion or decision among a group of people." [38] Regarding robotics, it

is usually called convergence to a single value among agents. In this work, this single value comprises three values: the x,y, and z coordinates of a detected object's position. The way to reach that common position is through communication and filtering. The works of Kristian Hengster-Movric from Czech Technical University in Prague accomplish the model for filtering. [39] It mentions that the future state of a value is a combination or sum of all values inside the network. This combination is the average of all current values in the network. By multiplying the averaged value with transition matrix A, it is possible to obtain a prediction. Mutual measurements of the robots in the network would define the forecast.

From the dynamics of networks, it is apparent that the agents come up with a single value for all the agents, which is termed consensus. It can be assured, given that the transition matrix A has eigenvalues within the unit circle. Let us see if that is true. The consensus here is known as Discrete Time Consensus.

$$x_i(k+1) = u_i(k) \tag{3.10}$$

$$u_i(k) = \frac{1}{1+d_i}(x_i(k) + \sum_j e_{ij}x_j(k)) \tag{3.11}$$

where e_{ij} is adjacency matrix, n is number of robots, x is state

$$e_{ij} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{3.12}$$

In this example, the value of robots is three. The transition matrix, in this case, is the following:

$$x(k+1) = (I+D)^{-1}(I+E)x(k) = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} x(k) \tag{3.13}$$

$$A = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.14}$$

The Eigenvalues of A are 0, 0, and 1. For a discrete-time system to be Lyapunov stable, it is necessary to have Eigenvalues be less than or equal to 1. The system is Lyapunov stable. However, since one of the Eigenvalues is exactly 1, the mathematician cannot consider the system asymptotically stable. Even though it is possible to observe that drones reach a common conclusion about the object's position, probably, methods exist to achieve asymptotic stability through differences from neighbors' values.

For example,

$$u_i(k) = \frac{1}{1+d_i}(\sum_j^3 e_{ij}(x_j(k) - x_i(k)) \tag{3.15}$$

,where $d_i = 2$
Then,

$$x_1 = \frac{1}{3}\begin{bmatrix}0 & 1 & 1\end{bmatrix}(x(k) - x_1(k)) = \frac{1}{3}\begin{bmatrix}0 & 1 & 1\end{bmatrix}\begin{bmatrix}0\\ x_2(k) - x_1(k)\\ x_3(k) - x_1(k)\end{bmatrix} \tag{3.16}$$

$$= \frac{1}{3}(0 + x_2(k) - x_1(k) + x_3(k) - x_1(k)) = \frac{1}{3}(-2x_1(k) + x_2(k) + x_3(k)) \tag{3.17}$$

Generalized,

$$x(k+1) = \frac{1}{3}\begin{bmatrix}-2 & 1 & 1\\ 1 & -2 & 1\\ 1 & 1 & -2\end{bmatrix}x(k) \tag{3.18}$$

$$A = \frac{1}{3}\begin{bmatrix}-2 & 1 & 1\\ 1 & -2 & 1\\ 1 & 1 & -2\end{bmatrix} \tag{3.19}$$

Then, the Eigenvalues of A are -1 -1 and 0. However, since it is a discrete-time system, this control formulation is also Lyapunov but not asymptotically stable. One could use both of the variants. However, since the first notation is more intuitive and stable, the first was chosen in this work.

3.4.3 Cases of detection

There are different scenarios when it comes to detecting an object and sending its position to other agents. There are 2 to power n cases, where n represents the number of robots in the network. In this work, formations with two and three drones were studied. When there are two drones, the possibility of detection is following. Fig.3.3 Tab.3.1

	1ST SCENARIO	**2ND SCENARIO**	**3RD SCENARIO**	**4TH SCENARIO**
1st drone	detects	doesn't detect	detects	doesn't detect
2nd drone	doesn't detect	detects	detects	doesn't detect

Table 3.1: Detection Scenarios

In the case of three drones, the possibility increases twice, and there are eight different scenarios, similar to the table above. However, the main point of this section is what to do when there is, for example, one missing measurement. In this case, for the two-drone configuration, the single drone

sees, and the other does not. What to do is: obtain measurement and co-variance, not take into consideration other measurements and co-variances. There are generally three cases of obtaining processing measurements and co-variances. Tab.3.2

	Single detection	**Double detection**	**Triple detection**
Measurement	$x(k) = x_i(k)$	$x(k) = \frac{1}{2} * \sum_i^2 x_i(k)$	$x(k) = \frac{1}{3} * \sum_i^3 x_i(k)$
Co-variance	$cov(k) = cov_i(k)$	$cov(k) = \frac{1}{2} * \sum_i^2 cov_i(k)$	$cov(k) = \frac{1}{3} * \sum_i^3 cov_i(k)$

Table 3.2: Co-variance derivation of detection scenarios

For the three drones, there are combinations of double detection, variations of single detection, and either triple detection or absence of detection. Finally, with an increasing number of drones, there is a large number of possibilities that are to be programmed. However, most situations reduce to the base cases (single, double, triple). By considering only measurements with detection, it is possible to construct the algorithm and obtain the resulting measured vector as a reference point for motion optimization.

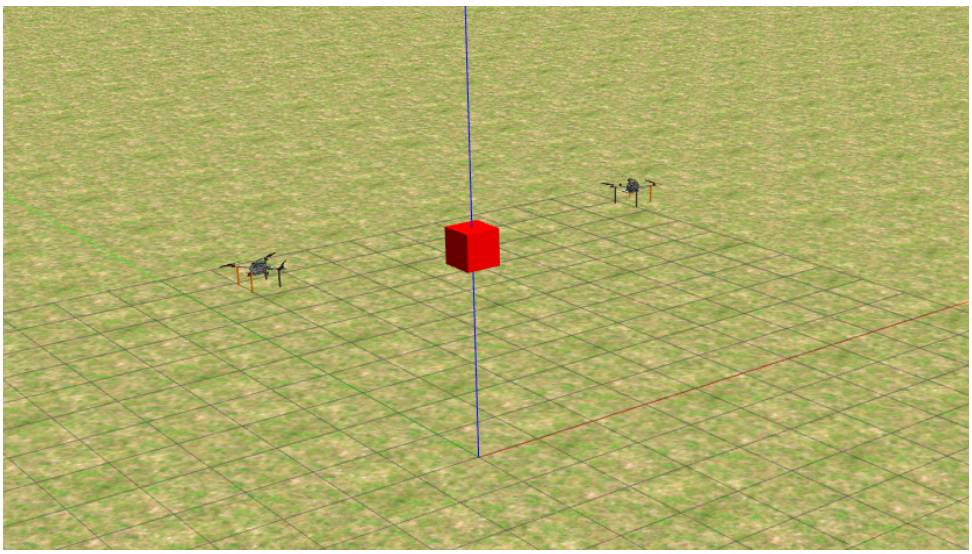


Figure 3.3: Two drones with a box. Converged state, line configuration.

3.5 The algorithm

Considering a multi-robot system, it is possible to utilize multiple robots and improve their sensing through collaboration. The following algorithm was designed to simultaneously measure the object's position from three cameras loaded on the three drones. Multi-robot sensing, also known as sensor fusion,

is the updated version of the Kalman filter. The latter is an algorithm to eliminate noisy measurements from a sequence of upcoming data. The algorithm works in two main steps, which are prediction and correction. In other words, the program predicts the possible future value of a measurement based on a model. Then it corrects the actual value by comparing it to the one obtained through calculation, i.e., from the model. Then, the computer chooses the most probable observation as a final result.

The core of the sensor fusion algorithm is its ability to utilize network information and make predictions based on collaboration. In the case of a multi-robot system, it receives values from other drones and considers them in the prediction step. The model used for prediction in the case of a network of robots is the following. The computer calculates the average of all final outputs from each robot's sensor fusion node. Furthermore, to make a prediction, this average value is transformed, or multiplied, by the transition matrix A:

$$\mathrm{A} = \begin{bmatrix} 1 & 0 & 0 & \partial t & 0 & 0 \\ 0 & 1 & 0 & 0 & \partial t & 0 \\ 0 & 0 & 1 & 0 & 0 & \partial t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

state vector , where x,y,z are coordinates and v_x, v_y, v_z are linear velocities

$$x_{state}^T = \begin{bmatrix} x_{avg} & y_{avg} & z_{avg} & v_{x,avg} & v_{y,avg} & v_{z,avg} \end{bmatrix} \tag{3.21}$$

$$x_{predicted} = A \cdot x_{state} \tag{3.22}$$

The computer averages co-variances of observations from the three drones similarly. After that, it transforms the matrix through A from both sides; furthermore, the addition of matrix R models noise. Algorithm: 1. The first measurement usually needs to be corrected; however, after a few seconds, the algorithm converges to one position value for each drone. Therefore, reaching a consensus, the robots are synchronized.

Algorithm 1: Modified Kalman Filter for Multi-UAV Sensor Fusion[29]

Data: z_h^k - measured human state from human detection block at instant k

$\mathcal{Z}_{k-1}^k$ - measured human co-variance from human detection block

$x_{h_i}^{k-1}$ - sensor fusion output - detected human state, with $i = 1..n$

$\mathcal{O}_{h_i}^{k-1}$ - sensor fusion output - detected human co-variance of observed state

Result: x_h^k - current sensor fusion output - fused detected human state

$\mathcal{O}_h^k$ - current sensor fusion output - fused detected human co-variance

Prediction step

$\hat{x}_h^k = \frac{A}{n} \sum_{i=1}^{n} x_{h_i}^{k-1}$

$\hat{\mathcal{O}}_h^k = A(\frac{\sum_{i=1}^{n} \mathcal{O}_{h_i}^{k-1}}{n})A^T + R$

Correction step

$K = \hat{\mathcal{O}}_h^k C^T (C\hat{\mathcal{O}}_h^k C^T + C\mathcal{Z}^k C^T)^{-1}$

$x_h^k = \hat{x}_h^k + K(z_h^k - C\hat{x}_h^k)$

$\mathcal{O}_h^k = (I - KC)\hat{\mathcal{O}}_h^k$

3.6 Modified perception part

In the second version of the diploma work, the blob detector algorithm was improved by adding the ability to detect objects of multiple colors. The colors are red, blue, violet, and purple. Green, yellow, and orange filters were skipped because the environment where the drones were deployed could contain objects of color in the background of the captured image. For example, the computer would identify grass as a big blob, which should not happen.

The structure of the blob detector is similar to the previous version. It again subscribes to RGB image and Depth camera coming from RealSense

camera. The detection steps are the same. However, the number of color masks has increased. To truly detect each colored object, it was necessary to implement separate color mask identifiers. At the end of the algorithm, they were combined to produce a resulting array of points of locations of each object. Fig.3.4 The initial idea was to store each object's locations in some array for each drone, process the points and synchronize them among the robots. This approach would lead to sending and receiving messages between all drones and point messages. However, a significant improvement can be applied if thinking differently. Instead of processing each point through the Kalman filter, it is possible to collect the observations and compute centroids every time the messages are received. The node that does this work is called SensFuse, which stands for sensor fusion. And to truly stand for sensor fusion, the algorithm should apply the filtering technique. Finally, it was decided to collect centroids, filter them by KF and store them in an array. The final estimation of the centroid, sent to the motion optimizer algorithm, is the average value of centroids of that array, called the "memory" of a robot. This memory enables drones to lose sight of objects. However, still know where the things were or where the previous centroid was.

