POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Autonomous navigation and obstacle avoidance for interconnected tethered drones in a partially unknown environment

Tesi di Laurea Magistrale in
Automation and Control Engineering

Author: **Fabrizio Cirillo**

Student ID: 945226
Advisor: Prof. Lorenzo Mario Fagiano
Co-advisors: Ing. Danilo Saccani, Ing. Michele Bolognini
Academic Year: 2021-2022

Abstract

Systems of tethered multicopters, called STEM, are multi-copter drones connected one to the other creating a chain by an electric tether providing energy and communication. Researchers and industry are more and more interested in these systems due to their versatility and prolonged flight time which otherwise is limited. They are power supplied from ground by the ground station. On the other hand, the tether, which couples the vehicles, introduces range limitations and for this reason, challenging control and navigation problems arise. This thesis describes a high level controller for the autonomous navigation of such system. The proposed approach to solve these challenges is based on a combination of off-line and real-time optimization. First, optimal configurations are found through an off-line mission planning method for these peculiar systems in the nominal environment, guaranteeing safety with regards to the presence of the obstacles and tether. Subsequently, an on-line, path following algorithm, based on Model Predictive Control (MPC), is presented to bring the system to the aforementioned optimal configuration. This procedure ensures collision avoidance for the vehicles and for the tether connecting them through the usage of LiDAR readings, which provide partial information of surrounding environment. The mentioned contributions are validated through simulation with a realistic model of the system.

Keywords: UAV, System of Tethered Multicopters, Numerical Optimization, Simulation

Abstract in lingua italiana

I sistemi multicottero interconnessi tramite cavo, chiamati STEM, sono dei droni con più rotori connessi uno all'altro in modo tale da creare una catena attraverso un cavo elettrico che fornisce energia e permette la comunicazione. I ricercatori e le industrie sono sempre più interessate in questi sistemi grazie alla loro versatilità e al loro tempo di volo prolungato, che altrimenti risulta essere limitato. Questi sistemi sono alimentati da terra attraverso una stazione di terra. D'altro canto, il filo, che accoppia i veicoli introduce delle limitazioni di portata e per questo motivo, devono essere introdotti dei problemi di controllo e di navigazione molto impegnativi. La tesi descrive un controllore di alto livello per la navigazione autonoma di questi sistemi. L'approccio proposto per risolvere queste sfide, si basa sulla combinazione di un'ottimizzazione off-line e in tempo reale. In primo luogo, le configurazioni ottime si trovano attraverso una metodo di pianificazione della traiettoria offline che agisce nell'ambiente nominale, garantendo sicurezza con riguardo alla presenza di ostacoli e quella del filo. Successivamente, viene presentato un algoritmo di inseguimento della traiettoria che opera in tempo reale che si basa sul modello di controllo predittivo, per portare il sistema alla suddetta configurazione ottimale. Questa procedura garantisce la prevenzione delle collisioni per i veicoli e per il cavo che li collega attraverso l'uso di letture LiDAR, che forniscono informazioni parziali sull'ambiente circostante. I contributi citati sono validati mediante simulazione con un modello realistico del sistema.
Parole chiave: UAV, Sistema di Multicotteri Interconnessi tramite cavo, Ottimizzazione Numerica, Simulazione

Contents

1 | Introduction

In the last decade, the cost of Unmanned Aerial Vehicles (UAVs) has decreased, for this reason they are more and more used in many applications such as mapping [4], monitoring of the built environment [13] and so on. The interest in UAVs has increased as well, in particular in the field of industry and academia. On the other hand, systems of tethered drones
are used in activities where the prerequisite is a prolonged flight time, which for battery-powered systems is limited. The topic of this thesis consists in a high level planner used by a system, which is called STEM (System of TEthered Multicopters) [7]. STEM is composed of a chain of multicopter drones, tethered to each other and to a ground station. The tethers permits communication and power supply to the drones [2].

Figure 1.1: Drone used in monitoring of building zone

1.1. State of the art

The problem of navigation with constraints and collision avoidance emerges in many different fields. In fact, the available literature offers numerous solutions and approaches to solve the aforementioned problem. There are approaches, based on A^* [10], a path search algorithm, where the environment is represented by a graph and the feasible path is searched on it. Some techniques related to Artificial Potential Field (APF) generation have been studied [5], where the field is artificially computed and the aim is to elaborate a trajectory which minimizes the potential energy related to it. Another example is represented by the family of algorithms related to Rapidly exploring Random Tree (RRT) [14], where tree of points are formed in the space from the starting position. Then, the best tree among the collection of them is chosen, according to some criterion (i.e. distance from the goal). For what concerns the trajectory tracking for a single drone in an on-line framework, the approach with MPC has been developed [11]. Here, each

obstacle is approximated as a set of linear inequalities. This approach, due to its nature, permits to have multiple inputs and can deal with linear constraints, guaranteeing safety. In addiction, in this research, a multi-trajectory MPC is described, proposing a comparison with the canonical MPC framework. On the other hand, few publications are available for what concerns design and control of STEM. In [7], the general model of STEM is presented, then a hierarchical control approach is described. It is composed by local controllers which have the aim to control the dynamic of the drones and of the winches and a Supervisory high level planner, which in this case is an MPC. Another research involves the control of a formation of drones acting on the tether tension force [12], through a feedback linearization technique, considering fixed the length of the tether. Another research takes into account the similarity between a system of tethered drones and a redundant manipulator [3], where the drones are modelled as three-dimensional spherical joints, while the tether connecting them are assumed to be mono-axial prismatic joints. Then, considering a known static environment a path planning strategy has been developed. The approach proposed in this research uses information of the environment provided by a map. Then, based on this, a offline path planner is used to retrieve a feasible path. The main novel aspects of the proposed approach with respect to the literature is to approximate the free-space where the drones can move with a convex polytope able to guarantee that the vehicles and tethers remain in an obstacle-free area.

Figure 1.2: An example of tethered drone

1.2. Main Contributions

This thesis can be seen as an extension of [1], where the environment is partially known through the usage of a map. The knowledge of the environment is used here by a offline planner, while the online path following algorithm is accomplished by MPC which take as inputs the map and the readings of LiDAR sensors. Then, the high level controller generates at each sampling time the reference points for the drones. Once the assigned targets are reached, a second configuration is found by offline planner. Finally, the second set of targets are reached, maybe including some backtrack process. To summarize, the main contributions of this research are:

- the formulation of an optimal offline planner in a well known environment whose aim is to compute an optimal configuration which the drones have to track;

- the development of a path following algorithm, based on MPC and LiDAR readings, which aims to bring the system to the desired configuration respecting constraints related to obstacle avoidance and to the distance between drones during simulation;
- the formulation of a second optimal offline planner which takes as inputs the final position reached by the drones and a new target and finds a new optimal configuration;
- the development of an online strategy which chooses if it is better to have a kind of backtracking of the drones or it is better to have the drones directly moved to the new targets;
- the development of path planner which can backtrack the drones according to the chosen strategy;
- test of the approach in simulation using a simplified model of the drones called "oriented control model".

1.3. Outline

The thesis is organized as follows:

- **Chapter 2 - Model of the system** describes the model of the system composed by drones, tether and winch involving simplifying assumption. Then, control oriented model is described. In the final part, environment and LiDAR sensors are described.
- **Chapter 3 - Proposed approach** describes the proposed solution. First, an overview of the offline path planning is given with the subsequently online path following approach. Furthermore, a new optimal configuration is obtained by the second offline path planning algorithm. Moreover, the strategy which chooses if backtrack the drones or not is described. In conclusion, the drones are brought to their previously found configuration with the execution of the online path following.
- **Chapter 4 - Results** shows the results obtained in simulation with the navigation algorithm, not only with known obstacles, but also with unknown ones. Different targets and different initial conditions are given to the algorithm to test its performance and robustness.
- **Chapter 5 - Conclusions** finally summarizes the main conclusions and contributions performed. Here, open problems and future developments are also considered.

2 | Description and model of the system

This research is based on a system of tethered drones, where several drones are connected one to the other with cables in a series structure. The drone at the end of the series is connected through another cable to the ground station. The electrical power for the drones is generated by the ground station and transmitted via the cables. In addition also data can be transmitted. It is possible to treat the problem of coordinating the motion of each aerial vehicle in a centralized or distributed way since they realize a communication network which can be represented with an all-to-all connection graph. Moreover the length of the cables can be modified with a winch, as the ground station and all drones except the last one are equipped with a winch that controls such length. Here, quadcopters are considered, even if their specifics are relevant for their control and simulation only and do not influence the autonomous navigation problem directly. Each drone is provided with a Inertial Measurement Unit (IMU) which can measure attitude and three-dimensional position and velocity with respect to a global inertial reference frame. Furthermore, winch position and speed are also measured. Moreover, each drone is also equipped with LiDAR sensor, which allows the scanning of environment in all directions. A right handed reference system fixed to the drone is taken, with height represented by z axis and pointing upwards. Scans are executed on the plane xy and on a plane perpendicular to that, in such a way that is possible to measure distances from obstacles above and below the drone. The ground station, which acts as a centralized controller and elaborates position and yaw angle references for each drone, takes as input this information. References of the drones are tracked by lower level local controllers which on one hand manipulate the four rotors and on the other hand control the length of the cable through the winch torque. The subject of this research is the high level control algorithm which autonomously elaborates position references for the drones. The global reference, which is used to discuss about the model, is a right-handed reference frame with z axis pointing upwards and fixed with respect to the environment. For simplicity and without loss of generality this represents the position of the ground station. $N_d \in \mathbb{N}$ drones are considered, and identified by the

index $i = 1, ..., N_d$.

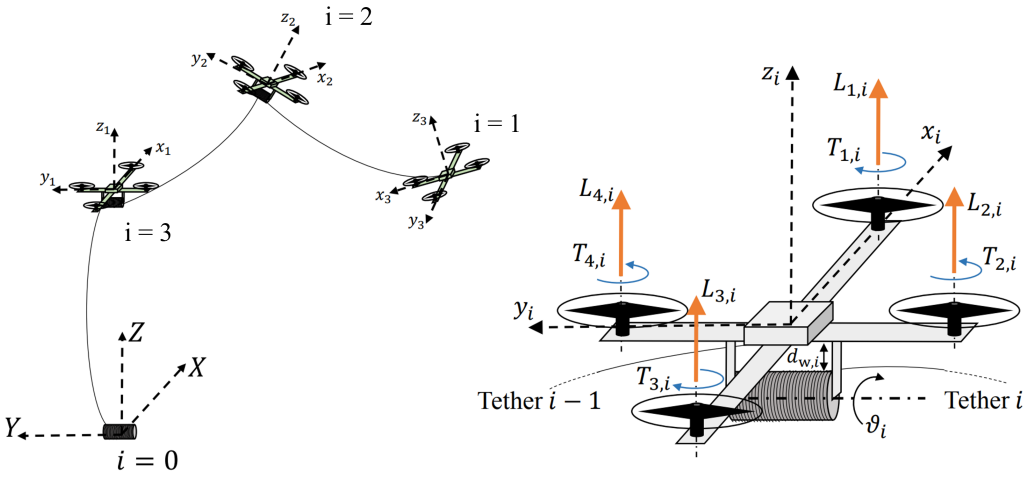

Figure 2.1: A representation of the scheme and convention of the system: Drone$_1$ called also "leader drone" is considered to be the farthest from the ground station [7].

Euler angles: roll, pitch and yaw, respectively $\phi_i(t), \vartheta_i(t)$ and $\psi_i(t)$, describe the attitude of the drones with respect to the global frame. Position and velocity of the i-th drone are denoted as $\boldsymbol{p}_i^g(t), \dot{\boldsymbol{p}}_i^g(t)$ where the subscript ^{g} indicates that the variables are referred to the global framework. Bold symbols denote vectors.

2.1. Drones model

The considered model of the quadcopters is the rather standard one and summarized here for completeness (see [8] for more details), with the contribution of forces and moments given by the tethers attached to the drone. All drones are assumed to be identical. The equation of a single drone is the following:

$$m_{d,i}(t) = \overline{m}_{d,i} + m_{w,i}(t) + \frac{1}{2} m_{t,i}(t), \tag{2.1}$$

where the mass of the i-th drone at each time instant is the sum of three contributions: the mass of the vehicle alone $\overline{m}_{d,i}$, the weight of the winch and the stored cable $m_{w,i}(t)$ and half of the weight of the extended cable $m_{t,i}(t)$ which connects drone i to drone $i+1$, with shared weight.

The four rotors produce a lift force and a drag torque, both proportional to the square of

the rotational speed of the rotor Ω, through the lift and drag coefficients of the drone, b and d:

$$\begin{aligned} L_{f_{i,j}}(t) &= b\Omega_{i,j}^2, \quad j = 1, ..., 4 \\ T_{i,j}(t) &= d\Omega_{i,j}^2, \quad j = 1, ..., 4 \end{aligned} \tag{2.2}$$

For control purpose, initial inputs such as the four lift forces and four drag torques are recombined in a linear combination in four new inputs. They represent, respectively, the total lift force along z_i axis, which is the vertical axis of the local reference frame fixed to the drone, and the yaw moments along the three axes of the same reference frame: x_i, y_i, z_i. For each drone, it can be written:

$$\begin{aligned} u_{i,1}(t) &= \sum_{j=1}^{4} L_{f_{i,j}}(t) \\ u_{i,2}(t) &= a(L_{f_{i,4}} - L_{f_{i,2}}) \\ u_{i,3}(t) &= a(L_{f_{i,3}} - L_{f_{i,1}}) \\ u_{i,4}(t) &= (T_{i,2} + T_{i,4}) - (T_{i,1} + T_{i,3}). \end{aligned} \tag{2.3}$$

In (2.3), a is the distance between the rotor and the center mass of the drone.
To derive the model of the system, it is applied Newton's law, taking into account pulling forces of tethers, the torque applied by onboard winch and the variable mass of the drone:

$$\begin{aligned} \ddot{\boldsymbol{p}}_i^g &= \frac{1}{m_{d,i}} \left(R_i^T \begin{bmatrix} 0 \\ 0 \\ u_{i,1} \end{bmatrix} + (\boldsymbol{F}_i^g - \boldsymbol{F}_{i-1}^g) \right) - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \\ \dot{p}_i &= \frac{I_{i,y} - I_{i,z}}{I_{i,x}} q_i r_i + \frac{u_{i,2} - J_p}{I_{i,x}} q_i \Omega_{r,i} + d_{i,w} F_i^y \\ \dot{q}_i &= \frac{I_{i,z} - I_{i,x}}{I_{i,y}} p_i r_i + \frac{u_{i,3} + J_p}{I_{i,y}} p_i \Omega_{r,i} - d_{i,w} F_i^x + \frac{u_{i,w}}{I_{i,y}} p_i \\ \dot{r}_i &= \frac{I_{i,x} - I_{i,y}}{I_{i,z}} p_i q_i + \frac{u_{i,4}}{I_{i,z}}, \end{aligned} \tag{2.4}$$

where $p_i(t), q_i(t), r_i(t)$ are the angular velocities of drone i around its axes x_i, y_i, z_i, R_i^T is the rotation matrix which transforms the local coordinates in global ones and depends on Euler angles, F_i describes the force acting along the directions of the global reference frame, g is the gravity acceleration, $I_{i,*}$ is the angular mass of the drone around specific axis, J_p is the moment of inertia of the propeller and $d_{i,w}$ represents the distance between the drone center of gravity and the point where the cables leaves the winch, applying its torque. Now, some assumptions which simplify the problem have been considered:

- drag forces of the cable are neglected;
- variation of moments of inertia of the drones due to the unwinding of the cable are neglected.

2.2. Tether and Winch models

Winches are identified by the progressive index i, where $i = 0$ corresponds to the ground station, and the subsequent $i = 1, ..., N_d - 1$ correspond with the index used for the drones. Even the cable is identified with the index of the corresponding winch, (i.e. the cable which connects the first drone to the ground station is denoted with $i = 0$). The angular position and velocity of the i-th winch, $\theta_i(t), \dot{\theta}_i(t)$, are its state. It is assumed that when the measured position $\theta_i(t) = 0$ the cable is completely wound around the winch. Then, assuming that the whole cable can be coiled on a single layer, i.e. the external radius of the winch is independent with respect the length of unreeled tether, it is possible to compute the mass of the winch as:

$$m_{w,i}(t) = \overline{m}_{w,i} + \left(\bar{l}_{t_i} - r_{e,i}\theta_i(t)\right)\rho_{t,i}, \tag{2.5}$$

where $r_{e,i}$ is the external radius (i.e. the mass of unwound tether is represented by the product $r_{e,i}\theta_i(t)$), $\rho_{t,i}$ is the unitary mass of the tether per length, $\bar{l}_{t_i}$ is the overall length of the tether i and $\overline{m}_{w,i}$ is the mass of the winch with no tether wounded. The moment of inertia of the winch can be approximated as a hollow drum, with internal radius $r_{\mathrm{i},i}$ and it is calculated as:

$$J_{w,i}(t) = \frac{1}{2} m_{w,i}(t)\left(r_{e,i}^2 + r_{\mathrm{i},i}^2\right). \tag{2.6}$$

The winch is physically defined by a viscous friction coefficient, which is assumed to constant and is denoted with $B_{w,i}$. The winch torque, which is a control input, is denoted as $u_{w,i}$ and is bounded in the interval $[\underline{u}_{w,i}, \overline{u}_{w,i}]$. The elongation of the tether $e_{t,i}(t)$ is compute as:

$$e_{t,i}(t) = max\left(0, \left\|\boldsymbol{p}_{i+1}^g(t) - \boldsymbol{p}_i^g(t)\right\|_2 - r_{e,i}\theta_i(t)\right). \tag{2.7}$$

Finally, the vector of forces which the tether exerts on the drone, expressed in global coordinates, is calculated from the elongation $e_{t,i}$ of the tether itself and its stiffness K_t, which is assumed to be constant:

$$\boldsymbol{F}_{t,i}^g(t) = K_{t,i}e_{t,i}(t)\frac{\boldsymbol{p}_{i+1}^g(t) - \boldsymbol{p}_i^g(t)}{\left\|\boldsymbol{p}_{i+1}^g(t) - \boldsymbol{p}_i^g(t)\right\|_2}, \tag{2.8}$$

where $p_0 = (0, 0, 0)$ is the ground station position. Therefore, using again Newton's law, it is possible to derive the state equation of the winch, recalling to the equilibrium of moments around the axis of rotation. In fact, for each winch, the equation of motion is:

$$\ddot{\theta}_i(t) = \frac{1}{J_{w,i}(t)} \left(r_{e,i} \left\| \boldsymbol{F}_{t,i}^g \right\|_2 - \beta_{w,i} \dot{\theta}_i(t) + u_{w,i}(t) \right), \tag{2.9}$$

Aerodynamic drag, as assumption, is neglected, considering negligible the speed of wind relative to the tether. In addition, in equation (2.8), it is assumed that drones i and $i+1$ exchange forces along the direction which connects their centre of mass, not along the direction of the two points where the cable is connected to the vehicles. Since the distance between the drones is higher than the distance between the center of mass and the cable attachment, this assumption can be considered valid.