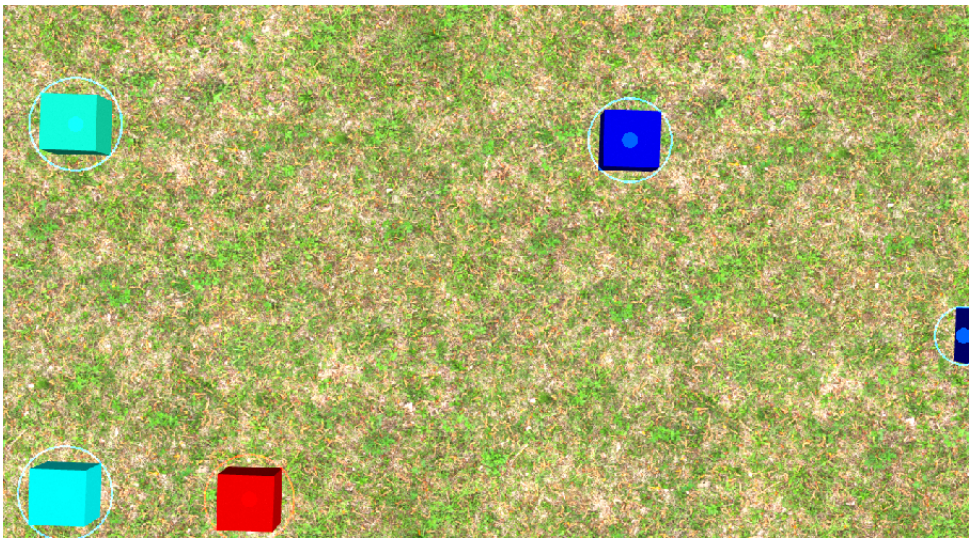


Figure 3.4: Detected objects: red, blue filters. Gazebo

3.7 How to visualize observations with co-variances

The work below is from the second version of formation control(observation of objects below). Co-variance eigenvectors were used to show the co-variance of each drone measurement. Their values, respectively, determine the direction of the ellipsoid in three dimensions and are used for the ellipsoid equation. Eq.3.26

$$a = v_1 \cdot x^2 \tag{3.23}$$

$$b = v_2 \cdot y^2 \tag{3.24}$$

$$c = v_3 \cdot z^2 \tag{3.25}$$

$$\frac{a^2}{k_1 * {\sigma_x}^2} + \frac{b^2}{k_2 * {\sigma_y}^2} + \frac{c^2}{k_3 * {\sigma_z}^2} <= 1 \tag{3.26}$$

where, v_1,v_2,v_3 are eigenvectors of co-variance matrix, x, y, z : are the points, σ_x,σ_y,σ_z: are deviations in three respective axes and k_1,k_2,k_3: are scale coefficients. Fig.3.5

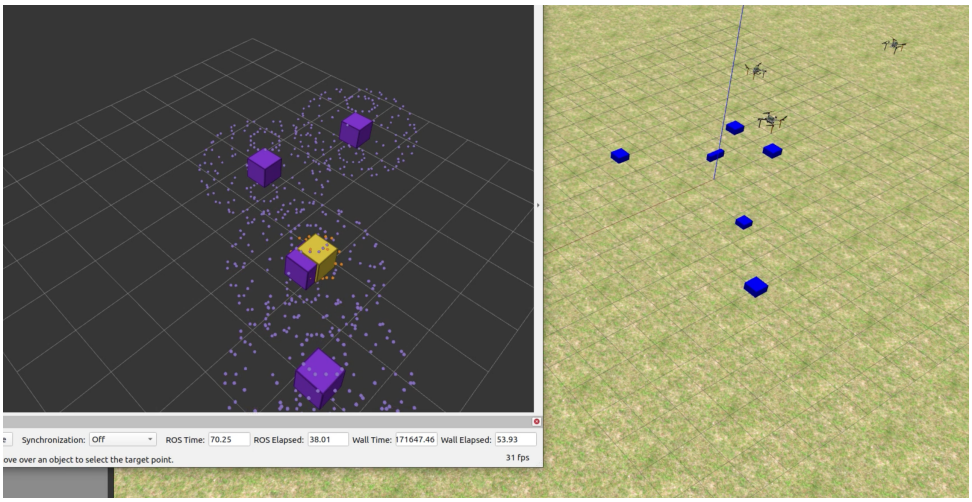


Figure 3.5: Detected objects in violet and their centroid in yellow colors. Gazebo.

3.8 Discussion

The multi-robot collaboration in sensing is a novel approach to the old Kalman filter algorithm. The main difference between the original and modified versions is that the latter considers network information and takes an average of the values. The theoretical background obtained at Czech Technical University in Dynamics and Control of Networks paved the way for this accomplishment.[39]

3.9 Conclusion

In conclusion, tuning process noise and measurement noise co-variances is vital, which are usually diagonal matrices. The user can define the strength of each filtering side: measurement, and model; through diagonal coefficients of the two matrices.

Chapter 4

Cooperative control

4.1 Motion Control Optimisation

A control algorithm is a methodology to make drones fly to a particular position or in a specific way. The Multi-robot Systems group from Czech Technical University has already implemented the low-level control of drone motors. They used MPC to fly the robot. The algorithm proposed in this work acts as a higher-level layer that sends the computed position to the drone's low-level control layer as a reference point message. With this layer, the drone would fly smoothly and reach the necessary destination. This layer is an application to the control part of the system in this work.

There are several reasons for the necessity to use this layer of control. First, it is essential to tell the drone to fly to a point relative to the detected object or the leader. Secondly, the proposed algorithm reduces drone motion vibrations and makes it glide smoothly. Finally, the drone utilizes the quality of each measurement, that is, co-variance. (the first version of algorithm) Adding co-variance into control law is only possible by introducing the higher layer because the low-level control takes only a reference, *go_to*, or a relative direction point to fly to.

The layer takes the result of the sensor fusion processing part of the program and outputs an optimal reference point to the low-level control part of the system. It does so by solving an optimization problem, that is, by trying to find a minimum to a prescribed cost function. The solution of optimization problems is based on gradient descent, or moving in the opposite direction to the current value of gradient, which directs to the function's highest value. Therefore, minimizing a cost function or reaching the lowest point in a region is necessary for an optimization task. In this work, convex optimization is considered. Thinking of the mathematical function can be tremendously tricky because the dimensions of the cost function increase to 4 for a 3D point optimization. Since it is difficult to imagine the cost function, only cases with one or two dimensions will be explained.

The task is to find the optimal coordinate to fly to, such that there is a distance to an observed or detected position of another drone. Fig.4.1 The method considers the quadratic distance to the current measurement of a leader's location and the drone's position. The reason for using the quadratic

function is that it is the best choice for gradient-like optimization, and it converges fast. The consideration is that, by having obtained a coordinate of the detected object, such as x_{obj}, the computer can consider the drone's current position as x_{cur}. By squaring the distance between the two coordinates, it is possible to create a cost function, which would be minimal when the two coordinates are equal to each other and maximal when the distance increases. To incorporate the offset of the drone relative to the detected object, it is necessary to add this offset to the thing's coordinate. For example, in the following cost function:

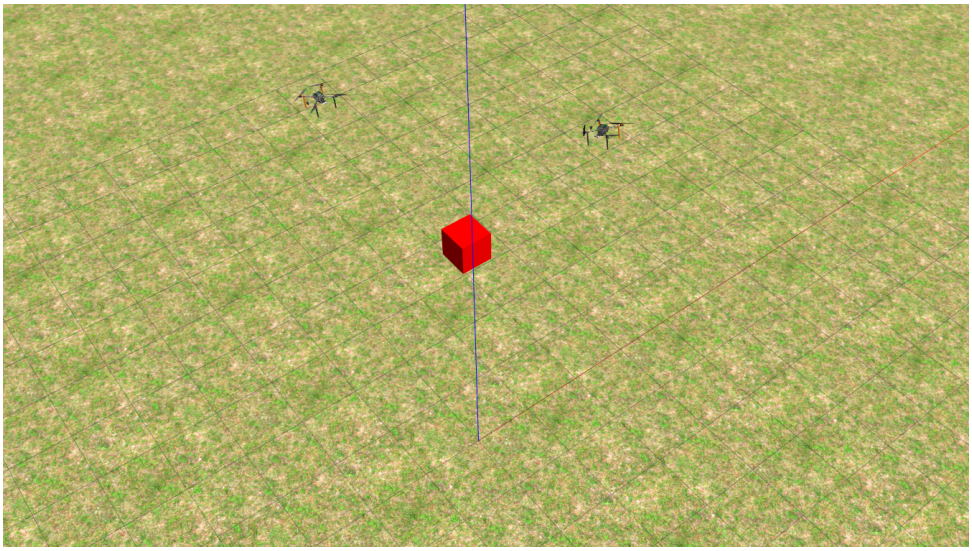


Figure 4.1: Two drones with a moving box.

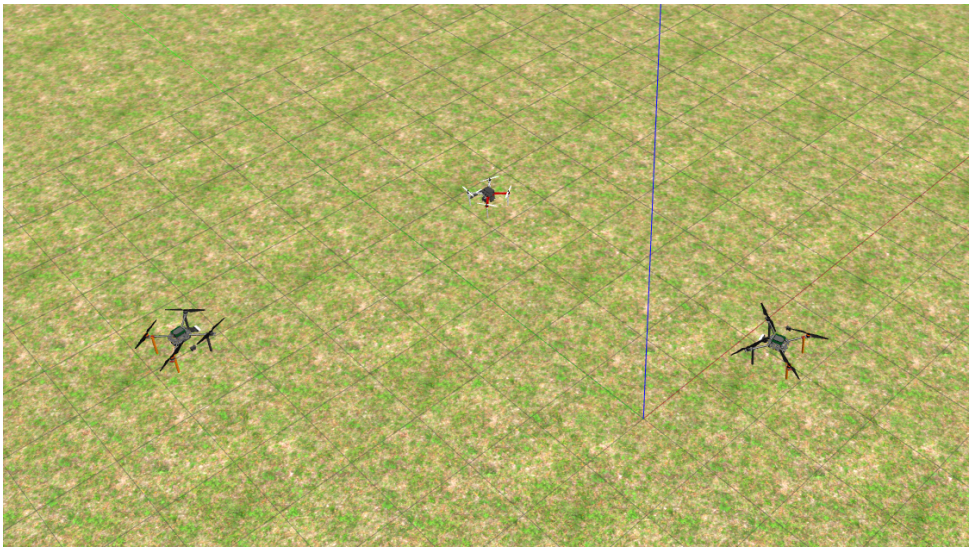


Figure 4.2: Two drones with the leader drone (white-red), converged

$$J_x = (x_{cur} - (x_{obj} + offset_x))^2 \tag{4.1}$$

Thus, the algorithm would try to minimize the value of J_x, leading to the current drone's position being equal to the observed drone's position with an offset. Fig.4.2

One of the benefits of this algorithm is that by taking a cost function, the programmer can tell the drone exactly where to move and does not need to repeat that command because the drone would always understand the relation to the detected object in terms of the position. Another benefit is that by computing fast and with a short step of the algorithm's computation, it is possible to give the drone the ability, to follow small goals that would lead him eventually to the final destination. This ability gives rise to the possibility of goal-driven behavior in robotics. Furthermore, adding other terms to the cost function is possible, such that the computer minimizes the drone's vibration and the drone's estimation co-variances. Finally, it is possible to add coefficients to each term in the cost function to tune the controller for the best performance.

4.1.1 Multi-dimensional composition of cost-function

The approach implemented in this work is that the drone would move to a prescribed destination, whatever path it chooses as optimal. The classical robot control method uses path planning in the environment by considering obstacles and map availability. However, in reality, especially when the drones are concerned, there is usually no precise defined location of an object on the map. Therefore, to follow a thing other than classical methods have to be used, such as goal-driven behavior. The idea is that the drone would move autonomously like a ball on a valley, where the maximum would be a position far away from the goal, while the minimum of the valley would represent the desired position for a drone to reach. Reaching a goal can be simplified when considering only one dimension. Therefore, the desired coordinate becomes, for example, x_{goal}, which is $x_{goal} = x_{obj} + offset_x$, the value of the detected object plus the prescribed distance from the thing.