2.3. Control-oriented model

The overall system of quadcopter, from now on, is assumed to be controlled with a low-level controller which has the same structure of the one described in [7]. This controller is assumed to be static and with a high enough working frequency (i.e. a typical value can be 50-100 Hz), to have good performance. The high-level planner (see Figure (2.2)), instead, computes the reference $\boldsymbol{P}_{ref} = \begin{bmatrix} P_{ref}^x & P_{ref}^y & P_{ref}^z \end{bmatrix}^T$ for the previous mentioned controller.

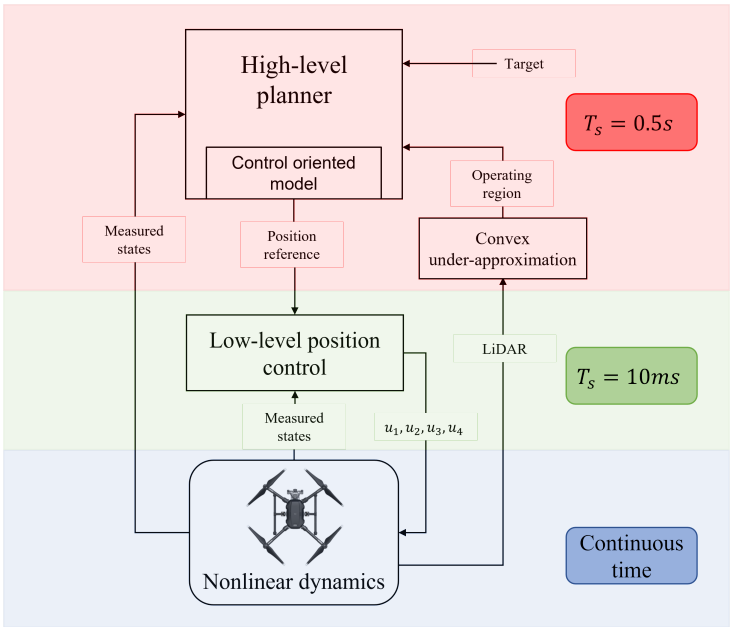

Figure 2.2: A representation of the hierarchy of control scheme: as said the low-level position controller works at high frequency, while the high-level one has an higher sampling time

The so called "control-oriented" model is the model used by the high-level controller, and it is a linear time invariant (LTI) model of the quadcopter in closed loop with the low-level controller. In this research, the considered model is a 2D one, involving only (x, y) coordinates of the drones. Now, a precise control-oriented model of the drone dynamics can be obtained considering the single i-th drone:

$$\begin{cases} \dot{\boldsymbol{x}}_i(t) = A_i \boldsymbol{x}_i(t) + B_i \boldsymbol{u}_i(t) \\ \boldsymbol{y}_i(t) = C_i \boldsymbol{x}_i(t) + D_i \boldsymbol{u}_i(t), \end{cases} \tag{2.10}$$

where the state vector is $\boldsymbol{x}_i(t) = \begin{bmatrix} P_i^x(t) & P_i^y(t) & V_i^x(t) & V_i^y(t) \end{bmatrix}^T$, $\boldsymbol{x}_i(t) \in \mathbb{R}^4$,the input vector is $\boldsymbol{u}_i(t) = \begin{bmatrix} P_{ref,i}^x(t) & P_{ref,i}^y(t) \end{bmatrix}^T$, $\boldsymbol{u}_i(t) \in \mathbb{R}^2$. $P_i^x(t), P_i^y(t)$ and $V_i^x(t), V_i^y(t)$ are respectively the (x, y) coordinates and x, y velocities of i-th drone, while $P_{ref,i}^x, P_{ref,i}^y$ are its reference position. To simplify the notation a more compact form is introduced, $\boldsymbol{P}_i(t) = \begin{bmatrix} P_i^x(t) & P_i^y(t) \end{bmatrix}$, $\boldsymbol{V}_i(t) = \begin{bmatrix} V_i^x(t) & V_i^y(t) \end{bmatrix}$ and $\boldsymbol{P}_{ref,i}(t) = \begin{bmatrix} P_{ref,i}^x & P_{ref,i}^y \end{bmatrix}$. The state matrices in explicit form are:

$$A_i = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_{vel}k_{pos} & 0 & -k_{vel} & 0 \\ 0 & -k_{vel}k_{pos} & 0 & -k_{vel} \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ k_{vel}k_{pos} & 0 \\ 0 & k_{vel}k_{pos} \end{bmatrix}, \tag{2.11}$$

$$C_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad D_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

where A_i is the state matrix, B_i is the input matrix, C_i is the output matrix (i.e. it is assumed that all the states are measured), finally k_{vel} and k_{pos} are gains which are tuned after a procedure of closed loop identification. Moreover, the system is discretized with zero-order hold (ZOH) method using $T_s = 0.5s$. This sampling time is suitable considering that the navigation control system is a high-level controller, while the dynamic control system of the drone works at higher frequencies (Fig 2.2).
Considering k as the discrete-time variable, the discrete-time system is:

$$\begin{cases} \boldsymbol{x}_i\left((k+1)T_s\right) = F_i \boldsymbol{x}_i\left(kT_S\right) + G_i \boldsymbol{u}_i\left(kT_s\right) \\ \boldsymbol{y}_i\left(kT_s\right) = H_i \boldsymbol{x}_i\left(kT_s\right), \end{cases} \tag{2.12}$$

where:

$$F_i = e^{A_i T_s}, \quad G_i = \int_0^{T_s} e^{A_i \sigma} B_i d\sigma, \quad H_i = C_i, \tag{2.13}$$

and, as above, F_i is the state matrix, G_i is the input matrix and H_i is the output matrix. Now, it is possible to obtain the full state-space model in discrete-time writing:

$$\begin{cases} \boldsymbol{x}\left((k+1)T_s\right) = A_{dt}\boldsymbol{x}\left(kT_s\right) + B_{dt}\boldsymbol{u}\left(kT_s\right) \\ \boldsymbol{y}(kT_s) = C_{dt}\boldsymbol{x}(kT_s), \end{cases} \tag{2.14}$$

where the matrices A_{dt}, B_{dt} and C_{dt} are obtained composing the matrices of the single drone in this way considering $N_d = 3$ drones:

$$A_{dt} = \begin{bmatrix} F_1 & 0^{4\times4} & 0^{4\times4} \\ 0^{4\times4} & F_2 & 0^{4\times4} \\ 0^{4\times4} & 0^{4\times4} & F_3 \end{bmatrix}, \quad B_{dt} = \begin{bmatrix} G_1 & 0^{4\times2} & 0^{4\times2} \\ 0^{4\times2} & G_2 & 0^{4\times2} \\ 0^{4\times2} & 0^{4\times2} & G_3 \end{bmatrix},$$

$$C_{dt} = \begin{bmatrix} H_1 & 0^{4\times4} & 0^{4\times4} \\ 0^{4\times4} & H_2 & 0^{4\times4} \\ 0^{4\times4} & 0^{4\times4} & H_3 \end{bmatrix}, \tag{2.15}$$

State and input vectors (omitting the time dependancy) are respectively $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{V}_1 & \boldsymbol{P}_2 & \boldsymbol{V}_2 & \boldsymbol{P}_3 & \boldsymbol{V}_3 \end{bmatrix}^T$ and $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{P}_1^{ref} & \boldsymbol{P}_2^{ref} & \boldsymbol{P}_3^{ref} \end{bmatrix}^T$, where $\boldsymbol{x} \in \mathbb{R}^{12}$ and $\boldsymbol{u} \in \mathbb{R}^6$.
This LTI system corresponds to the control-oriented model and it is used in section (3.4).

Finally, it is introduced a selection matrix $\mathbb{P}_{xy} \in \mathbb{R}^{4N_d \times 2N_d}$ which is defined such that $x(k)\mathbb{P}_{xy} = [x_1(k)^T, \ldots, x_{N_d}(k)^T]^T$. As a result, this matrix selects (x,y) coordinate for each drones.

2.4. Sensors

The drones are equipped with an IMU (Inertial Measurement Unit), provided with filtering algorithms to have a correct estimate of the full state, and a GPS (Global Positioning System) which permits the absolute localization with respect to a global reference frame. LiDAR sensors (Light Detection And Ranging) are also installed on each drone.

Figure 2.3: Example of LiDAR sensor

This sensor is used to perceive the environment and let the drones to localize obstacles during the flight. The readings of this sensor are then elaborated and commuted constraints used in the optimisation problem in section (3.4). Each sensor produces a vector $\gamma(k)$ of $N_r = \frac{2\pi}{\alpha_s}$ measurements, where φ_s corresponds to the angular resolution. The aforementioned measurements can be also expressed as a distance through the vector $d_i(k) = \gamma_i \begin{bmatrix} cos(\varphi_i) & sin(\varphi_i) \end{bmatrix}$, $i = 1, \ldots, N_r - 1$, $d(k) \in \mathbb{R}^{2 \times N_r - 1}$ where d_i represents the distance from the sensor to the closest obstacle in the direction expressed by φ_i. If in a certain direction φ_i no obstacle is detected, the value assigned to γ_i is equal to R_L the maximum range which the sensor can measure. Since the measurements from LiDAR sensors are provided with a frequency of at least 10Hz, it is possible to directly implement them in the algorithm presented in section (3.3).

2.5. Environment

As said previously, a 2D environment is considered. Here, is considered the system of three drones, the tethers, the ground station and the obstacles as it can be seen in Figure (2.4).

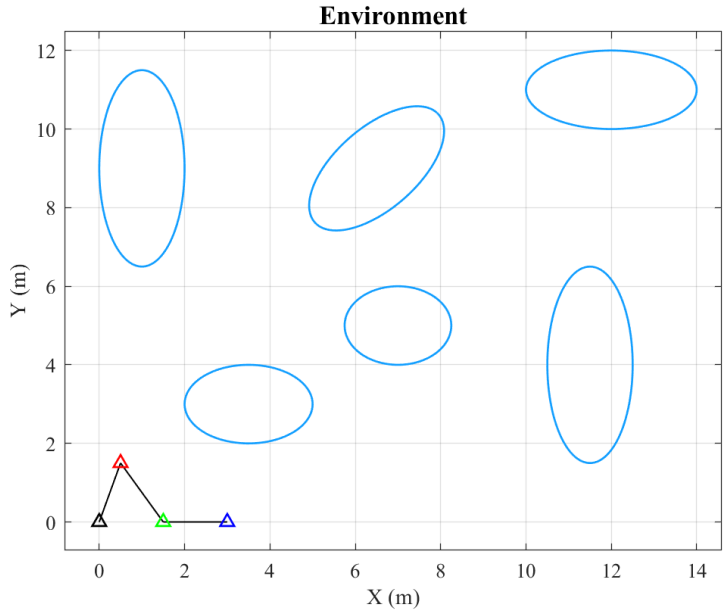

Figure 2.4: A representation of the environment. The three colored triangles represent the drones, while the black one correspond to the ground station. The tether is depicted as the black segments which connects the drones. In blue the obstacles.

To simplify the mathematical formulation of constraints in optimisation problem (see section (3.2)), obstacles are considered as ellipses. In fact, a compact set can describe them, involving the shape matrix H_j in this way:

$$O_j \coloneqq \left\{\chi \in \mathbb{R}^2 : (\chi - \chi_{c_j})^T H_j (\chi - \chi_{c_j}) \leq 1\right\}, \tag{2.16}$$

where H_j, χ_{c_j} represents respectively the geometric shape and the coordinate of the center of the j-th ellipse. Moreover, it is possible to define the set of all obstacles as $O \coloneqq \bigcup_{j=1}^{N_0} O_j$, with N_0 the total number of obstacles. Furthermore, ellipses are used here not only to ease the treatment, but also because is always possible to approximate a set of non-convex obstacles as a set of ellipses as it is shown in Figure (2.5). The information about the nominal environment are collected in a map, where each cell cell has a value representing the occupancy status of that cell. An occupied location is represented as true (1) and a free location is false (0). Using the occupancy map, it is possible to simulate the vector of ranges $\boldsymbol{\gamma}$ (see section (2.4)) considering the pose of the

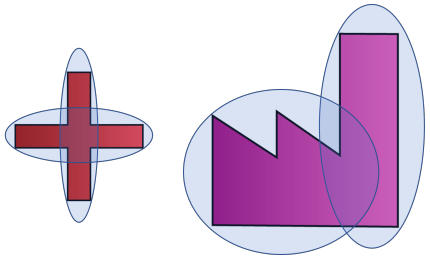

Figure 2.5: A non convex obstacle set approximated as a several convex ones using ellipses.

drone, which is a vector containing the position and the orientation of the drone, and the map itself. Moreover, the sensor has to be initialized with the horizontal angle φ_M, the minimum and maximum angle range, horizontal angular resolution φ_s, and the maximum range R_L, which have been previously defined. An example of map and simulation of LiDAR readings are shown in Figure (2.6).

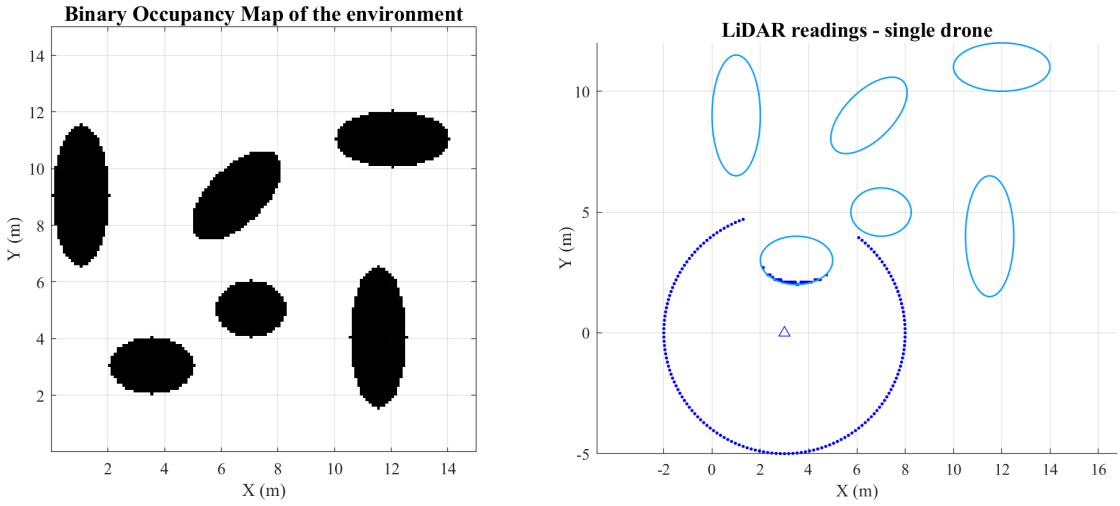

Figure 2.6: A representation of the map of the environment on the left. On the right, the simulation of LiDAR readings. The triangle represents the drone position, the dots are the readings, while in blue are depicted obstacles.

3 | Proposed approach

In this chapter an overview of the entire approach is discussed. First, a brief overview of the problem formulation is given. Then, the off-line mission planner is described: it consists of a non-linear non convex problem which has the aim to find a obstacle-free optimal configuration for the drones starting from the target assigned to the leader one, satisfying constraints related to obstacle avoidance and minimizing a proper cost function. Furthermore, a Linear Program (LP) used to find the optimal maximum ellipse inside merged readings is described, which is then used to obtain polytopic constraints representing obstacle-free regions containing any two pair of connected drones. After that, an online path following algorithm, based on MPC, is presented, this brings the drones to the previously found optimal configuration. At this point, the same offline planner as before with some modifications is used, in order to find a new optimal configuration assigning a new target to the leader drone as well. The approach is repeated at every sampling time and chooses the right strategy to impose to the algorithm, choosing between a backtracking strategy or a new path planning one. In conclusion, after the strategy is chosen, the drones are again brought towards their final configuration.

3.1. Problem formulation

It is useful for the next sections to introduce the concept of configuration of the system. This is a vector gathering the position of all drones $\boldsymbol{C}(t) = \bigcup_{i=1}^{N_d} \boldsymbol{P}_i(t)$. Such configuration is said to be admissible in the environment if the positions of all drones and the tethers connecting them belong to the free space $\boldsymbol{S}_{free} := \mathbb{R}^2 \setminus O$ and the tether lengths are bounded between a maximum value $\bar{l}$ and a minimum one $\underline{l}$:

$$\boldsymbol{P}_i + \alpha\left(\boldsymbol{P}_{i+1} - \boldsymbol{P}_i\right) \in \boldsymbol{S}_{free}, \;\; i = 0, \ldots, N_d, \;\; \forall \alpha \in [0,1] \tag{3.1}$$

$$\underline{l} \leq \left\|\boldsymbol{P}_{i+1} - \boldsymbol{P}_i\right\|_2 \leq \bar{l}, \;\; i = 0, \ldots, N_d - 1. \tag{3.2}$$

The position of the ground station is here defined as P_0 and it coincides with the origin $(0, 0)$. The complete problem can be defined in some steps:

- An offline optimisation problem, receiving as input the set of obstacles O and a target $\boldsymbol{P}_{target}$ assigned to the leader drone, has the aim to find, if it exists, an admissible optimal configuration $\boldsymbol{C}^*$, where this condition is verified: $\boldsymbol{P}^*_{N_d} = \boldsymbol{P}_{target}$.
- An online path following algorithm, based on MPC, has to track the optimal configuration $\boldsymbol{C}^*$, respecting some constraints related to the avoidance of known and possibly unknown obstacles.
- An offline optimisation problem, similar to the one previously mentioned, receiving as inputs a new target $\boldsymbol{P}_{target}$, has the aim to find a new optimal configuration $\boldsymbol{C}^*$. This offline planner is different from the previous one because it tends to the solution trying to move initially only the leader drone, then the number of moved drone is increased if no solution is found, until it moves all the drones.
- A strategy to choose whether to carry out a backtracking process, where the drones are rewound with a particular policy, or a path following algorithm, similar to the previous one.

3.2. Offline path planner

In this section an overview of the offline optimisation problem is given. It presents the algorithm in the basic mathematical formulation, then there is a focus on cost and constraints functions. Subsequently the "perpendicular line method" approach is explained, used to formulate non-linear constraints to keep a certain distance between tethers and obstacles.

This offline path planner produces the vector of optimal configuration $\boldsymbol{C}^* = \begin{bmatrix} \boldsymbol{C}^*_1 \\ \vdots \\ \boldsymbol{C}^*_{N_d} \end{bmatrix}$, where the vector $\boldsymbol{C}^*_i$ represents the coordinates of the target related to drone i.

3.2.1. Mathematical formulation

The initial position of the drones represents the initial condition of the algorithm. For this reason, six optimisation variables (in the case of three drones, assumed here for the sake of clarity and without loss of generality) are needed: $\boldsymbol{x} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & x_3 & y_3 \end{bmatrix}^T$, they represent the (x, y) coordinate of the drones. In this optimisation problem the target of the first drone is given, then, as output are obtained the targets of the other two drones. It is possible to write this problem as a general non-linear constrained Optimization Program

(OP):

$$\min_{\boldsymbol{x}\in\mathbb{R}^6} f(\boldsymbol{x}) \tag{3.3a}$$

$$s.t$$

$$g(\boldsymbol{x}) = 0 \tag{3.3b}$$

$$h(\boldsymbol{x}) \geq 0 \tag{3.3c}$$

with $f : \mathbb{R}^6 \to \mathbb{R}, g : \mathbb{R}^6 \to \mathbb{R}^p$ and $h : \mathbb{R}^6 \to \mathbb{R}^q$.
For simplicity in the discussion these vectors are introduced: $\boldsymbol{x}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$, with $i = 0, \ldots, 3$, where $\boldsymbol{x}_0$ represents the position of the ground station and $\boldsymbol{x}_T = \begin{bmatrix} x_T \\ y_T \end{bmatrix}$ which represents the coordinate of the target. In the cost function (3.3a) two terms are considered. The first is the minimization of the distance between the leader drone and the given target. It reads as follows:

$$f_1(\boldsymbol{x}) = \|\boldsymbol{x}_1 - \boldsymbol{x}_T\|_2^2 \tag{3.4}$$

The second term is related to the minimization of the distance between consecutive drones and it can be written as:

$$f_2(\boldsymbol{x}) = \|\boldsymbol{x}_3 - \boldsymbol{x}_{GS}\|_2^2 + \|\boldsymbol{x}_{GS} - \boldsymbol{x}_2\|_2^2 + \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2 \tag{3.5}$$

The total cost function is the following:

$$f(\boldsymbol{x}) = \begin{bmatrix} \alpha & \beta \end{bmatrix} \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix}, \tag{3.6}$$

where two weights have been introduced to emphasize the minimization of the first term respect the second one (i.e. a possible choice can be $\alpha = 100$ and $\beta = 0.1$).
In this optimisation problem no linear equality or inequality constraints are present.
On the other way around, non-linear inequality constraints (3.3c) have to be mentioned. First, the length of the tethers which connects two consecutive drones must be higher than a fixed minimum distance $\underline{l}$, and it must be less than a fixed maximum distance $\bar{l}$ as

mentioned in section (3.1). In fact, we can write these constraints as:

$$\boldsymbol{h}_1(\boldsymbol{x}) = \begin{bmatrix} -\|\boldsymbol{x}_3 - \boldsymbol{x}_{GS}\|_2^2 + \bar{l}^2 \\ -\|\boldsymbol{x}_{GS} - \boldsymbol{x}_2\|_2^2 + \bar{l}^2 \\ -\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2 + \bar{l}^2 \\ \|\boldsymbol{x}_3 - \boldsymbol{x}_{GS}\|_2^2 - \underline{l}^2 \\ \|\boldsymbol{x}_{GS} - \boldsymbol{x}_2\|_2^2 - \underline{l}^2 \\ \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2 - \underline{l}^2 \end{bmatrix} \tag{3.7}$$

Second, the optimisation variables $\boldsymbol{x}$ can not be inside the obstacles, they must be outside them. Now, it is necessary to do a change of coordinate since the obstacles are rotated ellipses. Generally the formula is:

$$\begin{aligned} X &= (x - x_0)\cos(\alpha) + (y - y_0)\sin(\alpha) \\ Y &= (y - y_0)\cos(\alpha) - (x - x_0)\sin(\alpha), \end{aligned} \tag{3.8}$$

where x_0, y_0 are the coordinate of ellipse center, and α is the angle of rotation with respect x axis, as reported in table 4.1. For example, if six obstacles are present, we have:

$$\begin{aligned} X_{i,j} &= (x_i - x_{0,j})\cos(\alpha_j) + (y_i - y_{0,j})\sin(\alpha_j) \\ Y_{i,j} &= (y_i - y_{0,j})\cos(\alpha_j) - (x_i - x_{0,j})\sin(\alpha_j), \end{aligned} \tag{3.9}$$

where $j = 1, \ldots, 6$ represent the j-th obstacle. At this point it is possible to write the second part of non-linear constraint function as:

$$\boldsymbol{h}_2(\boldsymbol{x}) = \begin{bmatrix} \left(\frac{X_{1,1}}{a_{e_1}}\right)^2 + \left(\frac{Y_{1,1}}{b_{e_1}}\right)^2 - (1+\delta)^2 \\ \vdots \\ \left(\frac{X_{3,6}}{a_{e_6}}\right)^2 + \left(\frac{Y_{3,6}}{b_{e_6}}\right)^2 - (1+\delta)^2 \end{bmatrix}, \tag{3.10}$$

where δ is a user defined parameter, it represents the minimum tolerated value of distance between the optimisation variables and ellipses.
The last constraint that is considered here is that the tethers must not lie inside the obstacles and the distance between obstacle and tether has to be at least σ. The general simplified case can be done with circles, see Figure (3.1):

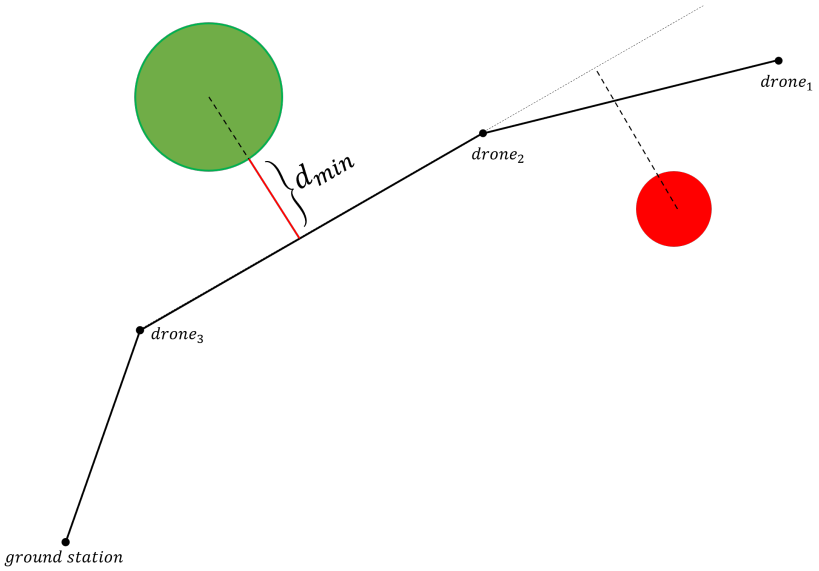

Figure 3.1: An example of perpendicular line method approach with circles.

The distance between the circle and the tethers, which is d_{min}, has to be greater than σ, a user defined minimum tolerated distance. The constraint is active on that segment only if the perpendicular line has an intersection with the tethers. This general method has been revised because here, ellipse are involved. The line parallel to the tether passing through the center of ellipse is traced, consequently discretized points on this line are found. We call d_{min} the distance between the ellipse and the tether and it is calculated as:

$$d_{min} = d_c - d_e, \tag{3.11}$$

where d_c is the distance between the center of j-th ellipse and the i-th tether, while d_e is the distance between the p-th discretized point and the j-th ellipse along the perpendicular line of i-th tether passing to that discretized point. This distance, has to be calculated for each discretized point $p = 1, \dots, N_{pe}$, for each obstacle $j = 1, \dots, N_O$, and for each tether $i = 0, \dots, N_d - 1$, then, it is imposed to be higher than σ. An example of this approach is described in Figure (3.2).

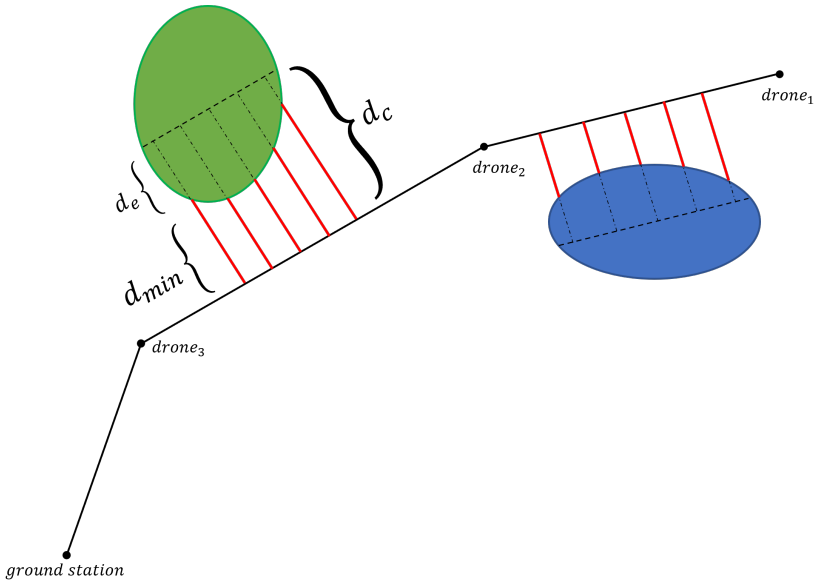

Figure 3.2: An example of perpendicular line method approach with ellipse

Similarly to the circle approach, the constraint is considered active only if the perpendicular line has an intersection with the tether. The third part of non-linear inequality constraint function can be constructed as:

$$\boldsymbol{h}_3(\boldsymbol{x}) = \begin{bmatrix} d_{min,1} - \sigma \\ \vdots \\ d_{min,N_e} - \sigma \end{bmatrix}, \tag{3.12}$$

where $N_e = N_{pe} \times N_O \times (M-1)$ is calculated as the product between the discretized points, the number of ellipses and the number of segments. Finally, the overall vector can be combined as:

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{x}) \\ \boldsymbol{h}_2(\boldsymbol{x}) \\ \boldsymbol{h}_3(\boldsymbol{x}) \end{bmatrix} \tag{3.13}$$

3.3. Convex Approximation of Free Space

This approach is based on a constant sampling rate, where k is the discrete-time variable. An obstacle-free area is needed by the path following online algorithm. In particular, a set containing two drones is obtained. First, the set containing all the LiDAR measurements of the i-th drone at time k is defined as $L_i(k) \coloneqq \{d_0(k), \ldots, d_{N_r-1}(k) \in \mathbb{R}^2\}$. Another example of independent LiDAR readings is given in Figure 3.3.

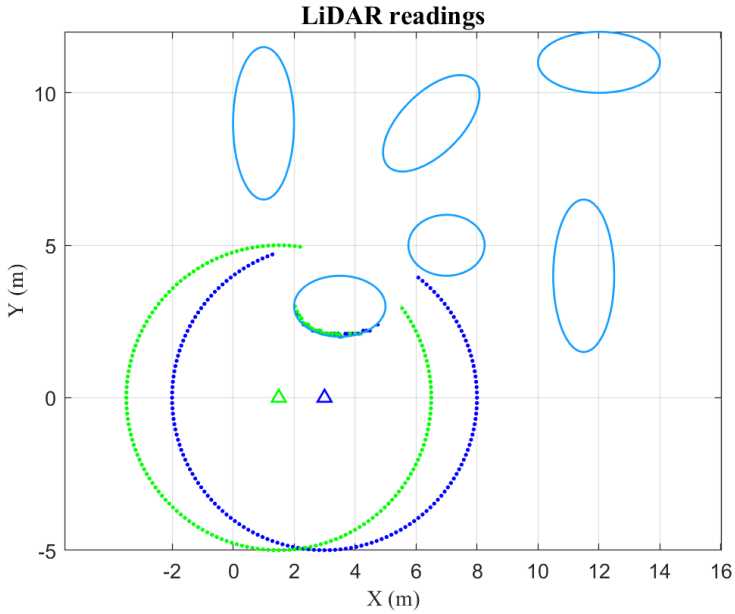

Figure 3.3: A representation of independent LiDAR readings. The triangles represent drone i and drone i+1, the colored dots represent the readings L_i and L_{i+1}, while in blue are reported obstacles.

At this point, to calculate the non-convex area defined by the LiDAR readings of two consecutive drones i and $i+1$, the set of overlapping measurements is computed:

$$\begin{aligned} L_d(k) = \{&d_m(k) \in L_i(k),\ \ d_n(k) \in L_{i+1}(k) : \\ &\|d_m(k) - \boldsymbol{P}_{i+1}(k)\|_2 < R_L \wedge \|d_m(k) - \boldsymbol{P}_i(k)\|_2 = R_L, \\ &\|d_n(k) - \boldsymbol{P}_i(k)\|_2 < R_L \wedge \|d_n(k) - \boldsymbol{P}_{i+1}(k)\|_2 = R_L, \\ &\forall m, n = 0, \ldots, N_r - 1.\} \end{aligned} \tag{3.14}$$

To summarize, given two consecutive drones only the interior points of the readings are selected here. Starting from this, it is possible to have the merged readings as:

$$L_{i,i+1}(k) = L_i(k) \cup L_{i+1}(k) - L_d(k) \tag{3.15}$$

After that, the cardinality of the aforementioned merged LiDAR readings $|L_{i,i+1}(k)|$ is denoted as $L_{i,i+1}$. A representation of merged LiDAR readings is shown in Fig 3.4:

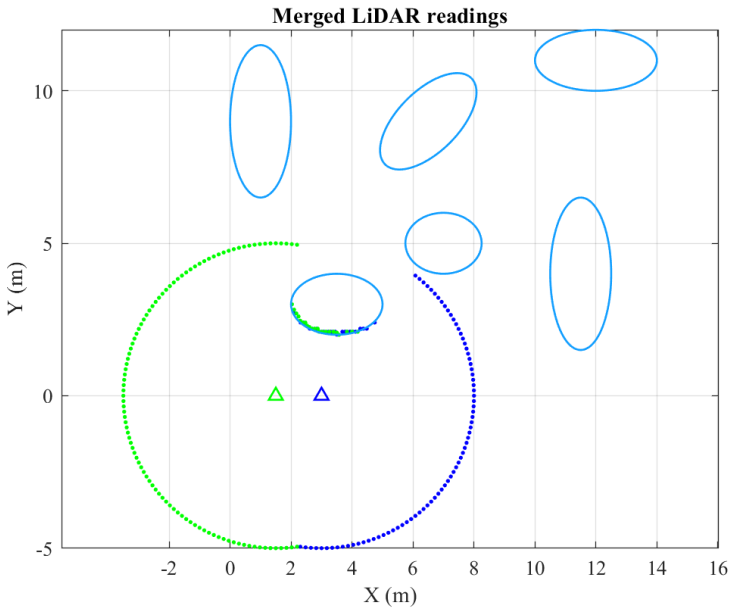

Figure 3.4: A representation of merged LiDAR readings. The triangles represent the drone i and $i+1$, while the colored dots are the merged LiDAR readings $L_{i,i+1}$. In blue are represented the obstacles.

Starting from the set $L_{i,i+1}(k)$, the goal is to obtain the largest convex set containing the two vehicles. To obain this, the following steps are executed: first, the largest ellipse in the readings is found, through a linear optimisation problem; second, from ellipse, an under approximation of polytopic constraints are retrieved, then, the vertices of the polytope are iteratively expanded to have the maximum area. To simplify the problem, it is assumed that the ellipse is rotated with the same angle α which the cable forms with x axis. This problem can be structured as a linear program of the following form:

$$\min_{k1,k2} \boldsymbol{c}'\boldsymbol{k} \tag{3.16a}$$

$$s.t.$$

$$A\boldsymbol{k} \leq \boldsymbol{b} \tag{3.16b}$$

To find the cost function (3.16a), is necessary to recall the canonical form of a rotated ellipse:

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{y_1}{b}\right)^2 = 1, \tag{3.17}$$

where x_1 and y_1 have the same expression of (3.8):

$$\begin{aligned} x_1 &= (x - x_0)\cos(\alpha) + (y - y_0)\sin(\alpha) \\ y_1 &= (y - y_0)\cos(\alpha) - (x - x_0)\sin(\alpha) \end{aligned} \tag{3.18}$$

At this point, substituting $k_1 = \frac{1}{a^2}$ and $k_2 = \frac{1}{b^2}$ a linear expression is obtained, with only two unknown linear variables:

$$k_1 x_1^2 + k_2 y_1^2 = 1 \tag{3.19}$$

The aim of this optimisation problem is to find the maximum ellipse, in other words a minimization of k_1 and k_2 is needed. In fact it corresponds to a maximization of a and b, which are ellipse parameters. Since these parameters have been maximized together, the cost function can be written easily considering vector $\boldsymbol{c} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and vector $\boldsymbol{k} = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$, which represents the vector of optimisation variables. The first set of constraints considered in this OP is that the previously found merged readings must be outside the ellipse. Mathematically this is expressed as:

$$k_1 x_{1,i}^2 + k_2 y_{1,i}^2 \geq 1 \tag{3.20}$$

where $x_{1,i}, y_{1,i},\ i = 1, \ldots, N_r - 1$ have again the same expression written in (3.18) and x, y are coordinate of LiDAR readings d_i^x, d_i^y, as mentioned in section (2.4), where given reading γ_i, it is possible to obtain $d_i^x(k) = \gamma_i cos\varphi_i$ and $d_i^y = \gamma_i sin\varphi_i$. The canonical form of matrix of constraints is $A\boldsymbol{x} \leq \boldsymbol{b}$, in this case it is possible to write $A_1\boldsymbol{k} \leq \boldsymbol{b}_1$ where

$$A_1 = \begin{bmatrix} -x_{1,1}^2 & -y_{1,1}^2 \\ \vdots & \vdots \\ -x_{1,N_r}^2 & -y_{1,N_r}^2 \end{bmatrix}, \boldsymbol{b}_1 = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \tag{3.21}$$

with N_r total number of readings. The second set of constraints which is considered is related to the position of the two drones. The drones i and $i+1$ referred to LiDAR readings set $L_{i,i+1}(k)$, need to be inside the ellipse. Mathematically, this means

$$\begin{aligned} k_1 x_{1,i}^2 + k_2 y_{1,i}^2 &\leq 1 \\ k_1 x_{1,i+1}^2 + k_2 y_{1,i+1}^2 &\leq 1 \end{aligned}$$

where in this case

$$\begin{aligned} x_{1,i} &= (P_i^x - x_0)\cos(\alpha) + (P_i^y - y_0)\sin(\alpha) \\ y_{1,i} &= (P_i^y - y_0)\cos(\alpha) - (P_i^x - x_0)\sin(\alpha) \end{aligned}$$

for i-th drone, and

$$\begin{aligned} x_{1,i+1} &= (P^x_{i+1} - x_0)\cos(\alpha) + (P^y_{i+1} - y_0)\sin(\alpha) \\ y_{1,i+1} &= (P^y_{i+1} - y_0)\cos(\alpha) - (P^x_{i+1} - x_0)\sin(\alpha) \end{aligned} \tag{3.22}$$

for the second drone. In matrix form, these expressions can be resumed as

$$A_2 = \begin{bmatrix} x^2_{1,i} & y^2_{1,i} \\ x^2_{1,i+1} & y^2_{1,i+1} \end{bmatrix}, \boldsymbol{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.23}$$

Third, the last set of constraints is used to enforce a specific behaviour. In fact, a real ellipse must have positive semi-major and semi-minor axes, which basically means $a_e > 0$ and $b_e > 0$. In addition, the semi-major axis is preferred to be higher than the semi-minor one: $\frac{a_e}{b_e} > q$ where q is a proper user-defined value. These constraints, as a function of optimisation variables, can be written as:

$$\begin{aligned} k_1 &> 0 \\ k_2 &> 0 \\ k_1 q^2 &< k_2 \end{aligned} \tag{3.24}$$

Now, in matrix form

$$A_3 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ q^2 & -1 \end{bmatrix}, \boldsymbol{b}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.25}$$

At this point, linear constraints can be written gathering the previous part as:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{bmatrix} \tag{3.26}$$

The result of this process is shown in Figure (3.5).