To solve this problem, imagine a convex parabola-type cost function with a tip of it (minimal value) located strictly at x_{goal}. Therefore, even a simple gradient descent algorithm will always lead the drone's x_{drone} to converge to x_{goal}. In two dimensional case, the cost function takes one more dimension. Thus 2+1 = 3-dimensional function. The simplified version of the cost function, without co-variances and vibration minimization, has the shape of a paraboloid with the tip coordinates located strictly at (x_{goal}, y_{goal}). It is possible to imagine this cost function, which differs from a 3-coordinates optimization. It is impossible to imagine a 3+1 = 4-dimensional function.

There is an opinion that gradient descent is not beneficial because of the sharpness of the robot's turns. This opinion is usually because of the

application of the control program, such as autonomous driving. In that scenario, the car cannot take sharp corners due to its physical configuration. However, in the drone system, there are no such restrictions. Although it is possible to smooth the path, it will require planning and knowledge of the location of every point around the drone, which is computationally costly. Therefore, there exists a better version of gradient descent applied to learning a neural network in some other project. Using multi-dimensional optimization has significant benefits, such as tight control, which does not require planning time. As the drone observes the object, it would, in a few milliseconds, start to fly to the relevant destination, which is only sometimes possible when using classical planning, which requires time. Some algorithms like A* would take significant time when planning in 3D. Another benefit is that it is possible to configure the cost function almost anyhow and add additional terms, which will be described later.

Finally, since the 4-dimensional case is hard to imagine, to be more assured, it was decided to use a 3d paraboloid for the basis of the cost function. However, the paraboloid is insufficient since it does not capture the repulsive features of the proposed cost functions. The implementation of the current state cost function is described in the latest modifications section.

4.1.2 iRPROP+

There are several reasons for using the state-of-the-art version of the cost minimization algorithm. There are significant benefits, such as avoiding getting stuck at a point in the cost function, fast convergence, and step-wise convergence towards the goal. iRPROP+ is a version of the RPROP algorithm modified for better performance. [40] RPROP stands for resilient back-propagation. The algorithm performs an adaptation of the weight step based on the value of the local gradient. The computer performs optimization based on only the sign of the partial derivative. Initially, the RPROP algorithm was invented by Martin Riedmiller and Heinrich Braun in 1992.[28] The basic idea is to update the weight depending upon a given situation, with one of the two update ways. The update ways are either adding current $\Delta w_i(t)$ or subtracting previous $\Delta w_i(t-1)$. The two cases where the first update is used are when the product of current and previous gradients is greater than 0 or it is precisely 0. The contrary case, when the product of gradients is less than 0, involves back-stepping by delta if the cost function has increased and sets the current gradient value to zero in the other case. The choice of $\Delta w_i(t)$ is also dependent on gradients: in the first two cases, it is minus sign of the product of the current gradient value with current $\Delta_i(t)$ and when the product of gradients is less than zero, $\Delta w_i(t)$ is not updated. $\Delta w_i(t)$ also depends on $\Delta_i(t)$, which is updated when the product of gradients is either greater than or less than 0, not equal. The value for the two cases are respective: minimal of $\Delta_i(t-1)$ * η^+ and Δ_{max}, maximal of $\Delta_i(t-1)$ * η^- and Δ_{min}, where η^+ and η^- are coefficients that are set to 1.2 and 0.5 respectively. The computer calculates all of the above steps in a loop that parses through a range of coordinates, which is 4 (XYZ, yaw) for the case of

this work. Algorithm 2

Algorithm 2: $iRPROP + algorithm$[40]

Data: $w_i, E^{(t-1)}, E^{(t)}, \Delta_i^{(t)}, \Delta w_i^{(t)}$

Result: $w_i^{(t+1)}$

for $i = 0;\ i < 4;\ i = i + 1$ **do**

  if $(\frac{\partial E}{\partial w_i})^{(t-1)} \cdot (\frac{\partial E}{\partial w_i})^{(t)} > 0$ **then**

    $\Delta_i^{(t)} := min(\Delta_i^{(t-1)} \cdot \eta^+, \Delta_{max})$

    $\Delta w_i^{(t)} := -sign(\frac{\partial E}{\partial w_i}^{(t)}) \cdot \Delta_i^{(t)}$

    $w_i^{(t+1)} := \mathrm{w}_i^{(t)} + \Delta w_i^{(t)}$

  else

    if $(\frac{\partial E}{\partial w_i})^{(t-1)} \cdot (\frac{\partial E}{\partial w_i})^{(t)} < 0$ **then**

      $\Delta_i^{(t)} := max(\Delta_i^{(t-1)} \cdot \eta^-, \Delta_{min})$

      $w_i^{(t+1)} := w_i^{(t)}$ - $\Delta w_i^{(t-1)}$

      $(\frac{\partial E}{\partial w_i})^{(t)} := 0$

    else

      if $(\frac{\partial E}{\partial w_i})^{(t-1)} \cdot (\frac{\partial E}{\partial w_i})^{(t)} = 0$ **then**

        $\Delta w_i^{(t)} := -sign(\frac{\partial E}{\partial w_i}^{(t)}) \cdot \Delta_i^{(t)}$

        $w_i^{(t+1)} := w_i^{(t)} + \Delta w_i^{(t)}$

      end

    end

  end

end

The benefits of the algorithm are clear because, from the algorithm, it is possible to observe that there is only one check and update per each coordinate. Therefore, the algorithm's performance is 4*2 checks is about eight operations to make an update. Considering the maximal possibility of computation steps, the number is four operations per 4 coordinates, hence 16 operations.

Considering a few steps, for example, 10, the algorithm would find a close position to move to in 160 operations, which is relatively fast. Although the algorithm is complicated, it is a modification of the gradient descent idea that uses different delta values for other cases depending upon signs of the current and previous gradients.

4.1.3 Cost function

The cost function for guiding the robot toward a goal destination is a paraboloid and the inverse of a paraboloid for repulsing. This choice makes convergence harmonic and smooth. Moreover, it is fast. The convergence also depends upon a step parameter in the gradient descent algorithm. For every coordinate, x y z yaw, there is a joint 3D objective function.

$$(x - (x_{obj} + offset_x))^2 \tag{4.2}$$

Collision avoidance

Taking the inverse function of the quadratic parabola makes it possible to avoid collisions. The function appears as

$$\frac{1}{(x - obstacle_i)^2} \tag{4.3}$$

To eliminate the division-by-zero problem,a bit of offset was added to the function. Consequently, it takes the following formulation:

$$\frac{1}{(x - obstacle_i)^2 + 0.001} \tag{4.4}$$

The resulting cost function combines goal parabola terms and other obstacle inverse parabola terms. The collection of inverse parabola terms is their sum.

$$\sum_i \frac{1}{(x - obstacle_i)^2 + 0.001} \tag{4.5}$$

4.1.4 Additional terms of cost functions

The cost function is already pre-configured for a robot to reach its destination. However, the question remains how to do that so motion is smooth? From tests, it is possible to observe that it is necessary to reduce something to obtain a more desirable behavior. The reduction is inside the difference of steps of a drone. In other words, it is necessary to tell the drone to take small steps at the time, that is, to reduce the distance between the current and previous positions' coordinates. By taking the quadratic function of

$$J_i^{additional} = \alpha * (x_i(k) - x_i(k-1))^2 \tag{4.6}$$

, where α is a control parameter, it is possible to reduce unwanted vibrations from the robots' motion.

Furthermore, it is possible to improve the algorithm by adding another term to the cost function, which will incorporate the information gained from state and object detection estimations. This information is the co-variance matrix. Because the dimension of the cost function in one coordinate is equal to 1, it is necessary to represent co-variance as a single number. To do that, it was decided that the determinant of the co-variance matrix would be a computation tool. In other words, the drone would try to reduce the determinant of the state co-variance matrix and the determinant of observation of the object co-variance matrix.

4.1.5 Tests

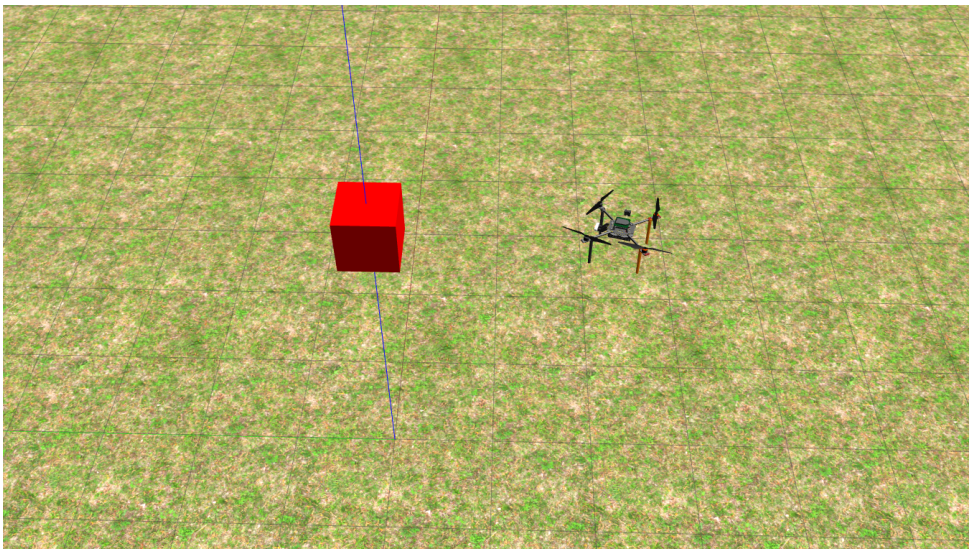


Figure 4.3: Single drone and a box.

Single, two, and three drone formations were tested in simulation. The drone correctly defines the position of the leader drone and moves to the respective coordinate. Fig.4.3

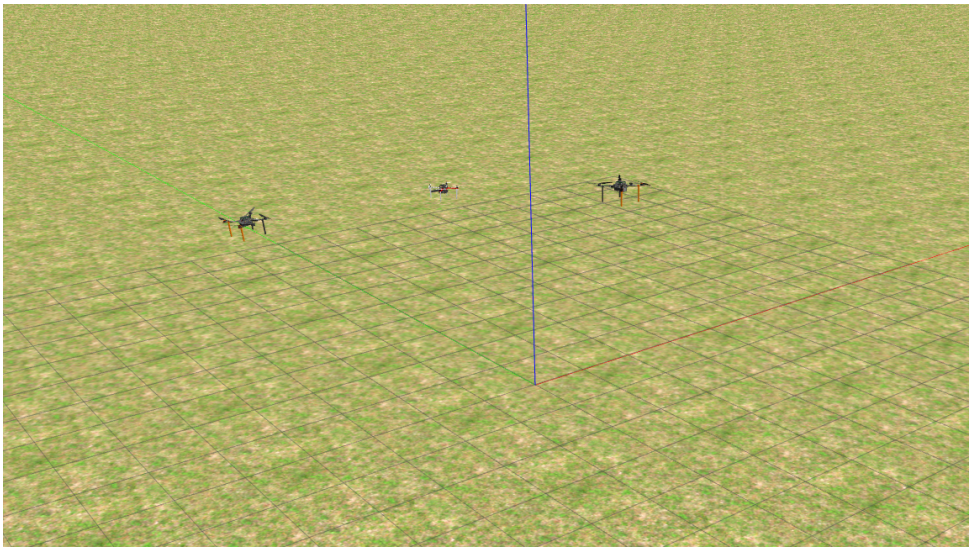


Figure 4.4: Two followers and a leader.