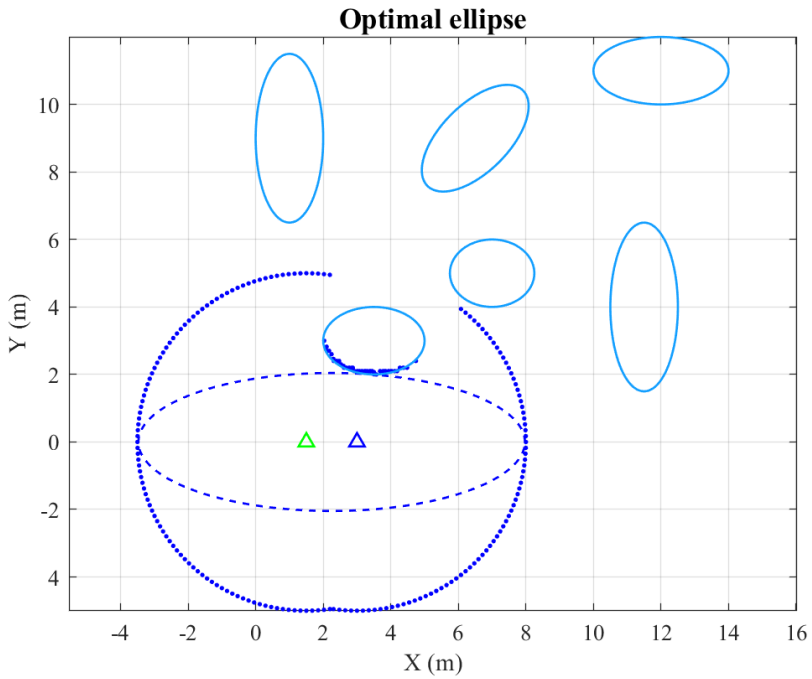

Figure 3.5: A representation of maximum ellipse inside LiDAR readings $L_{i,i+1}$. The maximum ellipse is represented as the dashed line, the triangles represent drone i and $i+1$, while the obstacles are depicted in blue.

It can be seen that this ellipse, not only is aligned with the tether which connects drone i and drone $i+1$, but also does not contain any LiDAR readings. Now, optimal ellipses previously found are discretized in N_p user-defined points selecting them with a constant angular distance one from each other. These points represent the vertices of the initial polytope, which is the under-approximation of the free space, $D_j(k), j = 1, \ldots, N_d$, which subsequently has to be expanded, as mentioned in section (3.1). This process begins with the expansion of one vertex taking the line which connects the vertex to the center of ellipse as direction of expansion. The vertex is moved away by a user-defined variable ε, which represents how far the vertex is moved, from its previous position along the direction of expansion. If the expanded vertex does not encounter any obstacle, that vertex is frozen, otherwise it is moved back to its previous position, and this position is kept. The operation stops when all vertices cannot be moved away from the center or have reached the maximum distance from it. To take into account the dimensions of vehicles, its maximum encumbrance is removed from the LiDAR readings before the computation of the convex approximation of the free space. Figure (3.6) shows the process of expansion:

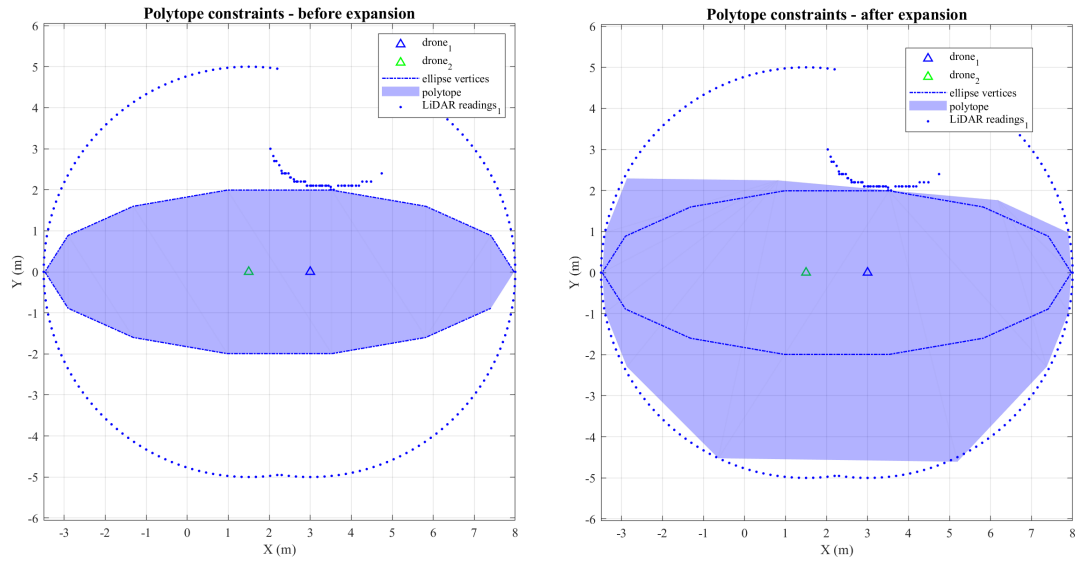

Figure 3.6: An example of the process of expansion of a polytope starting from the vertices of optimal ellipse.

In conclusion, expanded vertices are converted in inequality constraints in the form $A\boldsymbol{x} \leq \boldsymbol{b}$ where $\boldsymbol{x}$ is a vector formed by (x, y) general coordinates. The outputs of this process are matrix A_j and vector $\boldsymbol{b}_j$.

Remark: By construction, the proposed algorithm ensures that the current position P_i^x, P_i^y of consecutive drone i and $i+1$ and their tether is contained in the convex under-approximation of the free space $D_j(k)$ which can be considered a safe set. In conclusion, N_d convex polytopes computed at time k for each pair of drones are collected in the set:

$$S(k) = \{D_j(k), \;\; j = 1, \ldots, N_d\} \tag{3.27}$$

Once the polytope is created, it is necessary to reduce it, for security reason. In fact, it is created in such a way that its vertices are very close to the readings, which in some cases correspond to obstacles, as it can be seen in Figure (3.6). The aim of this reduction process is to have an offset which guarantees at least a distance, which is called $d_{emergency}$, between the polytope and the readings. It is important to remark that this process is done only when the drones are far away from their respective targets. The process can be explained in few steps. First, the equation of the distance from a point to a line is introduced:

$$d = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}, \tag{3.28}$$

where a, b and c are the coefficients of a line in canonical form: $ax + by + c = 0$. Secondly,

recalling that a polytope is a set of linear constraints, it is possible to write:

$$\begin{bmatrix} a_{p,1} & b_{p,1} \\ \vdots & \vdots \\ a_{p,i} & b_{p,i} \\ \vdots & \vdots \\ a_{p,N_e} & b_{p,N_e} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} c_{p,1} \\ \vdots \\ c_{p,i} \\ \vdots \\ c_{p,N_p} \end{bmatrix}, \tag{3.29}$$

where a_p, b_p and c_p represent the i-th coefficients of the component of polytope inequality and N_p is the total number of them. At this point, for all i components a new coefficient $c^*_{p,i}$, which is responsible for the offset, has to be founded. The distance from a generic point d (i.e. the position of the drone) to the i-th line which composes the polytope is calculated, after that, this distance is perturbed as $d_p = d - d_{emergency}$. Now, from (3.28), $c^*_{p,i}$ is computed as follows:

$$a_{p,i}x + b_{p,i}y - c_{p,i} \geq 0 \implies c^*_{p,i} = d_p\sqrt{a^2_{p,i} + b^2_{p,i}} - a_{p,i}x - b_{p,i}y \tag{3.30}$$

$$a_{p,i}x + b_{p,i}y - c_{p,i} \leq 0 \implies c^*_{p,i} = -d_p\sqrt{a^2_{p,i} + b^2_{p,i}} - a_{p,i}x - b_{p,i}y$$

To conclude, the reduced polytope in canonical inequality form is:

$$\begin{bmatrix} a_{p,1} & b_{p,1} \\ \vdots & \vdots \\ a_{p,i} & b_{p,i} \\ \vdots & \vdots \\ a_{p,N_p} & b_{p,N_p} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} -c^*_{p,1} \\ \vdots \\ -c^*_{p,i} \\ \vdots \\ -c^*_{p,N_p} \end{bmatrix} \tag{3.31}$$

3.4. Online Path Following

This section explains how a Non-linear Model Predictive Control (NMPC) algorithm works well with this type of problem and what are its components. First, a mathematical formulation of optimisation problem is given, with a brief view on cost function and constraints, then, path following strategy used to reach the targets of the drone is shown. Finally, the concept of recursive feasibility is explained.

3.4.1. Mathematical formulation

A real-time path following strategy is now necessary to bring the drones to the optimal configuration $\boldsymbol{C}^*$ obtained in section (3.2). In this algorithm, LiDAR readings, which are provided in real-time, are used to derive an approach which is able to react to unexpected obstacles. We consider a trajectory which is able to stop the vehicle inside $S(k)$, the obstacle free region, within the prediction horizon $N \in \mathbb{N}$. The MPC algorithm consists in solving at each iteration time a Finite Horizon Optimal Control Problem (FHOCP). The decision variables are optimisation variables, which can be collected in a vector $U = [\boldsymbol{u}(0|t)^T, ..., \boldsymbol{u}(N|t)^T]^T \in \mathbb{R}^{2N_d(N+1)}$. Denoting the state of the system (2.14) with $\boldsymbol{x}(j|k)$, at time $k+j$ and $\boldsymbol{x}_g$ as the vector of goal, the OP reads:

$$\min_{U} J(\boldsymbol{x}(k), \boldsymbol{x}_g(k)) \tag{3.32a}$$

$$s.t.$$

$$\boldsymbol{x}(j+1|k) = A_{dt}\boldsymbol{x}(j|k) + B_{dt}\boldsymbol{u}(j|k), \ \ \forall j \in \mathbb{N}_0^{N-1} \tag{3.32b}$$

$$\boldsymbol{x}(0|k) = \boldsymbol{x}_0 \tag{3.32c}$$

$$\boldsymbol{u}(0|k) = \boldsymbol{u}_0 \tag{3.32d}$$

$$-\overline{\boldsymbol{A}} \leq K_{vel}(K_{pos}(\boldsymbol{u}(j|k) - \boldsymbol{x}_{i_{1:2}}(j|k)) - \boldsymbol{x}_{i_{3:4}}(j|k)) \leq \overline{\boldsymbol{A}}, \ \ \forall j \in \mathbb{N}_0^{N-1}, \ \ \forall i \in \mathbb{N}_1^{N_d} \tag{3.32e}$$

$$-\overline{\boldsymbol{V}} \leq \boldsymbol{x}_{i_{(3:4)}}(j|k) \leq \overline{\boldsymbol{V}}, \ \ \forall j \in \mathbb{N}_0^{N}, \ \ \forall i \in \mathbb{N}_1^{N_d} \tag{3.32f}$$

$$\boldsymbol{x}_{i_{1,2}}(j|k) \in D_i \cap D_{i+1} \ \ \forall j \in \mathbb{N}_0^{N}, \ \ \forall i \in \mathbb{N}_1^{N_d-1} \tag{3.32g}$$

$$\underline{d}^2 \leq \left\| \boldsymbol{x}_{i+1_{(1:2)}} - \boldsymbol{x}_{i_{(1:2)}} \right\|_2^2, \ \ \forall i \in \mathbb{N}_1^{N_d-1} \tag{3.32h}$$

$$\boldsymbol{x}_{N_d}(j|k) \in D_{N_d}, \ \ \forall j \in \mathbb{N}_1^{N} \tag{3.32i}$$

$$\boldsymbol{x}_{i_{(3,4)}}(N|k) = \boldsymbol{0}^{2\times 1}, \ \ \forall i \in \mathbb{N}_1^{N_d}, \tag{3.32j}$$

Where all inequalities and equalities are element-wise, $\mathbb{N}_a^b = \{n \in \mathbb{N} |\ \ a \leq n \leq b\}$. In the FHOCP, the stage cost (3.32a) expresses a tracking criterion. In fact, it can be written as:

$$J(\boldsymbol{x}(k), \boldsymbol{x_g}(k)) = \sum_{j=1}^{N}\sum_{i=1}^{N_d} \left\| Q(\boldsymbol{x}_{i_{(1:2)}}(j|k) - \boldsymbol{x}_{g_i}(k)) \right\|_2^2 + \sum_{j=1}^{N} \|T_u(\boldsymbol{u}(j|k) - \boldsymbol{u}(j-1|k))\|_2^2, \tag{3.33}$$

where $Q \in \mathbb{R}^{3\times 3}$, $T_u \in \mathbb{R}^{6\times 6}$ are positive-definite weighting matrices. In (3.33) two terms are minimized: the square of the error between a desired state and the simulated, and the square of the difference between two consecutive inputs. The vector of state goals $\boldsymbol{x}_g$

is subsequently introduced in the next subsection "path following".

3.4.2. Path following

The aim of this subsection is to explain the strategy of path following. First, the concept of path is introduced. Considering a $2D$ space we can define a set of points $\boldsymbol{Q}_i$, $i = 0, \ldots, M$. Then, it is possible to define a set of lines l_i, $i = 0, \ldots, M-1$, which connects the points. In particular l_i connects Q_i and Q_{i+1}. Each line l_i is also defined by its angle of rotation Ψ_i with respect x axis, taking a positive angle with a counterclockwise rotation. An example of points connected by lines is represented in Figure (3.7).

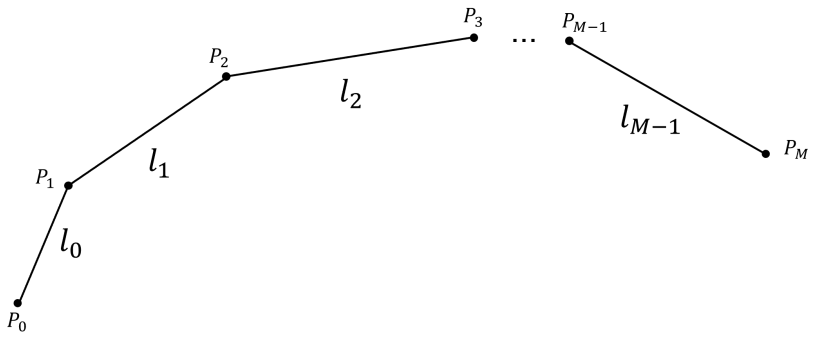

Figure 3.7: A Representation of points connected by lines

We define a path as the set of all intermediate points s_i , $i = 0, \ldots, M_P$ on the lines such that each point is sampled every τ meters (with τ user-defined variable), starting from the initial point S_0, to the final one S_M. In other words, two consecutive points s_i and s_{i+1} have always the same distance τ, in addition the first point of the path is $s_0 = S_0$, while the final point of the path is $s_{M_P} = S_M$. The matrix related to the path is called $\Gamma \in \mathbb{R}^{2 \times M_P}$. It contains (x, y) coordinates of every points of the path and can be expressed in implicit form as:

$$\Gamma = \begin{bmatrix} s_0^x & s_0^y \\ \vdots & \vdots \\ s_i^x & s_i^y \\ \vdots & \vdots \\ s_{M_P}^x & s_{M_P}^y \end{bmatrix}, \tag{3.34}$$

while in explicit form it is described by:

$$\Gamma = \begin{bmatrix} s_0^x & s_0^y \\ S_0^x + \tau\cos(\Psi_0) & S_0^y + \tau\sin(\Psi_0) \\ S_0^x + 2\tau\cos(\Psi_0) & S_0^y + 2\tau\sin(\Psi_0) \\ \vdots & \vdots \\ S_1^x + \mu_1\tau\cos(\Psi_1) & S_1^y + \mu_1\tau\sin(\Psi_1) \\ \vdots & \vdots \\ S_2^x + \mu_2\tau\cos(\Psi_2) & S_2^y + \mu_2\tau\sin(\Psi_2) \\ \vdots & \vdots \\ S_{M-1}^x + \mu_{M-1}\tau\cos(\Psi_{M-1}) & S_{M-1}^y + \mu_{M-1}\tau\sin(\Psi_{M-1}) \\ \vdots & \vdots \\ s_{M_P}^x & s_{M_P}^y \end{bmatrix}, \tag{3.35}$$

where μ_i, $i = 1, \ldots, M-1$ is a coefficient calculated such that the distance between the point with this coordinates $\begin{bmatrix} S_i^x + \mu_i\tau\cos(\Psi_i) & S_i^y + \mu_i\tau\sin(\Psi_i) \end{bmatrix}$ and the previous one is exactly τ. An example of path, starting from the Figure (3.7), is depicted in Figure (3.8).