A similar situation is for two drone configurations. it was found that the position when there are 90 degrees between drones is the best for observing the target because of the color configuration of the leader drone and because drones can observe motion in two dimensions. Fig.4.4

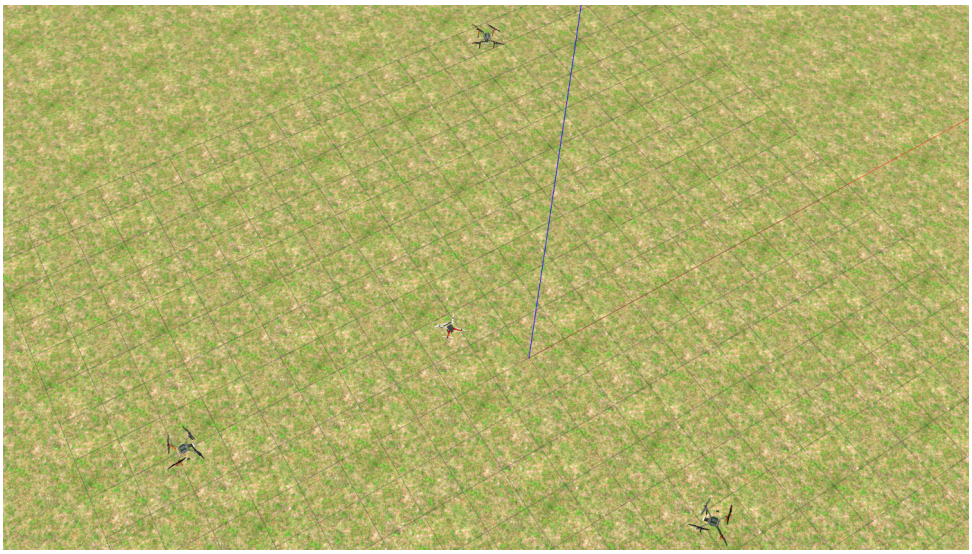


Figure 4.5: Three followers and a leader.

Three drone configurations were also tested, and the drones were in a circle with a radius of around 6 meters. The drones correctly detect the object and converge their positions to the ring around the leader drone. Fig.4.5

4.2 Time Synchronization

To manage two or more messages coming to the node, a time synchronization from message_filters library can be used. The synchronizer takes the configuration of the messages. For example, if both images are of type < Image>, the configuration would be <Image,Image>. The node responds to the upcoming messages with a callback, where every procedure on the message can be proposed and implemented. To test that time synchronization is working, two simple subscribers to a topic "/camera/image" were created, and the callback function was tested with a simple ROS_INFO command. After successfully obtaining of checking message in the console, it was decided to subscribe to a different topic from one subscriber, mainly to "/computations/goal_point", which sends the coordinates of a point. The process needs to be revised. The most crucial problem was that each message needed to be with a time stamp. In other words, the headers of messages have to contain a time variable, or stamp, inside them, which is essential for time synchronization.

4.3 Novel approach without time synchronization

During preparation for real-world experiments, it was found that time synchronization can cause problems and become a single point of failure in multi-robot systems. In other words, one drone can publish messages later than the other because of being started later, which will cause synchronization callback not to trigger. The possible solution to this problem is to treat each topic separately with a dedicated callback, which updates the node's global variables. However, a programmer must be careful and introduce the usage of a mutex to protect the part of the code that modifies a global variable.

4.4 Formation control applied to multi-drone observation

Introduction

After the first experiment's results, it was realized that the algorithm should be improved. One of the improvements is that the algorithm should trigger the drone's motion by a command outside the node. This external command would allow for movement, a request at a particular time. Another improvement is that it is possible to use ROS nodelets to improve data transfer between nodes because nodelet nodes act as plugins to a namespace and can share data without needing an external connection. Finally, synchronization issues were experienced with the previous version of the algorithm and improvement of synchronization between nodes was important. Therefore, it was decided to simplify and split two nodes algorithm into three nodes algorithms, but continuing in a decentralized manner of the system. The whole algorithm of the cost function was reviewed again, and multiple scenarios have been tested

and depicted in graphs to see how does algorithm minimize the cost function and how does the cost function look like. The drawback is that some parts of the cost function, such as motion smoothing, cannot be viewed in a graph. However, we can depict the ultimate goal and obstacle avoidance. Fig.4.6

Modified Control Part

In the new version of the formation control algorithm, the whole cost function estimation and minimum selection system were analyzed again and many improvements were found. First and foremost, it is not possible to visualize the objective function in the four dimensions, so the number was reduced to 3, thus the x and y coordinates and the cost. Because of this, it became possible to test whether the shape of the cost functions is right. The significant improvement was using different functions to represent attractive and repulsive forces inside the algorithm. Through iteration, it was found that it is possible to use the Gaussian normal distributions better than simple or naive parabola functions. The only difference between attractive and repulsive parts of the cost function is the sign of the function, positive for repulsion and negative for attraction. Another improvement was to use a widely-spread parabola function that would lead a drone to move to a goal from all possible positions in space. The cost function is composed of several terms:

H - cost function height
J_{vib} - vibration cost
J_{attr} - parabolic attraction
J_{obst1} - obstacles cost 1
J_{obst2} - obstacles cost 2
J_{goal} - goal cost
g_{depth} - goal function depth
$width$ - width(σ^2) of Gaussian function

$$J_{vib} = \frac{1}{2} * (x - x_{prev})^2 + \frac{1}{2} * (y - y_{prev})^2 \tag{4.7}$$

$$J_{attr} = \frac{1}{2} * (x - x_{goal})^2 + \frac{1}{2} * (y - y_{goal})^2 \tag{4.8}$$

$$J_{obst1} = e^{(\frac{(y-y_{obs})^2}{width}} * e^{\frac{(x-x_{obs_x})^2}{width}} \tag{4.9}$$

$$J_{obst2} = e^{(\frac{(y-y_{obs_2y})^2}{width}} * e^{(\frac{(x-x_{obs_2x})^2}{width}} \tag{4.10}$$

$$J_{goal} = -g_{depth} * e^{-\frac{(x-x_{goal})^2}{width}} * e^{-\frac{(y-y_{goal})^2}{width}} \tag{4.11}$$

$$COST = H + J_{vib} + J_{attr} + H * (J_{obst1} + J_{obst2}) + J_{goal} \tag{4.12}$$

A particular testing environment was prepared to estimate the minimum of the cost function and visualize the cost function itself. Python and matplotlib were used to depict the cost function and put points calculated by iPROP+

on top of the cost function. The test results revealed huge problems with the previous way of generating gradients and the cost function. Therefore, it was necessary to rewrite the whole part of gradient calculation, which was previously naive and was to compute gradient based on cost function change in the step directions. Finally, it was possible to derive the gradients mathematically:

$$J_{vib_{grad_x}} = x - x_{prev} \tag{4.13}$$

$$J_{attr_{grad_x}} = x - goal_x \tag{4.14}$$

$$J_{obst1_{grad_x}} = e^{-\frac{(y-obs_y)^2}{width}} * e^{-\frac{(x-obs_x)^2}{width}} * (-2 * \frac{x-obs_x}{width}) \tag{4.15}$$

$$J_{obst2_{grad_x}} = e^{-\frac{(y-obs_2y)^2}{width}} * e^{-\frac{(x-obs_2x)^2}{width}} * (-2 * \frac{x-obs_2x}{width}) \tag{4.16}$$

$$J_{goal_{grad_x}} = -g_{depth} * e^{-\frac{(x-x_{goal})^2}{width}} * e^{-\frac{(y-y_{goal})^2}{width}} * (-2 * \frac{x-goal_x}{width}) \tag{4.17}$$

$$GRADIENT_x = J_{vib_{grad_x}} + J_{attr_{grad_x}} + H * J_{obst1_{grad_x}} + H * J_{obst2_{grad_x}} + g_{depth} * J_{goal_{grad_x}} \tag{4.18}$$

$$J_{vib_{grad_y}} = y - y_{prev} \tag{4.19}$$

$$J_{attr_{grad_y}} = y - goal_y \tag{4.20}$$

$$J_{obst1_{grad_y}} = e^{-\frac{(y-obs_y)^2}{width}} * e^{-\frac{(x-obs_x)^2}{width}} * (-2 * \frac{y-obs_y}{width}) \tag{4.21}$$

$$J_{obst2_{grad_y}} = e^{-\frac{(y-obs_2y)^2}{width}} * e^{-\frac{(x-obs_2x)^2}{width}} * (-2 * \frac{y-obs_2y}{width}) \tag{4.22}$$

$$J_{goal_{grad_y}} = -g_{depth} * e^{-\frac{(x-x_{goal})^2}{width}} * e^{-\frac{(y-y_{goal})^2}{width}} * (-2 * \frac{y-goal_y}{width}) \tag{4.23}$$

$$GRADIENT_y = J_{vib_{grad_y}} + J_{attr_{grad_y}} + H * J_{obst1_{grad_y}} + H * J_{obst2_{grad_y}} + g_{depth} * J_{goal_{grad_y}} \tag{4.24}$$

, which leads to the correct estimation of the minimum of the objective function.

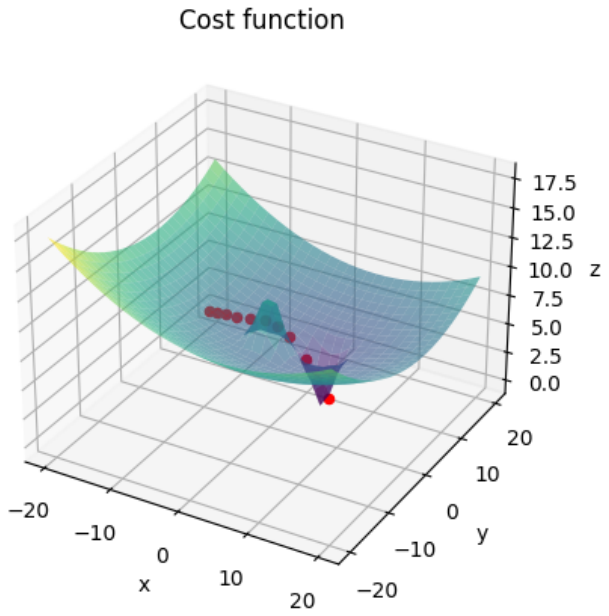


Figure 4.6: 3D objective function with a high peak located at an obstacle and a low extremum located at the goal, red dots are steps

A separate node was created and now, compared to the previous implementation, the node subscribes only to four sources: odom1, odom2, odom3, and goal. The goal point comes from the SensFuse node, which calculates the observed centroid. However, this time the algorithm does not depend wholly on the goal; it can move around a different average centroid in "searching" mode and consider the goal when moving in "form the formation" mode.

Controller interface for control of drones

To solve the problem of drones receiving commands directly from the run-time of the algorithm, which leads to the spontaneous, unpredictable motion of the robots, it was decided to add one more node to the system that would control drones such that they are not fully autonomous and a human operator can stop them. The benefits are that the drones are now deterministic, controllable by humans, and have different modes of operation. The mode of operations is an idea to split the process into two parts: searching and forming formation. Previously, without the modes, the drones were given the programs and were supposed to do everything at once, that is, to collect the points from the camera and form formation. Consequently, there was spontaneous motion and, of course, loss of information when drones did not see the objects. With this idea of modes, a program was created to toggle the mode of drones.

The first mode is searching; in this mode, the drones are supposed to move in formation around a common point and search for some time for objects on the ground. It is possible to operate drones in any direction by wasd buttons, and autonomous circle following is also feasible. During self-sustaining circle following, drones fly around the recorded centroid to stay in formation and move in a circular trajectory. While flying, they would collect points, send them to each other, and compute and store centroids. Finally, they have an average position of centroid inside their memory, which is mutual to each of the drones because of collected messages. In the second mode, forming formation, drones stop moving around a common point and move to the centroid, which is in their memory. They take a geometrical shape around the last common centroid. How to find an initial point for drones to search around? To answer this question, a button was added to record all drones' current average position. Having obtained a centroid of current positions, stopping the recording, selecting mode, and moving drones are possible.

Finally, there are additional buttons that tell drones to move or to stop. These are, again, toggle switches that enable some parts of the code in the motion optimizer node. In conclusion, this separation of actions led to significant improvements in system control and predictability. Now the user can interact with the system and decide which of the two modes to select. The whole interface is of a command-line type. The resulting table of commands: Tab.4.1

Button	Command
1	Toggle on/off motion of UAV 1
2	Toggle on/off motion of UAV 2
3	Toggle on/off motion of UAV 3
r	Toggle on/off recording of centroid between UAVs' positions
c	Toggle manual control/automatic circle following
w	Move formation +1 meter along x-axis
s	Move formation -1 meter along x-axis
d	Move formation +1 meter along y-axis
a	Move formation -1 meter along y-axis
m	Switch mode: searching/form formation
9	Decrease radius of the big formation circle
0	Increase radius of the big formation circle
i	Toggle on/off ignore observations mode
q	Send command to stop the three nodes

Table 4.1: Commands list

4.4.1 Formation reshaping

Formation reshaping is an algorithm to direct drones from their initial position toward the closest place of the geometrical formation. This problem is known as the task allocation problem, and there are two ways to solve it. The first is the greedy approach, and the second is the Hungarian. In the greedy

approach, a metric function is used as an objective function to allocate drones to the position. It is possible to plan paths toward the goal and select the shortest route. However, in this work's scenario, it was impossible to obtain the routes. Therefore, another criterion was applied, the distance between the drone and the allocated position in the formation. The other approach involves the Hungarian algorithm, but it was skipped because of its complexity and computational cost.

Finally, the greedy approach works well in the system, provided that drones share their desired position, such as position 0, position 1, or position 2, with other drones. The broadcast values notify each drone of the already allocated spots. Fig.5.11

4.4.2 Formation breaking mode

Formation breaking is the ability to separate drones from formation to stop at a found object or to prevent the group of drones from moving towards a false positive measurement. Once a drone sees a thing, it can stop and start tracking the object while other robots continue to move in formation. It is crucial to clarify the logical sequence of operations. The drone should separate when one drone detects a false positive measurement. Consequently, the logic is that when the drone sees, it should continue tracking, and when the user is sure, he can shape the whole formation around the observation. In case of false positives, the drone is supposed to miss further observations of objects and return to searching mode. Fig.4.7 Video.[41]

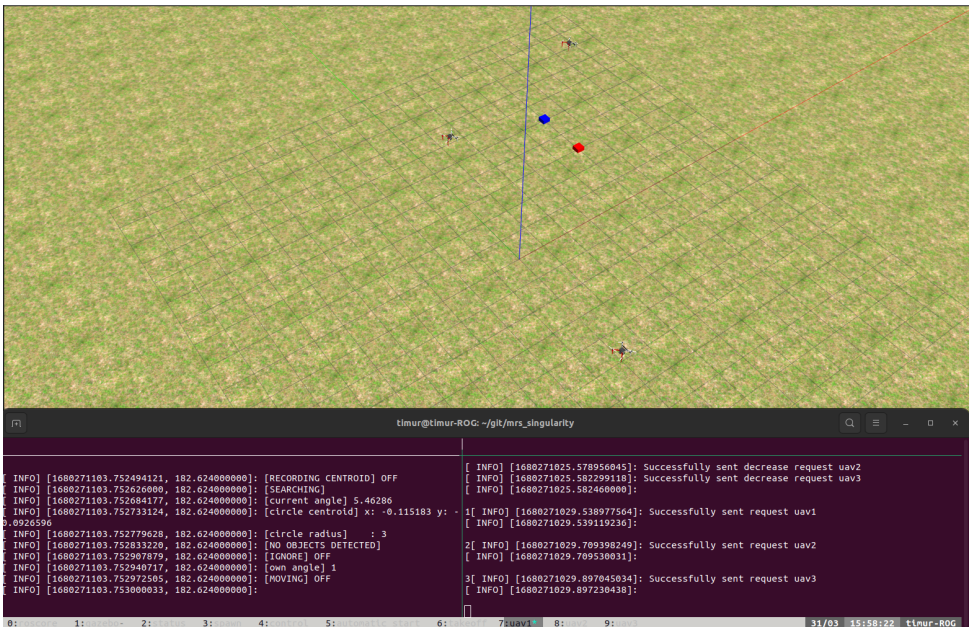


Figure 4.7: One drone leaves formation to track found objects. Gazebo

The main issue with the following approach was encountered when switching from form formation to searching mode. The drones would not leave their

observations and would continue to track them, leading further to the collision of drones. In the simulation, drones start to fly above each other and eventually land down. A workaround solution to create an additional button was proposed that would tell the drones to return to searching mode even though they detect objects. This button "ignores" the observations. Initially, drones ignore things, and when manually told to toggle "ignore" to off, they stop above the items. After that, a user can turn on the ignore mode and return to searching mode. The algorithm requires an explanation of the commander and how it works. In conclusion, formation breaking is achievable. However, problematic situations should be addressed.

4.5 Conclusion

With a motion optimization level of control, the drones move smoothly and, as expected, converge to the goal position. They do so by minimizing an optimization criterion: cost function value of each coordinate: x,y. iRPROP+ is one of the fastest possible algorithms for solving an optimization problem.

Chapter 5

Results

5.1 Simulation

5.1.1 Formation-control for power tower

The technique was to guide robots toward their master, a human. The drones first recognize humans by color in an environment and do this collaboratively. After that, they strive to reach the destination point programmed for them optimally. Finally, the robots track the motion of a human and preserve their position concerning him. An orange blob detector was used as a perception algorithm that helps identify the human worker. The optimal formation node was programmed with a non-linear solver using an improved version of the Resilient Back-propagation algorithm, iRPROP+. Finally, there is the introduction of a sensor fusion algorithm. Sensor fusion stands for observing each measurement of each robot and taking the average in the prediction step of each robot's Kalman filter. This node helps the robot identify a worker's precise position in the orange jacket. Fig.5.1 Developed software: [42]

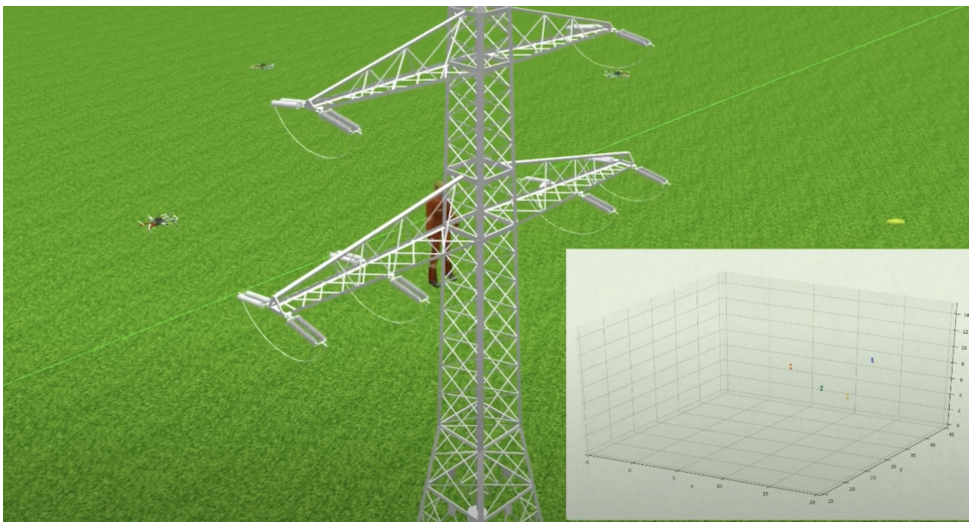


Figure 5.1: Following an electrician with a drone formation. Gazebo

System configuration

Three drones were spawned inside the Gazebo environment, equipped with RealSense RGBD cameras faced front-wise. Each drone has a GPS navigation system.

Experiment results

In this simulation, three UAVs had to fly in a formation while tracking a human worker during his work on a power line tower. The human uses an IPE in orange color. The formation must be in such a manner that the global covariance of observation must be minimized as well as the error between the desired and the real distance between each UAV and the human. For this case, we used the F550 UAV platform assembled by our group and that had a model created in Gazebo. [29]

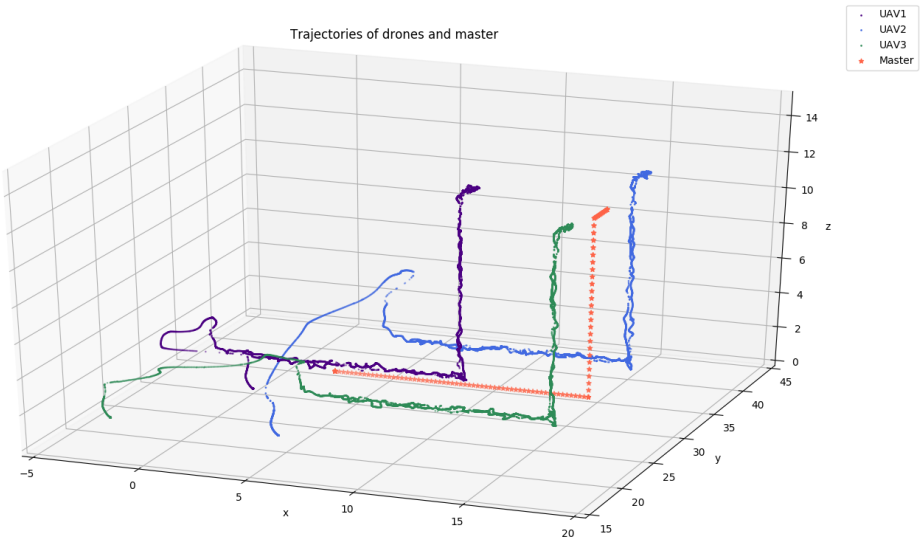


Figure 5.2: Trajectory performed by the UAV formation and the detected worker.

The final trajectory performed by all three UAVs and the worker is presented in Figure 5.2. The trajectory is smooth and the formation, although flexible, maintains a fixed form that minimizes the covariance of observation. This error minimization is also seen in figure 5.3 that presents the mean square error of each UAV with respect to the cost function minimization. [29]

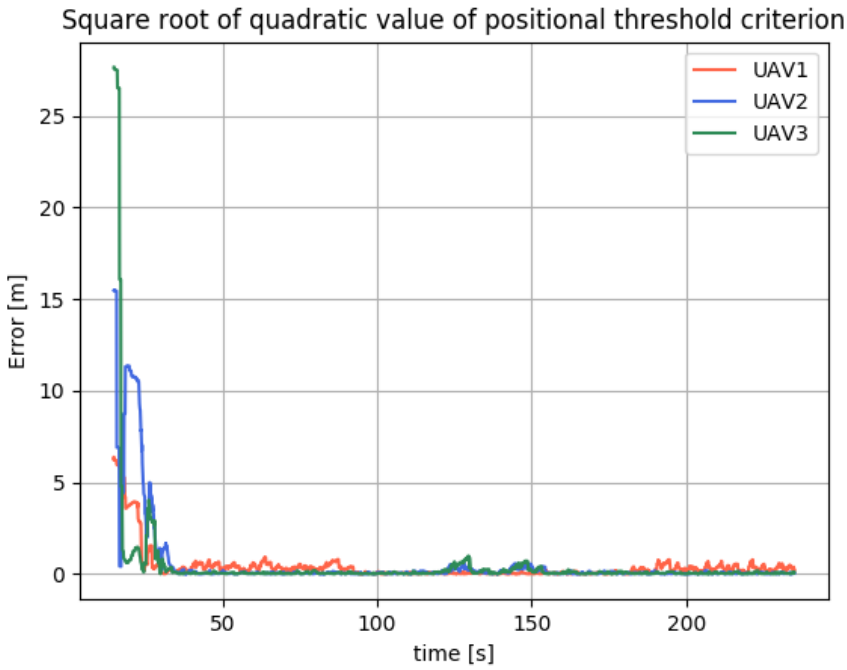


Figure 5.3: Mean square error measurement of all three UAVs.