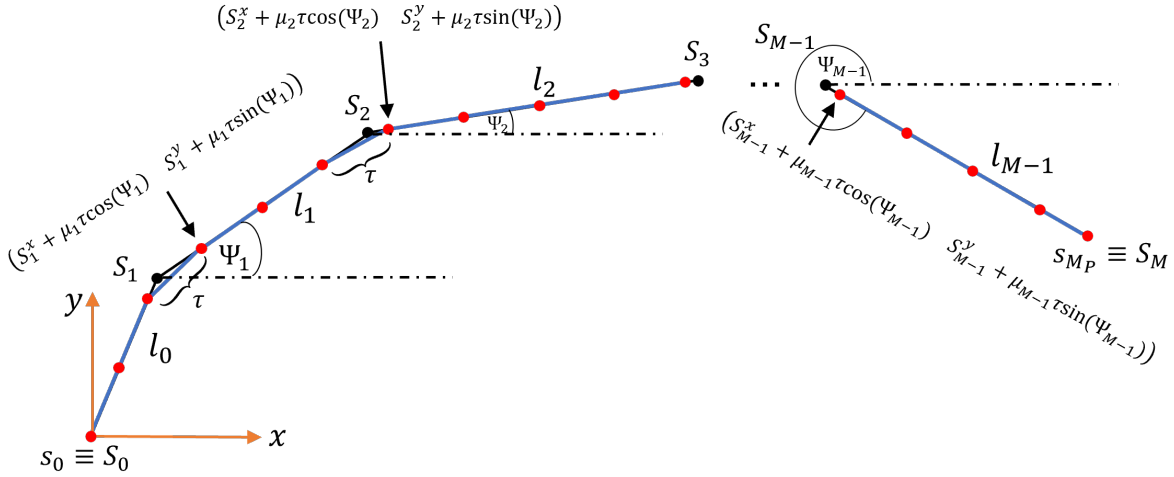

Figure 3.8: An example of path

Remark1: It is important to choose τ small (i.e. $0.5m$), otherwise the path is not created correctly.

Remark2: It can happen that the distance between the second last point s_{M_P-1} and the last one s_{M_P} (see Figure 3.8) is not equal to τ but this is not a problem.

In our case, recalling the section (3.2), we obtain from the offline planner the vector of optimal configuration $\boldsymbol{C}^*$. In fact, it is possible to create a path, considering as initial point the position of the ground station $\boldsymbol{P}_0$ and as the final point the target of the leader

drone $\boldsymbol{C}^*_{N_d}$. The matrix related to the path is:

$$\Gamma = \begin{bmatrix} P_0^x & P_0^y \\ P_0^x + \tau\cos(\Psi_0) & P_0^y \tau\sin(\Psi_0) \\ P_0^x + 2\tau\cos(\Psi_0) & P_0^y + 2\tau\sin(\Psi_0) \\ \vdots & \vdots \\ C_3^{*,x} + \mu_1\tau\cos(\Psi_1) & C_3^{*,y} + \mu_1\tau\sin(\Psi_1) \\ \vdots & \vdots \\ C_2^{*,x} + \mu_2\tau\cos(\Psi_2) & C_2^{*,y} + \mu_2\tau\sin(\Psi_2) \\ \vdots & \vdots \\ C_{N_d}^{*,x} & C_{N_d}^{*,y} \end{bmatrix}, \tag{3.36}$$

where Ψ_i, $i = 0, \ldots, N_d - 1$ corresponds to angle between tether i and x-axis, τ is a user-defined variable and μ_i is a coefficient which guarantees that all the points on the path are spaced by τ meters. To choose the proper value to assign as drone goal, another vector is needed: $\boldsymbol{t} = \begin{bmatrix} t_1 & \ldots & t_{N_d} \end{bmatrix}^T$, where t_i, $i = 1, \ldots, N_d$ represents a particular row of matrix Γ. To have the drones well spaced in the first iteration, we have to properly initialize the vector $\boldsymbol{t}$ and the vector $\boldsymbol{x}_{g_i}$, where $\boldsymbol{x}_{g_i}$ represents (x, y) coordinates of the goal of drone i. The vector of goals $\boldsymbol{x}_g$ is defined as $\boldsymbol{x}_g = \begin{bmatrix} \boldsymbol{x}_{g_1}^T & \ldots & \boldsymbol{x}_{g_{N_d}}^T \end{bmatrix}^T$. Assuming τ=0.5m an example can be

$$\boldsymbol{t} = \begin{bmatrix} \bar{t} + 4(N_d - 1) \\ \vdots \\ \bar{t} + 4 \\ \bar{t} \end{bmatrix}, \boldsymbol{x}_{g_i} = \begin{bmatrix} \Gamma(t(i)) \end{bmatrix}^T,$$

where, the expression $\Gamma(t(i))$ means to select the row $t(i)$, $i = 1, \ldots, N_d$ from the matrix Γ. Choosing the value $\bar{t}$ means to select a proper point of the path as goal for the drone$_{N_d}$. When all the distances between the position of drones and their relative goals is lower than a tolerance value the vector $\boldsymbol{t}$ is increased to assign the next points on the path: $\boldsymbol{t} = \boldsymbol{t} + \upsilon$, where υ represent the increment. A practical example is presented in Figure (3.9).

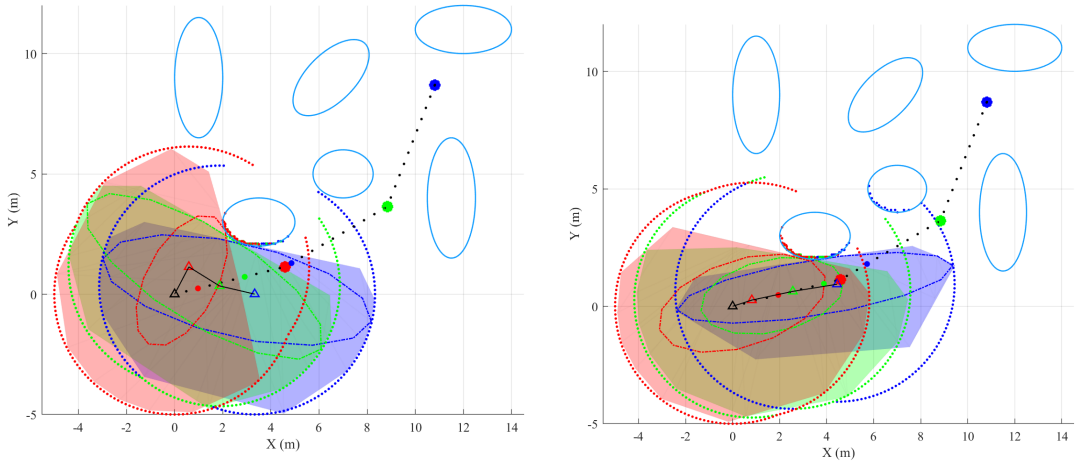

Figure 3.9: A representation of path. Small black dots are the points of the path, while colored dots represent the goal and big colored dots depict the targets. The drone are described as colored triangles, while the black one is the ground station. In blue the obstacles. The goals of the drones are changed at iteration $k+1$ (on the right) whenever the goals are reached by the drones at iteration k (on the left). In this example $v = 2$.

When the drone i is close to its target $\boldsymbol{C}_i^*$, then the target $\boldsymbol{C}_i^*$ is assigned as goal $\boldsymbol{x}_{g_i}$, which essentially means $\boldsymbol{x}_{g_i} = \boldsymbol{C}_i^*$, instead of assigning a point on the path. In Figure (3.10) it is possible to see an example.

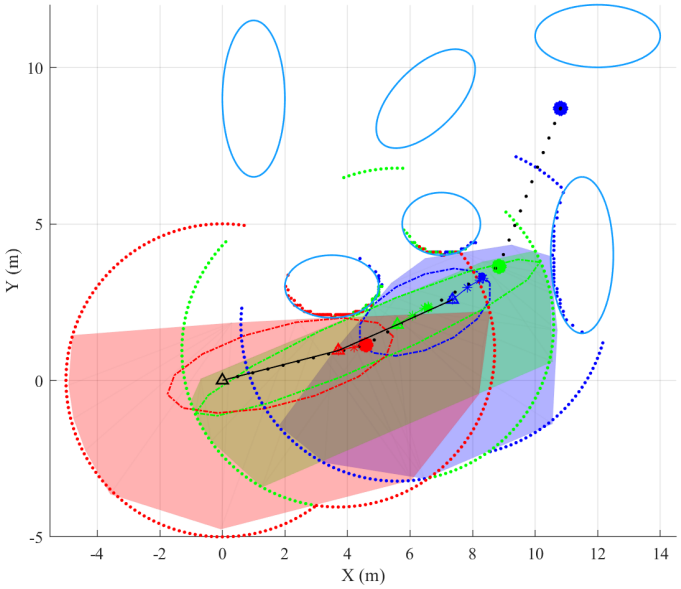

Figure 3.10: An example of path following, with drone i close to target $\boldsymbol{C}_i^*$. In this case drone 3 is close to $\boldsymbol{C}_3^*$, so as said $\boldsymbol{x}_{g_3} = \boldsymbol{C}_3^*$.

The path following algorithm terminates when all the targets are reached by the drones and the velocity of each drone is equal to zero.

3.4.3. Obstacle check

In this subsection it is described a function whose aim is to find when leader drone has to stop because of unexpected obstacle encountered on the path which blocks its motion and consequently the motion of the entire chain of drones. The inputs are the matrices of merged LiDAR readings obtained in section (3.3): $L_{i,i+1}$. Then, a filtering process is done, obtaining only the readings which are not at maximum distance R_L. It is necessary to select readings which respect this condition:

$$\begin{aligned} L_{d,i,i+1}(k) = \{&d_m(k) \in L_{i,i+1}(k) : \\ &\|d_m(k) - \boldsymbol{P}_{i+1}(k)\|_2 < R_L \\ &\|d_m(k) - \boldsymbol{P}_{i}(k)\|_2 < R_L \\ &\forall m = 0, \dots, N_r - 1.\} \end{aligned} \tag{3.37}$$

Moreover, a new matrix of readings is obtained combining them as

$$L_n(k) = \begin{bmatrix} L_{d,0,1}(k) \\ \vdots \\ L_{d,N_d-1,N_d}(k) \end{bmatrix}, \tag{3.38}$$

where the matrix $L_n(k) \in \mathbb{R}^{N_{L_n} \times 2}$ contains all merged readings which are not at maximum distance. The next step is to create through linear inequalities a triangle with these vertices: $\boldsymbol{x}_{g_1}$, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$. Since these three points tend to be aligned for the nature of path following itself, it is necessary, only for this scope, to considered a perturbed position of leader drone:

$$\begin{aligned} P_{1_p}^x &= P_1^x + \lambda \\ P_{1_p}^y &= P_1^y + \lambda \end{aligned} \tag{3.39}$$

where λ is a user-defined parameter and has to be chosen very small, an example can be $tol \leq 0.1$. These points are collected in a matrix:

$$V_{triangle} = \begin{bmatrix} P_{1_p}^x & P_{1_p}^y \\ P_2^x & P_2^y \\ x_{g_1}^x & x_{g_1}^y \end{bmatrix} \tag{3.40}$$

Then, a set of linear inequalities is computed starting from the vertices of the triangle (3.40)

$$A_{triangle}\boldsymbol{x} \leq \boldsymbol{b}_{triangle} \tag{3.41}$$

At this point the idea is to know when a reading is inside the triangle. For this purpose, a linear program is built to have a feasibility check. In fact, every reading of matrix L_n has to be checked. If a point is inside the triangle, it means that there is an obstacle, consequently the path following algorithm is stopped. Otherwise, the triangle is considered free. The LP has to be solved for every LiDAR reading $L_{n,i}$, $i = 1, \dots, N_{L_n}$, and it can be written in this form:

$$\min_{x,y} const \tag{3.42a}$$

$$s.t.$$

$$A_{triangle} \begin{bmatrix} x \\ y \end{bmatrix} \leq \boldsymbol{b}_{triangle} \tag{3.42b}$$

$$x_0 = \begin{bmatrix} L^x_{n,i} \\ L^y_{n,i} \end{bmatrix} \tag{3.42c}$$

This function is not always active, to prevent unwanted stops. It works when the distance between leader drone and its goal is relatively small, otherwise, as we can see in Figure (3.11), it can happen that the triangle is not considered empty in wrong cases.

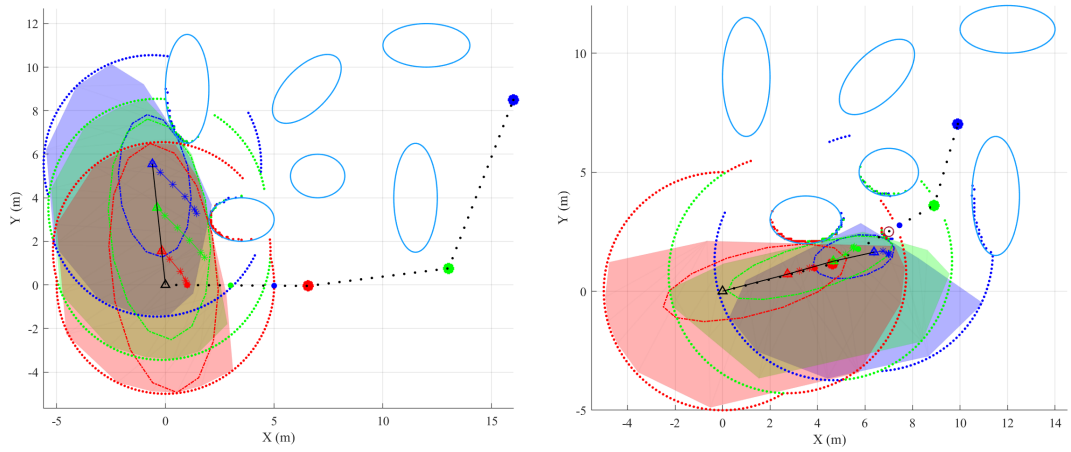

Figure 3.11: On the left, an example of wrong obstacle check. The triangle is not considered empty because the check is always done, even if the leader drone is not close to its target. On the right, it is represented the proper functioning of obstacle check function. In fact, in this case, the function is activated only when the distance between the leader and its target is less than a tolerance.

When an obstacle is checked, the actual position of the drones becomes their goal: $\boldsymbol{x}_{g_i} = \boldsymbol{P}_i$, until they stop moving.

3.4.4. Constraints

The first constraints considered in OP (3.32) are related to the convex set where the drones can move. As it is seen previously, polytopes are a set of inequality constraints and are created according to the rule of "visibility", it means that generally the drone i has to move in its polytope and in the polytope of the i+1 drone, $i = 1, \dots, N_d - 1$, while the drone$_{N_d}$ can move in polytope D_{N_d}. To express this concept gathered matrices are used:

$$A_{p,i,i+1} = \begin{bmatrix} A_{p,i} \\ A_{p,i+1} \end{bmatrix}, \boldsymbol{b}_{p,i,i+1} = \begin{bmatrix} \boldsymbol{b}_{p,i} \\ \boldsymbol{b}_{p,i+1} \end{bmatrix}, \tag{3.43}$$

where $i = 1, \dots, N_d - 1$, and $A_{p,i}, A_{p,i,i+1} \in \mathbb{R}^{N_p \times 12}$, $b_{p,i}, b_{p,i,i+1} \in \mathbb{R}^{N_p}$ are matrix which extract a proper coordinate of the drone. In the first path following, the condition imposed to create polytope constraints are summarized in (3.32g),(3.32i). In addition, when the drone i is near the target $\boldsymbol{C}_i^*$, the drone $i-1$ can move freely in its polytope.
Second, constraints related to the model, (3.32b) are the same seen in Chapter 2 (see Equation (2.14)). An important part of the optimisation problem is the initialization of state (3.32c) and inputs variables (3.32d). At time $k = 0$, $\boldsymbol{x}_0$ takes into account the initial position of drones used in offline optimisation problem $\boldsymbol{x}_{i_{init}}$, while initial velocity of each drone is sat equal to zero for simplicity.

$$\boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{P}_1(0) & \boldsymbol{V}_1(0) & \boldsymbol{P}_2(0) & \boldsymbol{V}_2(0) & \boldsymbol{P}_3(0) & \boldsymbol{V}_3(0) \end{bmatrix}^T$$
$$\boldsymbol{P}_i(0) = \begin{bmatrix} x_{i_{init}} \\ y_{1_{init}} \end{bmatrix}^T, \boldsymbol{V}_i(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T, \quad \forall i = 1, \dots, N_d$$

When $k \geq 1$ the initialization changes in

$$\boldsymbol{x}_0 = A_{dt}\boldsymbol{x}(0|k-1) + B_{dt}\boldsymbol{u}(0|k-1)$$

This means that the initialization of states is done with the optimal trajectory obtained in the last iteration, thus taking into account continuity of states.
Similarly, optimisation variables needed to be initialized to speed up the solver. At time $k = 0$, there are no enough information to give to $\boldsymbol{u}_0$, so this initialization is done as follows

$$\boldsymbol{u}_0 = x(0|k) \tag{3.44}$$

On the other hand, when $k \geq 1$, the vector of optimisation variables is initialized with

$$\boldsymbol{u}_0 = \begin{bmatrix} u(1|k-1) \\ u(2|k-1) \\ \vdots \\ u(N-1|k-1) \\ u(N|k-1) \\ u(N|k-1) \end{bmatrix} \tag{3.45}$$