5.1.2 Formation-control applied to object detection

In the simulation environment, cubes of blue and red color were placed on the grass plane world. The cubes are to be detected by drones, forming a geometrical shape around them. Fig.5.4 Developed software: [43]

System configuration

Again, the three drones were spawned inside the Gazebo environment, equipped with RealSense RGBD cameras facing, this time, downwards. Each drone has a GPS navigation system.

Experiment results

All drones detect and store the found locations of objects below them. As expected, they record their positions in the world, find the average and create a searching circle around that center. When they see things, drones store the things' centroid, and when switched to "form formation" mode, they fly toward the objects.

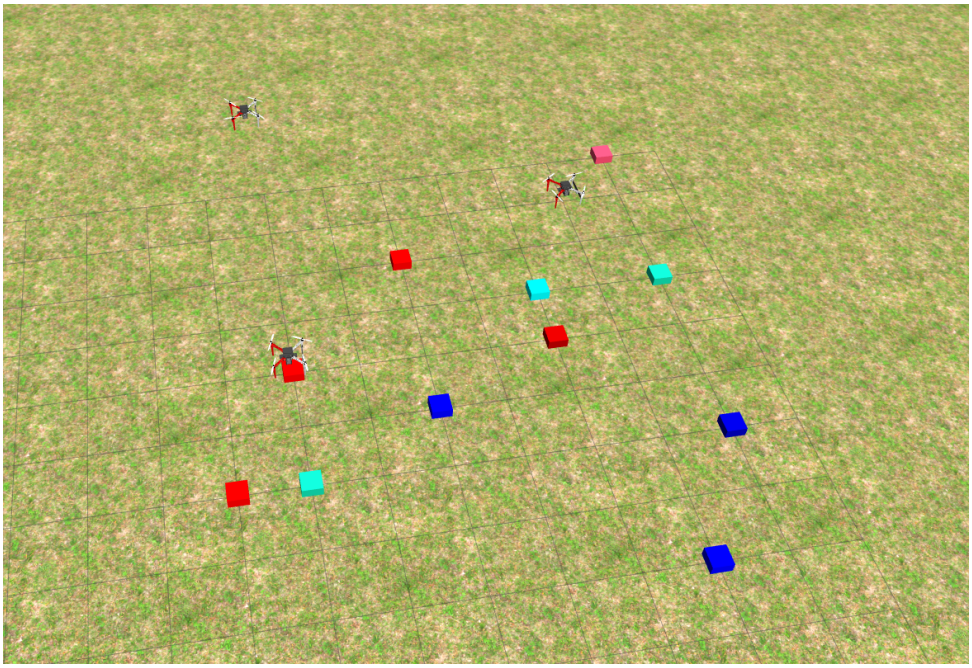


Figure 5.4: Observing multiple objects within a formation. Gazebo

Paths, obtained from the simulated experiment

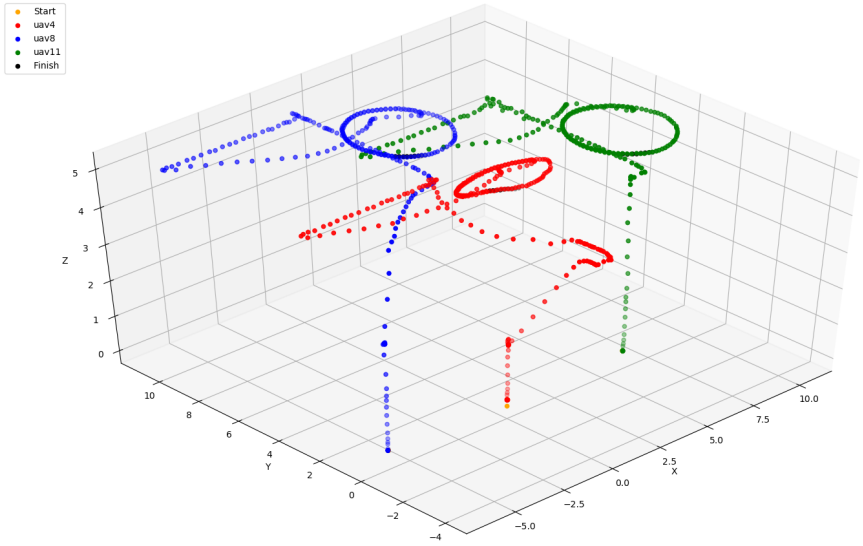


Figure 5.5: 3D drone paths during the simulated experiment. Gazebo

The plot (Fig.5.5) depicts the motion of drones during the entire simulated experiment. There were several phases during the flight, such as shaping into formation, manually controlled movement around the area, returning to found objects, and moving in a circle around the newly recorded centroid.

The orange points indicate the starting location of the drones. Black points represent the finish position. Although one skewed, all drones depicted the circle during automatic mode testing. Because of the similarities, in terms of shapes, between drones' paths, it is possible to conclude that the drones were flying in formation.

Detections

The sensor's measurements are depicted with the paths. Fig.5.6. It is possible to see that there is noise in the observations, such that the observed points are along the paths. Therefore, it can be concluded that there was a false positive measurement.

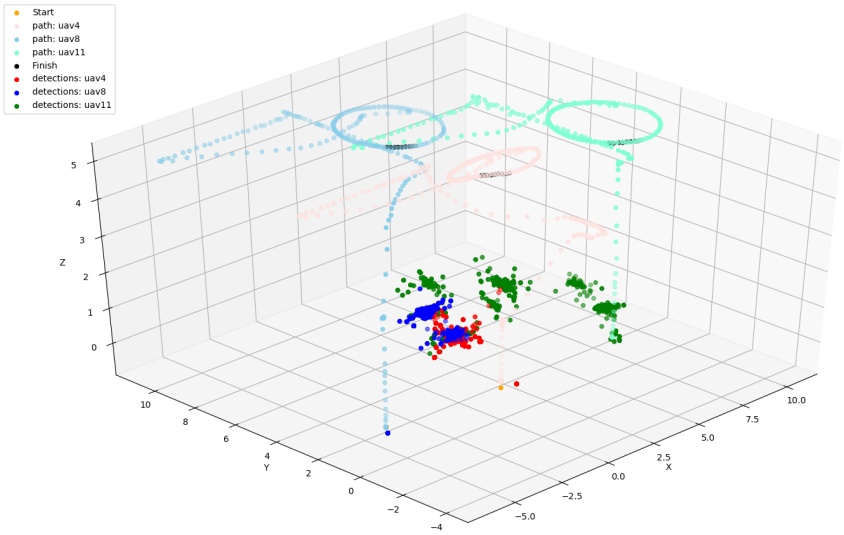


Figure 5.6: 3D drone paths with locations of observations during the simulated experiment. Gazebo

Cost function values

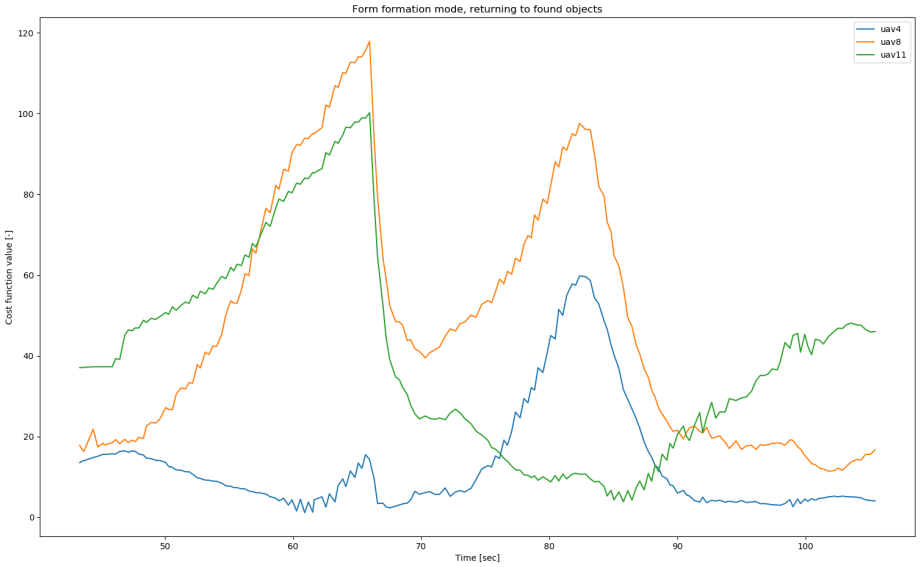


Figure 5.7: Convergence of cost values at the returning to the objects point during the simulated experiment

The drones were able to track the objects successfully, predominantly when they had been driven away from objects and returned to the things automatically. The cost values of the part of the experiment are depicted in Fig.5.7 At sec 65, the mode was switched to "form formation," and drones started immediately reducing the value of their cost function. UAV4 and UAV8 experienced a rise and fall in cost function as they were moving to shape the formation around the centroid. The rise of the cost value of UAV 11, at the end of the chart, is due to switching to "searching" and starting of circular trajectory. Fig.5.7

Distances to the tracked centroid

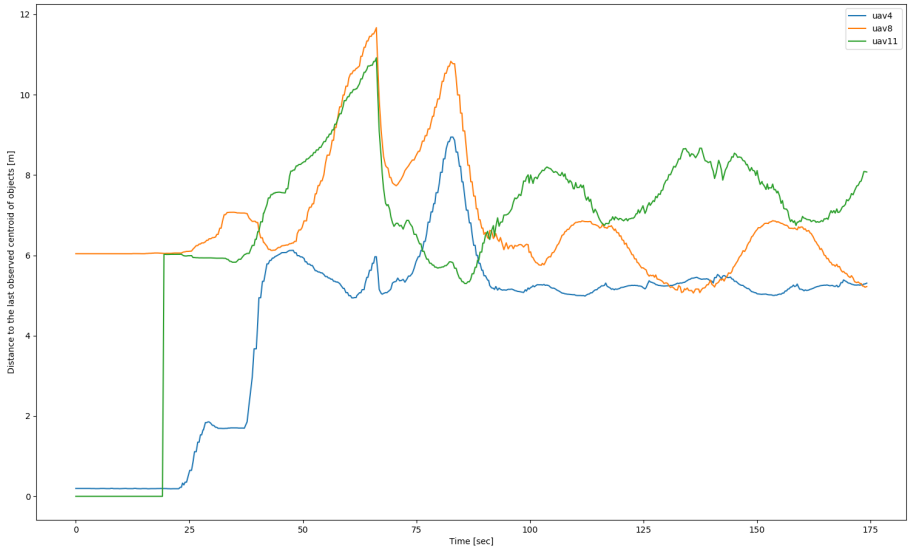


Figure 5.8: Distances between drones and lastly observed centroid of the tracked objects below. Simulation

During the simulated experiment, the distances between drones and the last tracked centroid could be observed in the following figure: Fig.5.8. It depicts the part of the automatic flight from the 50s to the 100s of the experiment. It also depicts how the drones converge to the initial formation from 0s to 50s: during this phase, initially, no centroid was detected. At around 100s, drones formed a triangular formation around the objects. After that, they were put on automatic searching mode, following a circle, resulting in a sinusoidal shape of distances. Finally, the drones were stopped, as all system functions were tested. Fig.5.8

5.2 Real-world experiments

It was necessary to drive outside Prague to conduct the experiment in a real-world environment, testing the drones on the field. The drones were deployed many times, however the most remarkable flights were the following two:

1) Formation initial shaping, moving and stopping. Video.[44]
2) Full algorithm, discussed in the following part of the chapter.Video.[45]

5.2.1 System configuration

In real-world experiments, we used drones developed by the MRS team. The computer is Intel NUC, and the flight controller is Pixhawk 4. A RealSense camera was on each drone facing downwards. The camera's location was

exactly at the front robot, similar to the configuration in the simulation. GPS was a localization sensor in the field. Fig.5.9

Figure 5.9: Real-world drone setup. Temeshvar camp

5.2.2 Experiment results, Temeshvar camp

Figure 5.10: Blue and red plates. Temeshvar camp

To put some tracking objects below drones, 3d-printed plates were used. They were red and blue. Fig.5.10 After experiencing real-world drone flight, it was possible to prepare more effectively and solve issues through experimentation. The main finding was that it was important to calibrate drones' cameras explicitly. Sunlight reflects in high contrast from the plates. And consequently, the color-exposure configuration has been tuned and reduced to reflect a feasible detection image. Other than that, the experiment was

successful because drones were flying in formation. They recognized the plates and formed a triangle around them. The real-world experiment still required at least two people to run the system: a safety pilot and a formation controller.