This practically means that old optimisation variables, except for $u(0|k)$, computed at the previous iteration, are used to initialize this vector.
Third, constraints on acceleration (3.32e) and velocity (3.32f) related to the drones are considered. Recalling the state matrices of the model in continuous time of a single drone (2.11), we can write the acceleration of the vehicle as:

$$\boldsymbol{A}(t) = K_{vel}(K_{pos}(\boldsymbol{P}_{ref}(t) - \boldsymbol{P}(t)) - \boldsymbol{V}(t)), \tag{3.46}$$

where $K_{vel}, K_{pos} \in \mathbb{R}^{2\times 2}$ are matrices pertaining to the gains k_{vel}, k_{pos} (see (2.11)) recollected in matrix form. From (3.46) we can write:

$$-\overline{\boldsymbol{A}} \leq K_{vel}(K_{pos}(\boldsymbol{u}(j|k) - \boldsymbol{x}_{i_{1:2}}(j|k)) - \boldsymbol{x}_{i_{3:4}}(j|k)) \leq \overline{\boldsymbol{A}}, \ \ \forall j \in \mathbb{N}_0^{N-1}, \ \ \forall i \in \mathbb{N}_1^{N_d}, \tag{3.47}$$

where $\overline{\boldsymbol{A}}$ is vector of the maximum acceleration of the drones, assumed here for simplicity to be identical for all vehicles. For what concern velocity constraints, since the velocity of the i-th drone is a state, it is possible to directly write the constraints as:

$$-\overline{\boldsymbol{V}} \leq \boldsymbol{x}_{i_{(3:4)}}(j|k) \leq \overline{\boldsymbol{V}}, \ \ \forall j \in \mathbb{N}_0^{N}, \ \ \forall i \in \mathbb{N}_1^{N_d}, \tag{3.48}$$

where $\overline{\boldsymbol{V}}$, which is the maximum velocity of the drones, is assumed to be the same for all drones.
To avoid contact between the drones during path following is necessary to include non-linear constraints involving distance between drones (3.32h). More in the details, the distance between drone i and drone $i+1$, $i = 1, \ldots, N_d - 1$ has to be higher than a security distance called $\underline{d}$, chosen by user.
The last set of constraints included in FHOCP are the terminal constraints, in fact all the simulated trajectories have to end with a velocity equal to zero (3.32j). To summarize, the FHOCP $\mathcal{P}(x(k), S(k), \boldsymbol{x}_g)$ presents linear constraints related to the system dynamics (3.32b), position (3.32g), (3.32i), velocity (3.32f), acceleration (3.32e), while the non-linear

constraints related to the distances between the drones are expressed in (3.32h). Finally, linear constraints are imposed on the terminal state (3.32j) and on the initial condition of states (3.32c)) and inputs (3.32d).
FHOCP is solved every sample time T_s, it is a non-convex problem, due to the non-convex constraints (3.32h) related to the distance between drones. Its solution is denoted as $U^*(\boldsymbol{x}(t), S(k), \boldsymbol{x}_g(k))$. At any time k the latest sampling instant is denoted as $\underline{k}(k) < k$ such that the FHOCP $\mathcal{P}(\boldsymbol{x}(\underline{k}(k)), S(\underline{k}(k)), \boldsymbol{x}_g(\underline{k}(k)))$ was feasible. FHOCP is embedded in the following receding horizon strategy:
Algorithm: path following

1. At time k collect the LiDAR measurements and fuse them according to drones coupling;
2. Find optimal ellipse contained in LiDAR merged readings;
3. Compute the set S(k) containing the safe sets for all coupled drones;
4. **if** FHOCP $\mathcal{P}(\boldsymbol{x}(k), S(k), \boldsymbol{x}_g(k))$ is feasible **then**
   apply to the system the first control input in the optimal sequence $U^*(\boldsymbol{x}(k), S(k), \boldsymbol{x}_g(k)$. Set $\underline{k}(k+1) = k$ and store the feasible set of constraints used to solve the problem $S(k)$ as $S(\underline{k}(k+1))$.
   else
   solve $\mathcal{P}(\boldsymbol{x}(k), S(\underline{k}(k)), \boldsymbol{x}_g(k))$ and apply to the system the first control input in the optimal sequence $U^*(\boldsymbol{x}(k), S(\underline{k}(k)), \boldsymbol{x}_g(k))$. Set $\underline{k}(k+1) = \underline{k}(k)$.
   end if
5. set $k = k + 1$ and go to 1).

Since the environment is assumed to be time invariant, the safe sets $D_j(k)$ depend only on the system state $\boldsymbol{x}(k)$. MPC approach results in a dynamic controller with internal states $\underline{k}(k)$ and $\boldsymbol{x}(k)$ and $\boldsymbol{x}_g$ as inputs:

$$\underline{k}(k+1) = \boldsymbol{\eta}(\boldsymbol{x}(k), \underline{k}(k)) \tag{3.49}$$

$$\boldsymbol{u}(k) = \boldsymbol{\kappa}(\boldsymbol{x}(k), \underline{k}(k), \boldsymbol{x}_g(k)) \tag{3.50}$$

where functions $\boldsymbol{\eta} : \mathbb{R}^{4N_d} \times \mathbb{Z} \to \mathbb{Z}$ and $\boldsymbol{\kappa} : \mathbb{R}^{4N_d} \times \mathbb{Z} \times \mathbb{R}^{2N_d} \to \mathbb{R}^{2N_d}$ are implicitly defined by Algorithm. The closed loop system is

$$\underline{k}(k+1) = \boldsymbol{\eta}(\boldsymbol{x}(k), \underline{k}(k)) \tag{3.51}$$

$$\boldsymbol{x}(k+1) = A\boldsymbol{x}(k) + B\boldsymbol{\kappa}(\boldsymbol{x}(k), \underline{k}(k), \boldsymbol{x}_g(k)) \tag{3.52}$$

In Algorithm, the role of variable $\underline{k}$(k) is to guarantee that at each time step a feasible FHOCP can be formulated, despite the time-varying nature of the safe convex set $S(k)$. This guarantee does not hold if in the environment are present time-varying obstacles.

Lemma. *Assume that the FHOCP at time $k = k_0$ is feasible and $\boldsymbol{x}(k_0)\mathbb{P}_{xy} \in S_{free}$, $(\boldsymbol{P}_{i+1}(k_0) - \boldsymbol{P}_i(k_0) \in \boldsymbol{S}_{free}, i = 0, \dots, N_d - 1$, this basically means that the drones and the tethers are initially in the obstacle-free region. Then, the trajectory of the closed loop system is such that $\boldsymbol{x}(k)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}$, $\boldsymbol{P}_{i+1}(k) - \boldsymbol{P}_i(k) \in \boldsymbol{S}_{free}$, $\forall k > k_0$.*

Proof. At $k = k_0$, problem $\mathcal{P}(\boldsymbol{x}(K_0), S(k_0), \boldsymbol{x}_g(k_0))$ is solved and $\underline{k}(k_0 + 1)$ is sat equal to k_0. For any $k \geq k_0$, the optimal safe input sequence computed by MPC algorithm, is denoted with $U*(k) = [\boldsymbol{u}^*(1|k)^T, ..., u^*(N|t)^T]$ be it by solving $\mathcal{P}(\boldsymbol{x}(k), S(k), \boldsymbol{x}_{goal}(k))$ or $\mathcal{P}(\boldsymbol{x}(k), S(\underline{k}(k)), \boldsymbol{x}_g((k)))$, leading to the optimal state trajectory $X^*(k) = [x^*(1|k)^T, \dots, x^*(N|k)^T]$ and with $\boldsymbol{x}^*(N|k)$, which is the corresponding safe terminal set. Then, for each $k \geq k_0 + 1$, there are only two possibilities:

1) If $\mathcal{P}(\boldsymbol{x}(k_0 + 1), S(k_0 + 1), \boldsymbol{x}_g(k))$ is feasible, $x(k_0 + 1)\mathbb{P}_{xy} \in \boldsymbol{S}(k)$

2) Conversely, if $\mathcal{P}(\boldsymbol{x}(k_0 + 1), S(k_0 + 1), \boldsymbol{x}_g(k))$ is not feasible, problem $\mathcal{P}(\boldsymbol{x}(k), S(\underline{k}(k)), \boldsymbol{x}_g(k))$ is solved, where a feasible sequence can be built considering the tail of $U^*(\underline{k}(k_0+1)) = U^*(k_0)$, i.e.$[u^*(1|\underline{k}(k_0+1))^T, \dots, u^*(N|\underline{k}(k_0+1))^T, 0^{1\times 2N_d}$. In fact, terminal state of $X^*\underline{k}(k_0+1) = X^*(k_0)$ is a steady state for the system. AS a consequence $x(k_0 + 1)\mathbb{P}_{xy} \in S(\underline{k}(k_0 + 1))$.

Therefore, in both cases **1)** and **2)**, $\boldsymbol{x}(k_0 + 1)\mathbb{P}_{xy}$ belongs to a set $S(j)$ with $j \leq k_0 + 1$. Now, by construction, the corresponding polytope set $D_j(k) \in S(k)$ is an under-approximation of the obstacle-free region containing two drones and also the segment $\boldsymbol{P}_{i+1} - \boldsymbol{P}_i$. To conclude, if $x(k_0)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}, (\boldsymbol{P}_{i+1}(k_0) - P_i(k_0) \in S_{free} \implies x(k_0+1)\mathbb{P}_{xy} \in \boldsymbol{S}_{free}, \boldsymbol{P}_{i+1}(k_0 + 1) - \boldsymbol{P}_i(k_0 + 1) \in S_{free}$. □

3.5. Replanning Strategy

A second offline optimisation is performed when the targets, which correspond to the optimal configuration $\boldsymbol{C}^*$, are reached by the drones. The policy of this OP is similar to the one seen previously, in section (3.2) but with substantial differences. A new target $\boldsymbol{P}_{target}$ is assigned to the leader drone, then, the offline planner starts to find a solution by moving just the leader drone while blocking the other ones. If a solution is not obtained, the number of moved drones is increased by one, while the number of blocked drones is decreased by the same quantity, unless all the drones are considered free and we have

exactly the same situation of the problem in subsection (3.2.1). The vector of optimization variables is the vector of (x, y) coordinates of all the drones $\boldsymbol{x} = \begin{bmatrix} x_1 & y_1 & \dots & x_{N_d} & y_{N_d} \end{bmatrix}$. The mathematical formulation of this OP is similar to (3.3):

$$\min_{\boldsymbol{x} \in \mathbb{R}^{2N_d}} f(\boldsymbol{x}) \tag{3.53a}$$

$$s.t$$

$$A\boldsymbol{x} = \boldsymbol{b} \tag{3.53b}$$

$$g(\boldsymbol{x}) = 0 \tag{3.53c}$$

$$h(\boldsymbol{x}) \geq 0, \tag{3.53d}$$

where (3.53a), (3.53c) and (3.53d) have the same expression of the first path planner problem. Instead, (3.53b) represents linear equality constraints and its form depends on the number of free drones N_F. This process begins considering the leader drone as free ($N_F = 1$), and drones i, $i = 2, \dots, N_d - 1$ blocked. In this case the matrices A and $\boldsymbol{b}$ are such that $A\boldsymbol{x} = \boldsymbol{b}$ correspond to

$$\begin{aligned} x_{N_d} &= P^x_{N_d} \\ y_{N_d} &= P^y_{N_d} \\ &\vdots \\ x_{N_d - i} &= P^x_{N_d - i} \\ y_{N_d - i} &= P^y_{N_d - i} \\ &\vdots \\ x_2 &= P^x_2 \\ y_2 &= P^y_2. \end{aligned} \tag{3.54}$$

where $i = 1, \dots, N_d - 3$. If a solution is not found, then, the number of free drones is increased by one ($N_F = N_F + 1$). This process continues until we consider as blocked just the drone which is attached to the ground station. In fact, when the drones i, $i = 1, \dots, N_d - 1$ are free, matrices A and $\boldsymbol{b}$ are such that $A\boldsymbol{x} = \boldsymbol{b}$ provide:

$$\begin{aligned} x_{N_d} &= P^x_{N_d} \\ y_{N_d} &= P^y_{N_d} \end{aligned} \tag{3.55}$$

If all the drones i, $\forall i = 1, \dots, N_d$ are considered free ($N_F = N_d$), it means that $A = [\ ]$ and $\boldsymbol{b} = [\ ]$, which is exactly the same situation of section (3.2). With this second offline

planner, a new optimal configuration is obtained:

$$\boldsymbol{C}^* = \begin{bmatrix} \boldsymbol{C}_1^{*T} & \dots & \boldsymbol{C}_{N_d}^{*T} \end{bmatrix}^T, \tag{3.56}$$

where $\boldsymbol{C}_i^* = \begin{bmatrix} C_i^x \\ C_i^y \end{bmatrix}$ represents (x, y) target coordinate of i-th drone.
Remark: If a drone i is considered blocked, the respective row of the vector of targets become $\boldsymbol{C}_i^* = \boldsymbol{P}_i$.
In our situation, since $N_d = 3$, only three cases are studied. If the leader drone is blocked, matrices $A, \boldsymbol{b}$ are constructed such that the operation $A\boldsymbol{x} = \boldsymbol{b}$ returns:

$$\begin{aligned} x_3 &= P_3^x \\ y_3 &= P_3^y \\ x_2 &= P_2^x \\ y_2 &= P_2^y \end{aligned} \tag{3.57}$$

If the leader drone and drone$_2$ are considered free the blocked coordinates are:

$$\begin{aligned} x_3 &= P_3^x \\ y_3 &= P_3^y \end{aligned} \tag{3.58}$$

Finally, when all drones are considered free, it means that $A = [\ ]$ and $\boldsymbol{b} = [\ ]$.

3.5.1. Triangles method for replanning

When the new targets of the drones are found with second offline optimisation, a new path is obtained, following the same steps of subsection (3.4.2). Now, a routine algorithm decides the correct strategy which has to be followed by the drones. It is based on the analysis of the triangles which is possible to check starting from drones position $\boldsymbol{P}_i$, $i = 1, \dots, N_d$, ground station position $\boldsymbol{P}_0$ and targets position $\boldsymbol{C}_i^*$. The simplest situation corresponds to get a solution from (3.53) just moving the leader drone, while drones i, $i = 2, \dots, N_d$, are blocked in their position. In this case, the only relevant triangle which is obtained has these vertices: $\boldsymbol{P}_1$, $\boldsymbol{C}_1^*$ and $\boldsymbol{P}_2$. An example is provided in Figure (3.12).

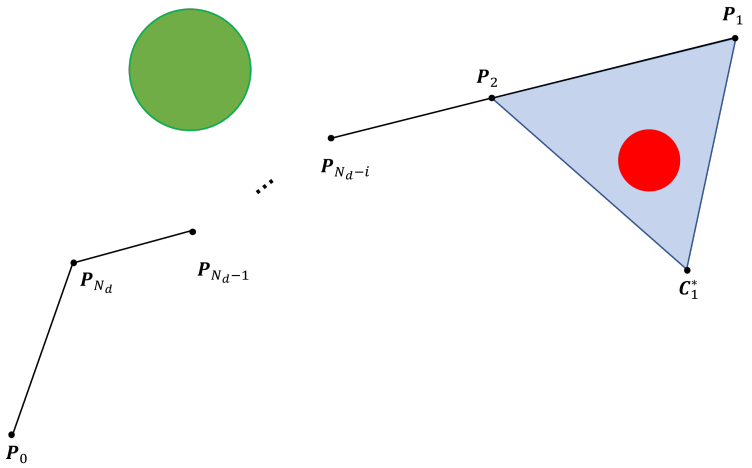

Figure 3.12: An example of triangle obtained when leader drone is free. In green and red the obstacles. $\boldsymbol{P}_i$ are the points which represent the coordinates of drone i.

Now, a LP similar to the one seen in subsection (3.4.3) has to be solved, where every L_n reading has to pass a feasibility check. The LP has to be solved for every LiDAR reading $L_{n,i}$, $i = 1, \dots, N_{L_n}$, and it can be written in this form:

$$\min_{x,y} const \tag{3.59a}$$

$$s.t.$$

$$A_{triangle} \begin{bmatrix} x \\ y \end{bmatrix} \leq \boldsymbol{b}_{triangle} \tag{3.59b}$$

$$x_0 = \begin{bmatrix} L_{n,i}^x \\ L_{n,i}^y \end{bmatrix} \tag{3.59c}$$

Here, the vertices of the triangle which produce the matrices $A_{triangle}, \boldsymbol{b}_{triangle}$ are:

$$V_{triangle} = \begin{bmatrix} P_1^x & P_1^y \\ P_2^x & P_2^y \\ C_1^{*,x} & C_1^{*,y} \end{bmatrix} \tag{3.60}$$

Finally, two situations are possible, if the triangle is occupied, it means that an obstacle is present, consequently the leader drone can not go directly towards its target $\boldsymbol{C}_1^*$ and has to backtrack towards drone$_2$ (strategy chosen is called "backtrack$_1$"), otherwise, its goal is imposed to be its target, $\boldsymbol{x}_{g_1} = \boldsymbol{C}_1^*$ and strategy "direct follow$_1$" is picked.
At this point, the same steps can be done extending the reasoning to all i, $i = 1, \dots, N_F$ free drones, while drones i, $i = N_F + 1, \dots, N_d$ are considered blocked, recalling that N_F is computed during the problem (3.53). The triangles which need to be tested are a function

of N_F. With $N_F = i$ we have N_F free drones and j drones blocked, $j = N_F + 1, \ldots, N_d$. From this information the triangles can be computed following these steps:

1. Group in G_i sets the coupled targets and drones i, $i+1$ with $i = 1, \ldots, N_F - 1$, $G_i = \left\{\boldsymbol{P}_i, \boldsymbol{C}_i^*, \boldsymbol{P}_{i+1}, \boldsymbol{C}_{i+1}^*\right\}, \ldots, G_{N_F-1} = \left\{\boldsymbol{P}_{N_F-1}, \boldsymbol{C}_{N_F-1}^*, \boldsymbol{P}_{N_F}, \boldsymbol{C}_{N_F}^*\right\}$;
2. From each group G_i extract the combination of 3 elements which are possible to obtain with the all 4 elements of the set. The number of combination for each set G_i is computed recalling to the probability theory as:

$$c_{4,3} = \frac{4!}{3!} = 4 \tag{3.61}$$

3. Name each combination of group G_i as T_j, $j = 1, \ldots, 4$ which correspond to the triangle j;
4. Add the last triangle, which is always described by these vertices $\boldsymbol{P}_{N_F+1}, \boldsymbol{P}_{N_F}, \boldsymbol{C}_{N_F}^*$.

Remark: If all the drones are considered free, $N_F = N_d$, the last triangle to be added is $\boldsymbol{P}_0, \boldsymbol{P}_{N_F}, \boldsymbol{C}_{N_F}^*$.
Once obtained all the triangles, a vector which contains boolean variables is introduced: $\boldsymbol{tr}$. The number of elements of this vector is a function of G_i. In fact, each set G_i is composed of four triangles plus the terminal one, so, for each set we have five elements. The elements of $\boldsymbol{tr}$ are the boolean variables tr_i, where tr_i stands for the i-th triangle T_i. If tr_i is equal to 0, it means that triangle T_i is free, otherwise, if it is equal to 1, it means that it is occupied. Then, studying the vector $\boldsymbol{tr}$, is possible to choose the correct strategy analyzing the free triangles. More in the details, the number of triangles to be analyzed change with N_F, in fact it is equal to $2N_F - 1$. The aforementioned vector $\boldsymbol{tr}$ is filled solving at each sampling time a LP similar to the one presented in (3.59), which has the aim to check when an obstacle is present or not in triangles.
In our case we can test the complete method with $N_F = 2$ and $N_F = N_d = 3$. Starting from $N_F = 2$, it is possible to construct only the set $G_1 = \{\boldsymbol{P}_1, \boldsymbol{C}_1^*, \boldsymbol{P}_2, \boldsymbol{C}_2^*\}$, as we can see in Figure (3.13).

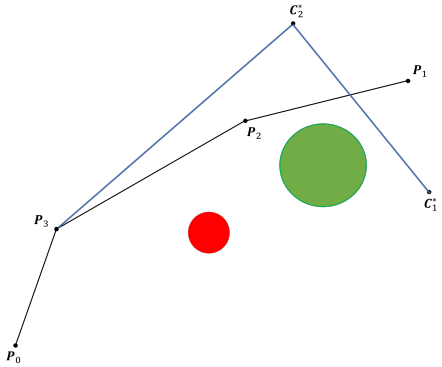

Figure 3.13: A representation of triangle method. In red and green the obstacles. $\boldsymbol{C}_i^*$ represent the target of drone i. Blue segments represent the final position of the tethers, while black ones represent the actual position.

Here, there are five triangles, which can be resumed in Table (3.1):

T_1	$\boldsymbol{P}_1$	$\boldsymbol{C}_1^*$	$\boldsymbol{C}_2^*$
T_2	$\boldsymbol{P}_1$	$\boldsymbol{P}_2$	$\boldsymbol{C}_2^*$
T_3	$\boldsymbol{P}_2$	$\boldsymbol{C}_1^*$	$\boldsymbol{C}_2^*$
T_4	$\boldsymbol{C}_1^*$	$\boldsymbol{P}_1$	$\boldsymbol{P}_2$
T_5	$\boldsymbol{P}_2$	$\boldsymbol{C}_2^*$	$\boldsymbol{P}_3$

Table 3.1: An example of triangles obtained when drone$_1$ and drone$_2$ are free

As discussed before, the vector $\boldsymbol{tr} = \begin{bmatrix} tr_1 & tr_2 & tr_3 & tr_4 & tr_5 \end{bmatrix}$ is filled with the output of the LP. Now, the options are discriminated through the vector $\boldsymbol{tr}$:

- if all the elements of the vector $\boldsymbol{tr}$ are 0, it means that all the triangles are free, consequently the new targets $\boldsymbol{C}_i^*$, $i = 1, 2$, are just assigned as goal. The aforementioned strategy is called "direct follow$_2$" and can be seen in the Figure (3.14).

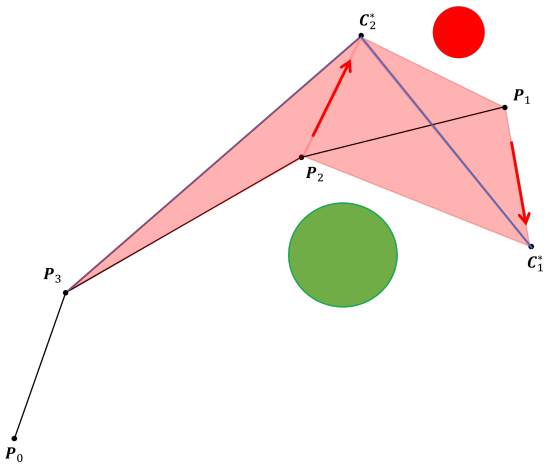

Figure 3.14: A representation of "direct follow$_2$" strategy. The five triangles all together must be free, they are represented as the red space. The green and the red circles are the obstacles.

- if tr_3, tr_4, and tr_5 are all equal to 0, the strategy assigned is called "d_1t_1-d_2t_2", it means that the new target $\boldsymbol{C}_1^*$ is assigned to drone$_1$, then, once this is reached, target $\boldsymbol{C}_2^*$ is assigned to drone$_2$. It is possible to visualize it in a proper way in the Figure (3.15).

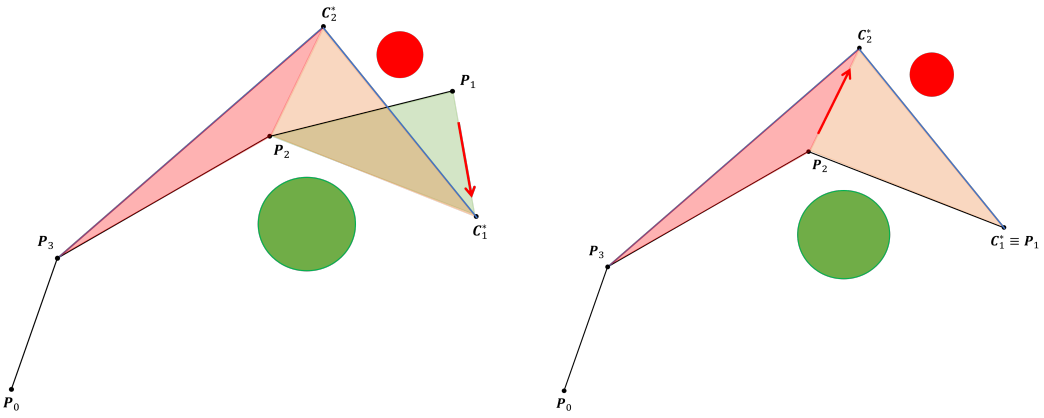

Figure 3.15: A representation of strategy "d_1t_1-d_2t_2", where T_4 corresponds to the green triangle, T_5 corresponds to the red triangle and T_3 is shown as the orange triangle.

- if T_1, T_2, and T_5 are all equal to 0, the strategy assigned is called "d_2t_2-d_1t_1". This means that first, new target $\boldsymbol{C}_2^*$ is assigned to drone$_2$ as goal, after that the target is reached, $\boldsymbol{C}_1^*$ is assigned to drone$_1$ as goal. This strategy is shown in Figure (3.16).

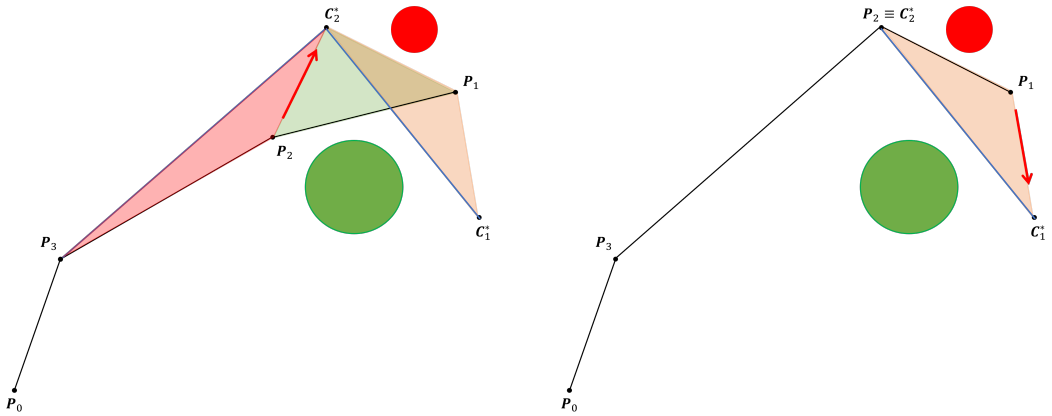

Figure 3.16: A representation of strategy "d_2t_2-d_1t_1", where T_2 is the green triangle, T_5 is represented as the red triangle and T_1 corresponds to the orange one.

- if none of the previous conditions is met, the chosen strategy is "backtrack$_2$".

The aforementioned strategies are summarized and explained in the details in the next subsection (3.5.2).
In the case of $N_F = N_d = 3$, the number of triangles considered increase, as we discuss before. In fact, we have two sets, $G_1 = \{\boldsymbol{P}_1, \boldsymbol{C}_1^*, \boldsymbol{P}_2, \boldsymbol{C}_2^*\}$, and $G_2 = \{\boldsymbol{P}_2, \boldsymbol{C}_2^*, \boldsymbol{P}_3, \boldsymbol{C}_3^*\}$, as it is possible to see in Figure (3.17).

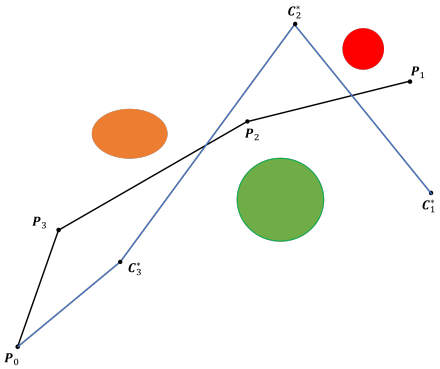

Figure 3.17: A representation of triangle method. In red and green the obstacles. $\boldsymbol{C}_i^*$ represent the target of drone i. Blue segments represent the final position of the tethers, while black ones represent the actual position.

Nine triangles have to be introduced, eight from the two set plus the terminal one. They are schematically resumed in table (3.2):

T_1	$\boldsymbol{P}_1$	$\boldsymbol{P}_2$	$\boldsymbol{C}_1^*$
T_2	$\boldsymbol{P}_2$	$\boldsymbol{C}_1^*$	$\boldsymbol{C}_2^*$
T_3	$\boldsymbol{P}_2$	$\boldsymbol{P}_3$	$\boldsymbol{C}_2^*$
T_4	$\boldsymbol{P}_3$	$\boldsymbol{C}_3^*$	$\boldsymbol{C}_2^*$
T_5	$\boldsymbol{P}_2$	$\boldsymbol{C}_2^*$	$\boldsymbol{C}_3^*$
T_6	$\boldsymbol{P}_3$	$\boldsymbol{P}_2$	$\boldsymbol{C}_3^*$
T_7	$\boldsymbol{P}_2$	$\boldsymbol{P}_1$	$\boldsymbol{C}_2^*$
T_8	$\boldsymbol{P}_1$	$\boldsymbol{C}_1^*$	$\boldsymbol{C}_2^*$
T_9	$\boldsymbol{P}_3$	$\boldsymbol{P}_0$	$\boldsymbol{C}_3^*$

Table 3.2: An example of triangles when drone$_1$, drone$_2$ and drone$_3$ are free

Similarly to the previous case case, the same LP, which provides a feasibility check, has to be solved. It finds when an obstacle is present or not in triangles, meaning whenever a triangle is free or it is occupied. The same vector of before, $\boldsymbol{tr}$, which contains logic variables, is introduced: $\boldsymbol{tr} = \begin{bmatrix} tr_1 & tr_2 & tr_3 & tr_4 & tr_5 & tr_6 & tr_7 & tr_8 & tr_9 \end{bmatrix}$, where tr_i stands for the i-th triangle. As before, tr_i equal to 0 means that triangle T_i is free, otherwise it means that it is occupied. The options are again explored using the vector $\boldsymbol{tr}$:

- if the vector $\boldsymbol{tr}$ is composed by all 0 elements, then the triangles are free. As a consequence, the new targets $\boldsymbol{C}_i^*$, $i = 1, \ldots, 3$ are assigned as goal directly. The strategy "direct follow$_3$" is chosen. An example of this strategy is described in Figure (3.18).

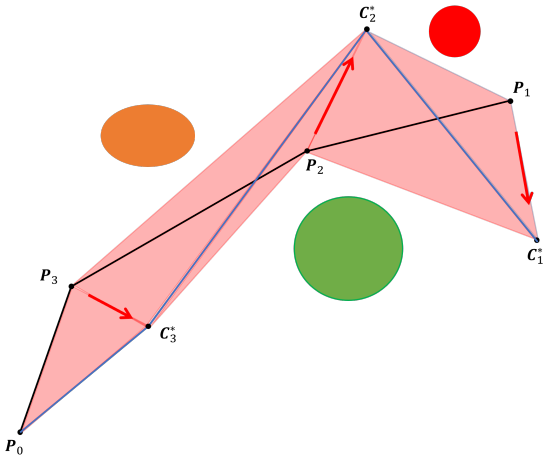

Figure 3.18: A representation of "direct follow$_3$" strategy. The nine triangles has to be considered all free. The colored circles are the obstacles.

- when tr_1, tr_2, tr_3, tr_4 and tr_9 are 0, the strategy which is chosen is called "d$_1$t$_1$-d$_2$t$_2$-

d_3t_3". This strategy assigns $target_1$ as $goal_1$, then, after the first target is reached, $target_2$ is assigned as $goal_2$. Finally, $target_3$ is assigned as $goal_3$. The Figure (3.19) represents this strategy

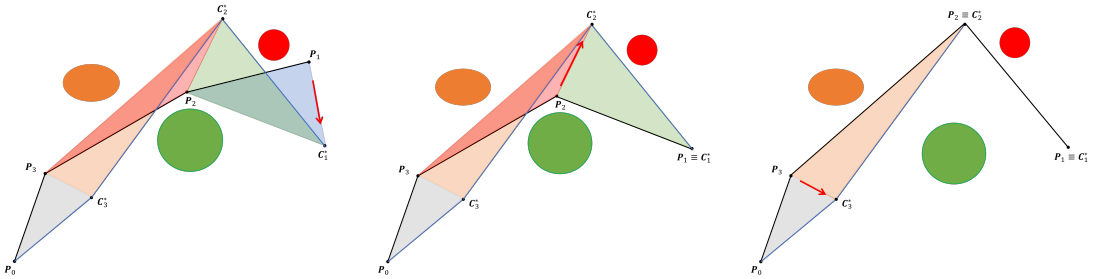

Figure 3.19: A representation of "d_1t_1-d_2t_2-d_3t_3" strategy. Here, T_1 is the blue triangle, T_2 is represented as the green triangle, T_3 corresponds to the red one, T_4 is the one with orange color while, T_9 has grey color.

- when tr_1, tr_2, tr_5, tr_6 and tr_9 are 0, the strategy which is picked is called "d_1t_1-d_3t_3-d_2t_2". This strategy assigns target $\boldsymbol{C}_1^*$ as $goal_1$, then, after reaching it, $\boldsymbol{C}_2^*$ is assigned as $goal_2$. Finally, whenever the previous target is reached, $\boldsymbol{C}_3^*$ is assigned as $goal_3$. This strategy is shown in Figure (3.20).

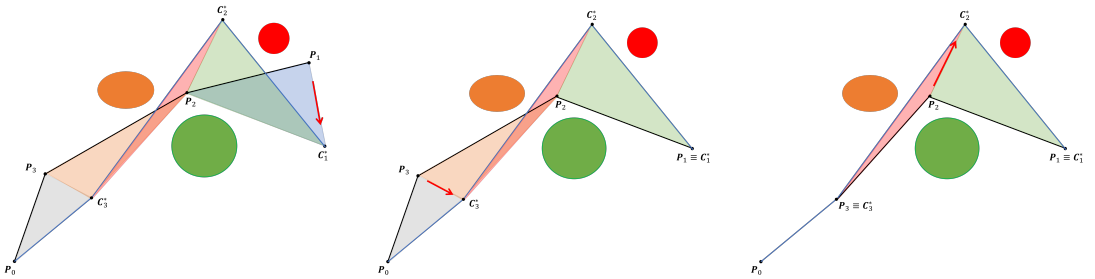

Figure 3.20: A representation of "d_1t_1-d_3t_3-d_2t_2" strategy. Here, T_1 is the blue triangle, T_2 is represented as the green triangle, T_5 corresponds to the red one, T_6 is the one with orange color while, T_9 has grey color.

- when tr_3, tr_6, tr_7, tr_8 and tr_9 are 0, the chosen strategy is called "d_2t_2-d_1t_1-d_3t_3". This strategy assigns initially target $\boldsymbol{C}_2^*$ as $goal_2$, then, after it is reached, target $\boldsymbol{C}_1^*$ is assigned as $goal_1$. Finally, target $\boldsymbol{C}_3^*$ is assigned as $goal_3$. This strategy is well shown in Figure (3.21).

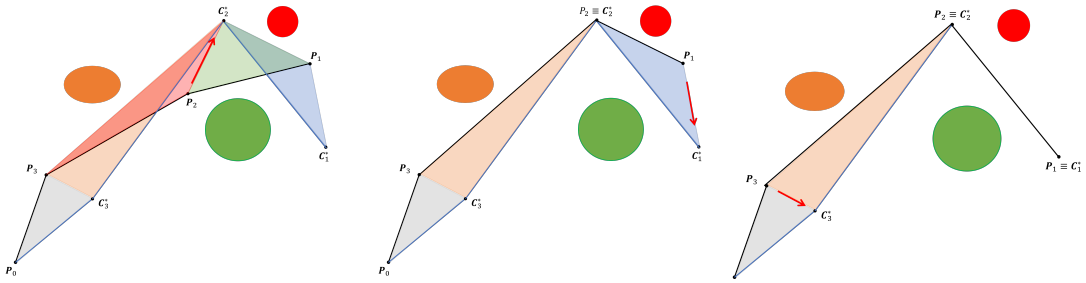

Figure 3.21: An example of strategy "d_2t_2-d_1t_1-d_3t_3". In this strategy, T_8 is the blue triangle, T_7 is the one with green color, T_3 corresponds to the red one, T_6 is represented as the orange triangle, while T_9 has grey color.

- when none of these conditions is satisfied, then, the adopted strategy is "backtrack$_3$"

Even in this case, the strategies seen before are better explained in the next subsection 3.5.2. As it is said, this are the results obtained with $N_d = 3$. For a generic system, composed by N_d and $N_d \geq 4$, the same steps need to be followed. As we can see, every strategy proposed here, differs from another one because of the space occupied by the triangles (see Figures (3.15) and (3.16) for $N_d = 2$, and Figures (3.19),(3.20),(3.21) for $N_d = 3$). In fact, any other combination of triangles does not produce any new strategy, because they occupy the same space. An overview of all treated strategies is presented in the table (3.3).