Figure 5.11: Initial convergence to formation. Temeshvar camp

Figure 5.12: Formation around detected objects. Temeshvar camp

Procedure of conducting real-world experiment

Due to the high-stress environment during the experiment preparation, it was decided to add the procedure that was followed during the successful implementation of the formation control algorithm.

Before flight:

1) Check configurations of the .bashrc file, which must include the workspace

2) Check if the propellers are ready and in a good state
3) Check if a Wi-Fi connection in the environment
4) Put on charge and ensure that radio transmitters' batteries are full
5) Charge computers as well because the experiment can take a significant amount of time
6) Charge the batteries of drones
7) It is recommended to set controller constraints to slow at the first run
During the flight:
1) Run tmux scripts and wait until there are no warnings
2) Check the angle [own angle] number; it should be different for each of the drones
3) Record the centroid by pressing the r button twice
4) Ensure that the recording of the centroid is off in the motion optimizer node
5) ensure that [current angle] changes with time
Although it may be tedious, this procedure has helped to save a lot of time during the preparation for and implementation of the experiment.

Paths, obtained from the experiment

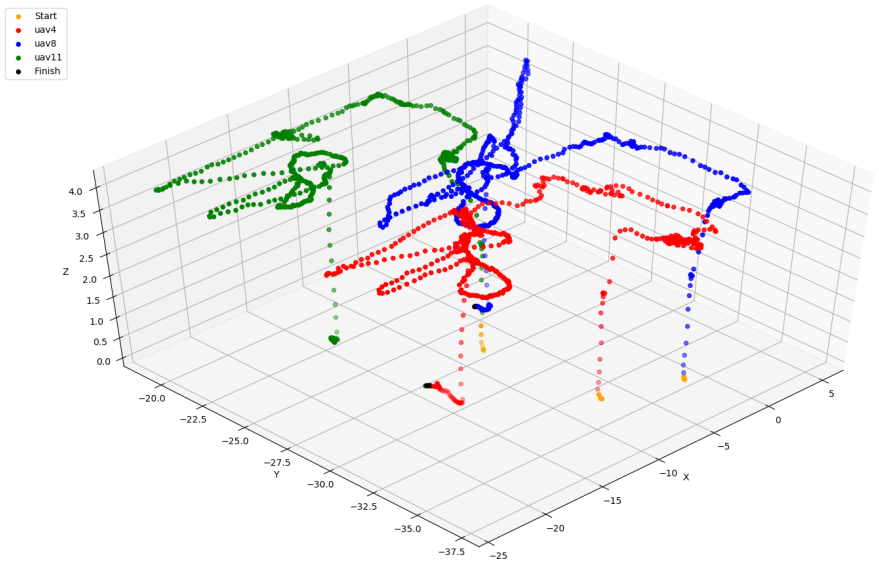


Figure 5.13: 3D drone paths during the experiment. Temeshvar camp

The plot (Fig.5.13) depicts the motion of drones during the entire experiment. There were several phases during the flight, such as search mode (manual, circle following), flying away from targets, and returning to the at the end of the experiment. The orange points indicate the starting location of the drones. The drones' X and Y positions were measured relative to the origin at the Temeshvar camp. Black points represent the finish position. Because

of the similarities, in terms of shapes, between drones' paths, it is possible to conclude that the drones were flying in formation.

Unexpected coincidence

If looking closely at Fig.5.13, it is possible to see that the path UAV 8 deviated in the middle of the experiment. In order to investigate the cause of the deviation, it was decided to plot the sensor's measurements and depict them with the paths. Fig.5.15. It is possible to see that there is noise in the observations, such that the observed points are along the paths. Therefore, it can be concluded that there was a false positive measurement. Furthermore, it can be verified by looking at the video captured by the drone's camera. Fig.5.14. There was some object that was hanging around the camera, and it caused inaccurate measurements.

Figure 5.14: False-positive measurement, noise by some hardware part

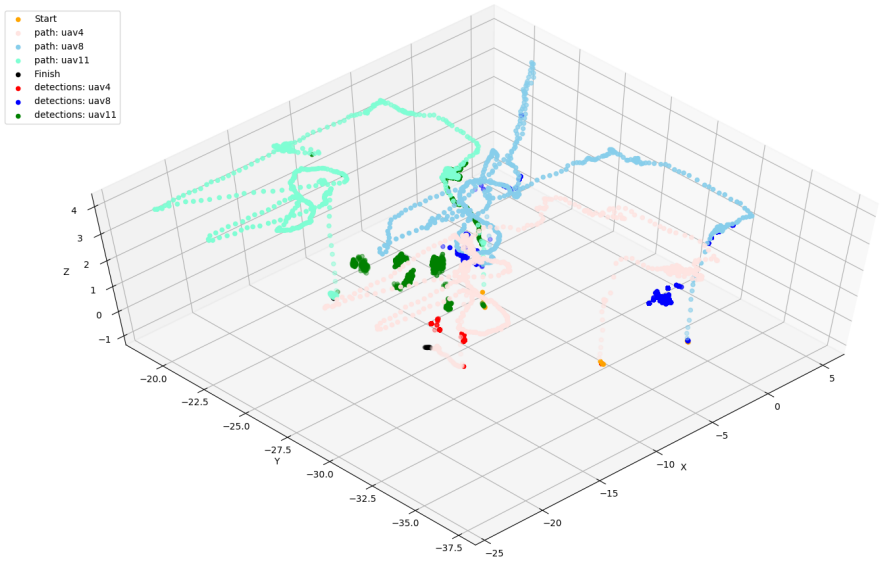


Figure 5.15: 3D drone paths with locations of observations during the experiment. Temeshvar camp

Cost function values

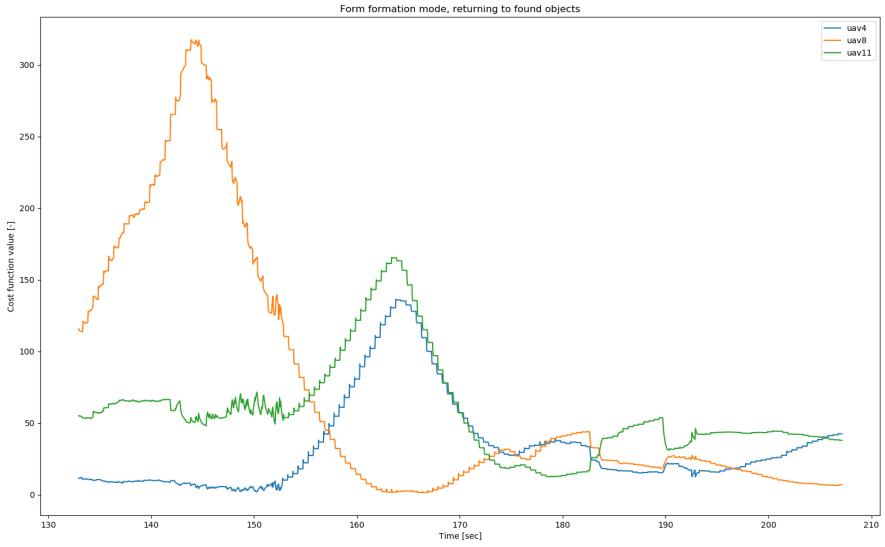


Figure 5.16: Convergence of cost values at the returning to the objects point during the experiment

Nevertheless, the drones were able to track the objects successfully, especially when they were driven away from objects and returned to the things automatically. The cost values of the part of the experiment are depicted in Fig.5.16

As can be observed, initially, drones 4 and 11 were close to the centroid of tracked objects, while UAV 8 was far from the center(high cost-function values). Moreover, eventually, the drones were driven away, making robots 4 and 11 farther from the centroid (high cost-function values), whereas robot 8 became closer to the objects(low cost-function value). Finally, the values converged to the cost of around ten because they all formed a triangle around the centroid. Fig.5.16

And the observation of multiple objects can be verified from a picture taken by the UAV 11: Fig.5.17 The drones have correctly identified blue and

Figure 5.17: Detected pink and blue plates on the grass of Temeshvar field. UAV11

red(pink) plates and marked them with the respective color.

Distances to the tracked centroid

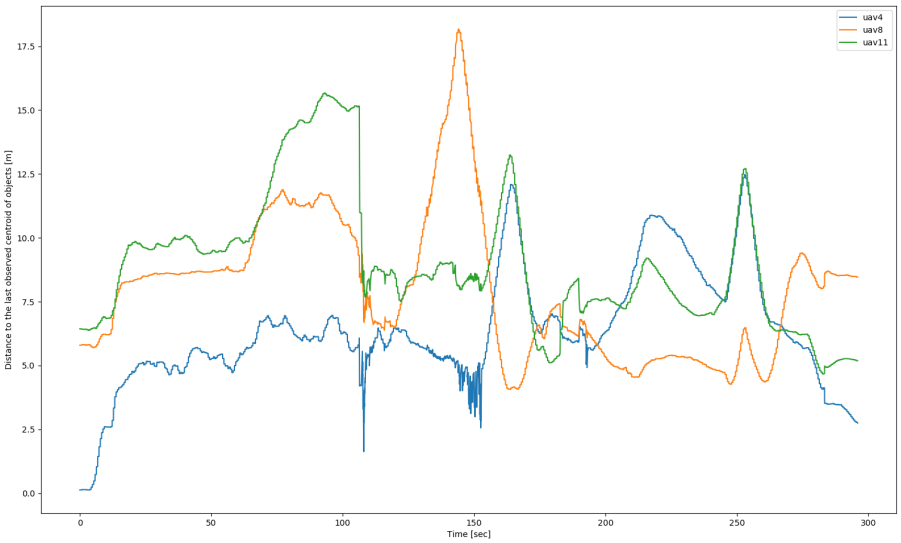


Figure 5.18: Distances between drones and lastly observed centroid of the tracked objects below.

During the entire experiment, the distances between drones and the tracked centroid could be observed in the following figure: Fig.5.18. It depicts the part of the automatic flight from the 130s to the 200s of the experiment. It also depicts how the drones converge to the initial formation from 0s to 130s: the curves on the figure are equally separated. At around 200s, drones formed a triangular formation around the objects, and after that, they were driven away by a person controlling the formation up to the 250s. After that, they were put on automatic searching mode, which was the following of a circle, resulting in a sinusoidal shape of distances. Finally, the drones were stopped, as all system functions were tested. It is possible to conclude from the graph 5.18 that the searching mode controlled by the user was from 130s to 180 s and, from 220s to 250s, approximately, form formation mode was applied from 180s to 220s, and searching mode(automatic) was tested at around 250s of the experiment. Formation mode was also applied at 100s, which resulted in a sudden drop in the distance values. Fig.5.18

Estimation values of the centroid between objects

P(t|t-1)						
x	4.09255	0	0	1.69626	0	0
y	0	4.09255	0	0	1.69626	0
z	0	0	4.09255	0	0	1.69626
v_x	1.69626	0	0	1.05069	0	0
v_y	0	1.69626	0	0	1.05069	0
v_z	0	0	1.69626	0	0	1.05069

Table 5.1: Prediction co-variance values, which were taken at time close to the beginning of experiment. UAV 8

P(t|t)						
x	2.2505	0	0	0.932776	0	0
y	0	2.2505	0	0	0.932776	0
z	0	0	2.2505	0	0	0.932776
v_x	0.932776	0	0	0.734244	0	0
v_y	0	0.932776	0	0	0.734244	0
v_z	0	0	0.932776	0	0	0.734244

Table 5.2: Filtering co-variance values, which were taken at time close to the beginning of experiment. UAV 8

P(t|t-1)						
x	0.947005	0	0	0.760002	0	0
y	0	0.947005	0	0	0.760002	0
z	0	0	0.947005	0	0	0.760002
v_x	0.760002	0	0	3.44158	0	0
v_y	0	0.760002	0	0	3.44158	0
v_z	0	0	0.760002	0	0	3.44158

Table 5.3: Prediction co-variance values, which were taken at time close to the end of experiment. UAV 8

P(t|t)						
x	0.796203	0	0	0.638979	0	0
y	0	0.796203	0	0	0.638979	0
z	0	0	0.796203	0	0	0.638979
v_x	0.638979	0	0	3.34445	0	0
v_y	0	0.638979	0	0	3.34445	0
v_z	0	0	0.638979	0	0	3.34445

Table 5.4: Prediction co-variance values, which were taken at time close to the end of experiment. UAV 8

5.3 Discussion

More and more practice leads to a better comprehension of drone systems and the loss of the fear of flying them. It is important to stress that real experience with the system is much more beneficial than a simulated one. However, it is hard to start the development of an algorithm on real drones, and this proves the usability and necessity of a simulation environment. Finally, both experiences: simulated and natural, are essential for the development of the robotic system, as well as in any other engineering discipline. As for the final results, the control of drone formations was achieved without a central planning unit that could guide the drones or let them fly autonomously. Moreover, due to the precise calibration of the camera and testing the perception on a grass field, it was possible to observe and recognize a series of plates on the grass and direct drones to form formations around them. Fig.5.12

Chapter 6

Conclusion

In the simulation, formation control was proven to be possible. The sensory mathematics behind sensor fusion was also proven to be stable. The motion is smooth, and the drones converge to the required positions. The primary goal of the work was achieved successfully. The drones move autonomously and in a triangular formation when they lock in the sight of the leader drone, and the leader drone moves slowly. The perception part of the algorithm is correct and fast because the observing node works at a 1000 Hz rate. Therefore, the delay period is equal to 1 millisecond, which is fast for image processing. The control part is sufficiently workable. However, the performance can be improved. The dependency on the target's speed makes it difficult to converge to a goal that is moving fast. At low rates, the controller works in a proper and expected manner.

In real-world environments, the operation of drones is workable as well. The situation of collaborative object detection was studied. All requirements of the project have been satisfied and completed. Fig. 5.11 Fig.5.12

6.1 Further research suggestion

In the future, it is possible to change the color-blob detector to a more sophisticated algorithm, such as a deep neural network. It is possible to train it on a data set of garbage items for off-shore cleaning projects. Furthermore, training a neural network to recognize other drones is possible. Finally, by using the swarming technique, it is possible to create attraction and repulsion vectors from drones and garbage items. This approach would eliminate the need for communication. However, it should be further researched and tested.

Bibliography

[1] A. M. de Souza Neto and R. A. F. Romero, "A decentralized approach to drone formation based on leader-follower technique," in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, 2019, pp. 358–362.

[2] X. Zhang, X. Yu, J. Lu, and W. Lan, "Distributed leader-following formation control for mobile robots with unknown amplitudes of leader's velocity," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 4889–4894.

[3] V. T. J. Antonio, G. Adrien, A.-M. Manuel, P. Jean-Christophe, C. Laurent, R. Damiano, and T. Didier, "Event-triggered leader-following formation control for multi-agent systems under communication faults: Application to a fleet of unmanned aerial vehicles," *Journal of Systems Engineering and Electronics*, vol. 32, no. 5, pp. 1014–1022, 2021.

[4] Z. Miao, H. Zhong, Y. Wang, H. Zhang, H. Tan, and R. Fierro, "Low-complexity leader-following formation control of mobile robots using only fov-constrained visual feedback," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4665–4673, 2022.

[5] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, and A. Ahmad, "Active perception based formation control for multiple aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4491–4498, 2019.

[6] M. J. van Pampus, A. Haseltalab, V. Garofano, V. Reppa, Y. H. Deinema, and R. R. Negenborn, "Distributed leader-follower formation control for autonomous vessels based on model predictive control," in *2021 European Control Conference (ECC)*, 2021, pp. 2380–2387.

[7] D. Menegatti, A. Giuseppi, and A. Pietrabissa, "Model predictive control for collision-free spacecraft formation with artificial potential functions," in *2022 30th Mediterranean Conference on Control and Automation (MED)*, 2022, pp. 564–570.

[8] H. Sun, J. Qi, C. Wu, and M. Wang, "Path planning for dense drone formation based on modified artificial potential fields," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 4658–4664.

[9] Z. Pan, C. Zhang, Y. Xia, H. Xiong, and X. Shao, "An improved artificial potential field method for path planning and formation control of the multi-uav systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1129–1133, 2022.

[10] J. Dai, Y. Sun, J. Ying, and H. Nie, "Research on cooperative obstacle avoidance control of uav formation based on improved potential field method," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 4633–4638.

[11] J. Shao, "Leader-following formation control of multiple mobile vehicles," *IET Control Theory and Applications*, vol. 1, pp. 545–552(7), March 2007. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-cta_20050371

[12] Z. Yuan, C. Du, J. Chen, and F. Ling, "Central-distributed control model of uav group and its application in perception module," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2019, pp. 677–681.

[13] Q. Han, A. Zhang, X. Wang, R. Cao, and G. Chen, "Leader-following consensus of multi-agent system with multiple leaders under formation control," in *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*, 2020, pp. 380–384.

[14] B. Kada, M. Khalid, and M. S. Shaikh, "Distributed cooperative control of autonomous multi-agent uav systems using smooth control," *Journal of Systems Engineering and Electronics*, vol. 31, no. 6, pp. 1297–1307, 2020.

[15] Z. S. Lippay and J. B. Hoagg, "Leader-following formation control with time-varying formations and bounded controls for agents with double-integrator dynamics," in *2020 American Control Conference (ACC)*, 2020, pp. 871–876.

[16] H. Kang, W. Wang, C. Yang, and Z. Li, "Leader-following formation control and collision avoidance of second-order multi-agent systems with time delay," *IEEE Access*, vol. 8, pp. 142 571–142 580, 2020.

[17] B. Zhang, J. Chen, and B. Hu, "Distributed control of 6-dof leader-following multispacecraft formation near an asteroid based on scaled twistors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 2, pp. 1168–1182, 2023.

[18] W. Dunbar and R. Murray, "Model predictive control of coordinated multi-vehicle formations," in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, vol. 4, 2002, pp. 4631–4636 vol.4.

[19] D. Gu and E. Yang, "A suboptimal model predictive formation control," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1295–1300.

[20] M. Q. Phan and J. S. Barlow, "Optimal model predictive control formations," in *2008 Chinese Control and Decision Conference*, 2008, pp. 55–64.

[21] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 7488–7493.

[22] T. P. Nascimento, A. P. Moreira, and A. G. Scolari Conceição, "Multi-robot nonlinear model predictive formation control: Moving target and target absence," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1502–1515, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889013001310

[23] E. Nejabat and A. Nikoofard, "Switched robust model predictive based controller for uav swarm system," in *2021 29th Iranian Conference on Electrical Engineering (ICEE)*, 2021, pp. 721–725.

[24] Z. Pu, T. Zhang, X. Ai, T. Qiu, and J. Yi, "A deep reinforcement learning approach combined with model-based paradigms for multiagent formation control with collision avoidance," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–16, 2023.

[25] D.-W. Zhang, G.-P. Liu, and L. Cao, "Proportional integral predictive control of high-order fully actuated networked multiagent systems with communication delays," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 2, pp. 801–812, 2023.

[26] A. D. Dang, H. M. La, T. Nguyen, and J. Horn, "Formation control for autonomous robots with collision and obstacle avoidance using a rotational and repulsive force–based approach," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, p. 1729881419847897, 2019. [Online]. Available: https://doi.org/10.1177/1729881419847897

[27] K. Chen, Y. You, G. Luo, and X. Guo, "Improved multi-uuv formation control for artificial potential fields and virtual navigators," in *2021 IEEE 7th International Conference on Control Science and Systems Engineering (ICCSSE)*, 2021, pp. 108–113.

[28] (2023) Rprop. Internet. [Online]. Available: https://en.wikipedia.org/wiki/Rprop

[29] T. Uzakov, T. P. Nascimento, and M. Saska, "Uav vision-based nonlinear formation control applied to inspection of electrical power lines," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 1301–1308.

[30] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1801–1807, 2018.

[31] C. Jiang, Z. Chen, and Y. Guo, "Learning decentralized control policies for multi-robot formation," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2019, pp. 758–765.

[32] Z. Sui, Z. Pu, J. Yi, and T. Xiong, "Formation control with collision avoidance through deep reinforcement learning," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[33] K. M. Kabore and S. Güler, "Distributed formation control of drones with onboard perception," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 3121–3131, 2022.

[34] W. Ding, X. Chen, W. Zhu, and Q. Ren, "Vision-based formation control for a heterogeneous multi-robot system," in *2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*, 2021, pp. 1791–1796.

[35] Z. Miao, H. Zhong, J. Lin, Y. Wang, and R. Fierro, "Geometric formation tracking of quadrotor uavs using pose-only measurements," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1159–1163, 2022.

[36] J. Lin, Y. Wang, Z. Miao, Q. Lin, G. Hu, and R. Fierro, "Robust linear-velocity-free formation tracking of multiple quadrotors with unknown disturbances," *IEEE Transactions on Control of Network Systems*, pp. 1–12, 2023.

[37] J. Lin, Z. Miao, Y. Wang, G. Hu, and R. Fierro, "Aggressive formation tracking for multiple quadrotors without velocity measurements over directed topologies," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–12, 2023.

[38] (2023) Consensus | english meaning - cambridge dictionary. Online. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/consensus

[39] "Dynamics and control of networks," Prague, Czech Republic, 2020. [Online]. Available: https://moodle.fel.cvut.cz/courses/BE3M35DRS

[40] C. Igel and M. Husken, "Improving the rprop learning algorithm," ICSC Academic Press, 2000.

[41] T. Uzakov. Simulation of formation breaking. diploma thesis. Youtube. [Online]. Available: https://youtu.be/Ekw2QI0HSYo

[42] ——, "The first approach to formation control," GitHub. [Online]. Available: https://github.com/uzakotim/semestral_project_uav

[43] ——, "The second approach to formation control," GitHub. [Online]. Available: https://github.com/uzakotim/formation_v.2

[44] ——. Drones flying in formation. diploma thesis experiment. Youtube. [Online]. Available: https://youtu.be/9yjoc-Zi1w8

[45] ——. Full test of the algorithm. diploma thesis experiment. Youtube. [Online]. Available: https://youtu.be/urFFrfw2J0k