Overview		
N_F	STRATEGY	TYPE
1	direct follow$_1$	follow
	backtrack$_1$	backtrack
2	direct follow$_2$	follow
	d_1t_1-d_2t_2	follow
	d_2t_2-d_1t_1	follow
	backtrack$_2$	backtrack
3	direct follow$_3$	follow
	d_1t_1-d_2t_2-d_3t_3	follow
	d_1t_1-d_3t_3-d_2t_2	follow
	d_2t_2-d_1t_1-d_3t_3	follow
	backtrack$_3$	backtrack

Table 3.3: Overview of strategies

3.5.2. Backtrack/Follow strategy

In this subsection it is explained in the detail how the navigation algorithm behaves with "backtrack" or "follow" strategies type (see Table (3.3)) according to subsection (3.4.2). The first group of follow type strategies are "direct follow$_i$" $i = 1, \dots, N_d$. As explained in subsection (3.5.1), this essentially means that if one of these strategies is chosen, the vector of goal changes in $\boldsymbol{x}_{g_i} = \boldsymbol{C}_i^*$, $\forall i = 1, \dots, N_F$, while drones j, $j = 1, \dots, N_d - N_F$, remain still. The position constraints (3.32g), needed by FHOCP are constructed as:

$$\boldsymbol{x}_{i_{1,2}}(j|k) \in D_i \;\; \forall j \in \mathbb{N}_0^N, \;\; \forall i \in \mathbb{N}_1^{N_d} \tag{3.62}$$

An example of "direct follow$_2$" strategy is provided in Figure (3.22).

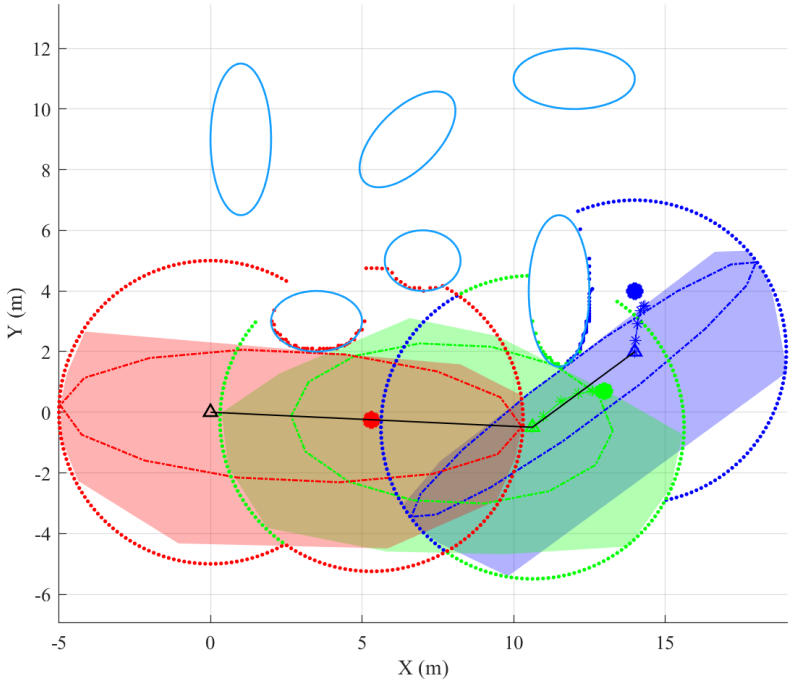

Figure 3.22: An example of "direct follow$_2$" strategy. The colored triangles represent the actual position of the drones after the first path following. The black one is the ground station, while the big colored dots represent their target. In this strategy the vector $\boldsymbol{x}_g$ is constructed in this way: $\boldsymbol{x}_{g_1} = \boldsymbol{C}_1^*, \boldsymbol{x}_{g_2} = \boldsymbol{C}_2^*, \boldsymbol{x}_{g_3} = \boldsymbol{P}_3$.

The second group is described by the general strategy "d$_i$t$_i$-d$_j$t$_j$", $i, j = 1, \dots, N_d - 1$, $i \neq j$. In this type of strategy, first, $\boldsymbol{x}_{g_i} = \boldsymbol{C}_i^*$ becomes the goal of drone$_i$, while drone$_j$ remains still, $\boldsymbol{x}_{g_j} = \boldsymbol{P}_j$. After that the previous target is successfully reached, the target

$\boldsymbol{C}_j^*$ is assigned as goal of the drone$_j$: $\boldsymbol{x}_{g_j} = \boldsymbol{C}_j^*$. Position constraints, in this case are:

$$\boldsymbol{x}_{i_{1,2}}(j|k) \in D_i \;\; \forall j \in \mathbb{N}_0^N, \;\; \forall i \in \mathbb{N}_1^{N_d} \tag{3.63}$$

An example of "d_2t_2-d_1t_1" strategy is reported in Figure (3.23).

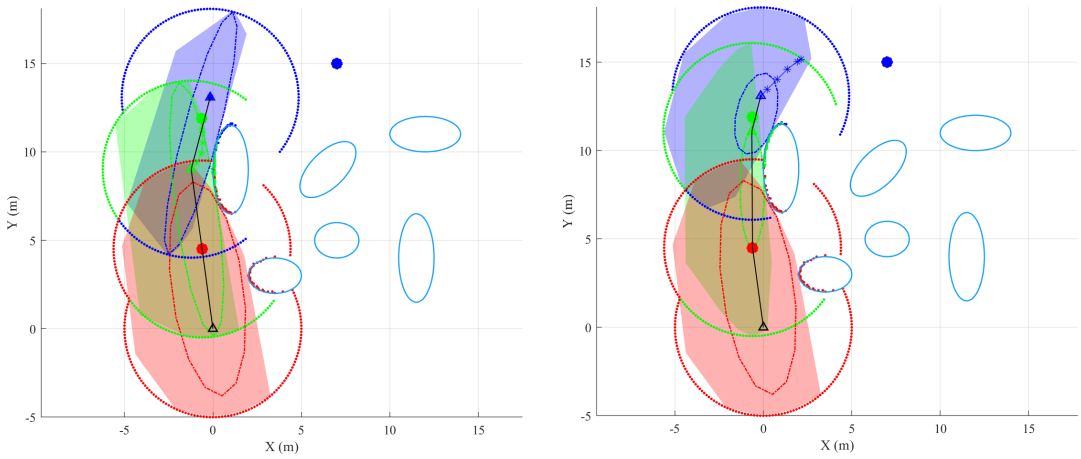

Figure 3.23: An example of "d_2t_2-d_1t_1" strategy. The colored triangles represent the actual position of the drones after the first path following. The black one is the ground station, while the big colored dots represent their target. In blue the obstacles. On the left, the first step of this strategy, where the vector $\boldsymbol{x}_g$ is constructed in this way: $\boldsymbol{x}_{g_1} = \boldsymbol{P}_1, \boldsymbol{x}_{g_2} = \boldsymbol{C}_2^*, \boldsymbol{x}_{g_3} = \boldsymbol{P}_3$. On the right, the vector $\boldsymbol{x}_g$ changes as soon as the drone$_2$ is very close to its target. In fact, it becomes $\boldsymbol{x}_{g_1} = \boldsymbol{C}_1^*, \boldsymbol{x}_{g_2} = \boldsymbol{C}_2^*, \boldsymbol{x}_{g_3} = \boldsymbol{P}_3$.

It is important to remark that this is the correct strategy to apply to the system in this case. If the strategy algorithm had chosen an alternative strategy, this would have provoked the collision between the tether$_2$ and the obstacle on the upper-left in the Figure (3.23).

The third group is described by the general strategy "d_it_i-d_jt_j-d_lt_l", with $i, j, l = 1, \ldots, N_d$, $i \neq j$, $j \neq l$, $i \neq l$). In the first part, the target of drone$_i$ is assigned as goal, $\boldsymbol{x}_{g_i} = \boldsymbol{C}_i^*$, while drone$_j$ and drone$_l$ remain still in their positions: $\boldsymbol{x}_{g_j} = \boldsymbol{P}_j$ and $\boldsymbol{x}_{g_l} = \boldsymbol{P}_l$. After that the previous target is successfully reached, the target $\boldsymbol{C}_j^*$ is assigned as goal of the drone$_j$, $\boldsymbol{x}_{g_j} = \boldsymbol{C}_j^*$, while drone$_l$ continue to remain still: $\boldsymbol{x}_{g_l} = \boldsymbol{P}_l$. Finally, When drone j reaches its target, it is finally possible to assign $\boldsymbol{C}_l^*$ as goal of drone$_l$, $\boldsymbol{x}_{g_l} = \boldsymbol{C}_l^*$. Even in this last case, the position constraints of FHOCP are constructed as:

$$\boldsymbol{x}_{i_{1,2}}(j|k) \in D_i \;\; \forall j \in \mathbb{N}_0^N, \;\; \forall i \in \mathbb{N}_1^{N_d} \tag{3.64}$$

On the other hand, if one of the backtrack$_i$ strategy is chosen, things are a little bit more

complicated. In all the backtrack strategies a path following approach, similar to the one presented in subsection (3.4.2) is used. First, the path is created, recalling that in this case the initial point of the path is the position of leader drone $\boldsymbol{P}_1$, while the final point is the position of the ground station $\boldsymbol{P}_0$. Recalling the general matrix Γ in (3.35), where each row represent the coordinates of a point of the path, we can write the matrix Γ as:

$$\Gamma = \begin{bmatrix} P_1^x & P_1^y \\ P_1^x - \tau\cos\Psi_{N_d-1} & P_1^y - \tau\sin\Psi_{N_d-1} \\ P_1^x - 2\tau\cos\Psi_{N_d-1} & P_1^y - 2\tau\sin\Psi_{N_d-1} \\ \vdots & \vdots \\ P_2^x - \mu_1\tau\cos\Psi_{N_d-2} & P_2^y - \mu_1\tau\sin\Psi_{N_d-2} \\ \vdots & \vdots \\ P_{N_d}^x - \mu_{N_d}\tau\cos\Psi_0 & P_{N_d}^y - \mu_{N_d}\tau\sin\Psi_0 \\ \vdots & \vdots \\ P_0^x & P_0^y \end{bmatrix} \tag{3.65}$$

where recalling to subsection (3.4.2), Ψ_i, $i = 0, \ldots, N_d - 1$ corresponds to angle between tether i and x-axis, τ is a user-defined variable and μ_i is a coefficient which guarantees that all the points on the path are spaced by τ meters. An example of path in backtrack strategy is represented in Figure (3.24)

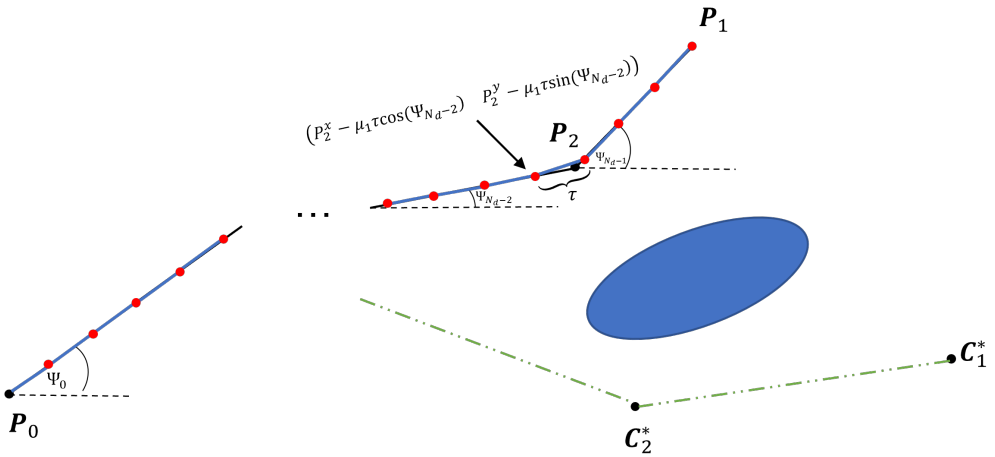

Figure 3.24: An example of path obtained during baktrack$_i$ strategy

To choose the proper value to assign as drone goal, vector $\boldsymbol{t} = \begin{bmatrix} t_1 & \ldots & t_i & \ldots & t_{N_d} \end{bmatrix}^T$,

has to be initialized correctly. A proper initialization for the backtrack phase can be

$$\boldsymbol{t} = \begin{bmatrix} \bar{t} \\ \bar{t}+4 \\ \vdots \\ \bar{t}+4(N_d-1) \end{bmatrix} \tag{3.66}$$

where, the expression $\Gamma(t(i))$ means to select the row $t(i)$, $i = 1, \dots, N_d$, from the matrix Γ. Choosing the value $\bar{t}$ means to select a proper point of the path as goal for the drone$_1$. From the choice of $\bar{t}$, the vector $\boldsymbol{t}$ is entirely constructed. The next concepts to introduce is the generic behaviour of the system during the "backtrack$_i$" strategy. When one of the "backtrack$_i$" strategy is chosen, it means that we want to have j drones, $j = 1, \dots, i$, backtracked towards $i+1$ drone (except for the strategy "backtrack$_{N_d}$" where we want all drones i, $i = 1, \dots, N_d$, backtracked towards the ground station). In Figure (3.25) we can see the behaviour of this strategy.

STRATEGY	MEANING	STEPS
backtrack$_1$	drone$_1$ towards drone$_2$	$x_{i_{1,2}}(j\|k) \in D_i, \forall i \in \mathbb{N}_1^{N_d}, \forall j \in \mathbb{N}_0^N$ $\boldsymbol{x}_{g_1} = \Gamma(t(1))^T$ $\boldsymbol{x}_{g_i} = \boldsymbol{P}_i, \forall i \in \mathbb{N}_2^{N_d}$ **IF** at time k, $dist(\boldsymbol{P}_1, \boldsymbol{P}_2) < \epsilon$ **THEN** exit backtrack$_1$
backtrack$_2$	drone$_i$ $i = 1,2$ towards drone$_3$	$x_{i_{1,2}}(j\|k) \in D_i, \forall i \in \mathbb{N}_1^{N_d}, \forall j \in \mathbb{N}_0^N$ $\boldsymbol{x}_{g_1} = \Gamma(t(1))^T, \boldsymbol{x}_{g_i} = \boldsymbol{P}_i, \forall i \in \mathbb{N}_2^{N_d}$ **IF** at time k, $dist(\boldsymbol{P}_1, \boldsymbol{P}_2) < \epsilon$ **THEN** $\boldsymbol{x}_{g_i} = \Gamma(t(i))^T, \forall i \in \mathbb{N}_1^2, \boldsymbol{x}_{g_i} = \boldsymbol{P}_i, \forall i \in \mathbb{N}_3^{N_d}$ **IF** at time k, $x_{1_{1,2}}(j\|k) \in D_1 \cap D_2, \forall j \in \mathbb{N}_0^N$ **THEN** $x_{1_{1,2}}(j\|k) \in D_1 \cap D_2, x_{2_{1,2}}(j\|k) \in D_2, \forall j \in \mathbb{N}_0^N$ $\boldsymbol{x}_{g_i} = \Gamma(t(i))^T, \forall i \in \mathbb{N}_1^2, \boldsymbol{x}_{g_i} = \boldsymbol{P}_i, \forall i \in \mathbb{N}_3^{N_d}$ **IF** at time k, $dist(\boldsymbol{P}_2, \boldsymbol{P}_3) < \epsilon \vee exit\ condition_2$ **THEN** exit backtrack$_2$
⋮	⋮	⋮
backtrack$_{N_d}$	drone$_i$ $i = 1, \dots, N_d$ towards ground station	$x_{i_{1,2}}(j\|k) \in D_i, \forall i \in \mathbb{N}_1^{N_d}, \forall j \in \mathbb{N}_0^N$ $\boldsymbol{x}_{g_1} = \Gamma(t(1))^T, \boldsymbol{x}_{g_i} = \boldsymbol{P}_i, \forall i \in \mathbb{N}_2^{N_d}$ **IF** at time k, $dist(\boldsymbol{P}_1, \boldsymbol{P}_2) < \epsilon$ **THEN** $\boldsymbol{x}_{g_i} = \Gamma(t(i))^T, \forall i \in \mathbb{N}_2^{N_d}$ **IF** at time k, $x_{1_{1,2}}(j\|k) \in D_1 \cap D_2, \forall j \in \mathbb{N}_0^N$ **THEN** $x_{1_{1,2}}(j\|k) \in D_1 \cap D_2, x_{2_{1,2}}(j\|k) \in D_2, \forall j \in \mathbb{N}_0^N$ $\boldsymbol{x}_{g_i} = \Gamma(t(i))^T, \forall i \in \mathbb{N}_1^2, \boldsymbol{x}_{g_i} = \boldsymbol{P}_i, \forall i \in \mathbb{N}_3^{N_d}$ ⋮ **IF** at time k, $dist(\boldsymbol{P}_{N_d-1}, \boldsymbol{P}_{N_d}) < \epsilon$ **THEN** $\boldsymbol{x}_{g_i} = \Gamma(t(i))^T, \forall i \in \mathbb{N}_1^{N_d}$ **IF** at time k, $x_{N_d-1_{1,2}}(j\|k) \in D_{N_d-1} \cap D_{N_d}, \forall j \in \mathbb{N}_0^N$ **THEN** $x_{i_{1,2}}(j\|k) \in D_i \cap D_{i+1}, \forall i \in \mathbb{N}_1^{N_d-1}$ $x_{N_d}(j\|k) \in D_{N_d}, \forall j \in \mathbb{N}_0^N$ $\boldsymbol{x}_{g_i} = \Gamma(t(i))^T, \forall i \in \mathbb{N}_1^{N_d}$ **IF** at time k, $dist(\boldsymbol{P}_0, \boldsymbol{P}_{N_d}) < \epsilon \vee exit\ condition_{N_d}$ **THEN** exit backtrack$_{N_d}$

Figure 3.25: An overview of backtrack$_i$ strategies

In this Figure, we can see how the position constraints and the vector $\boldsymbol{x}_g$, which are used in the FHOCP, are changed during the simulation, according to specific rules presented in the column "STEPS". The variable ϵ is a user-defined variable. For what concerns the "exit condition$_i$", $i = 2, \dots, N_d$, it represents a valid condition to terminate the backtrack algorithm. In fact, in some cases, it is not necessary that this algorithm continues working, since it may happen that the leader drone can see a free target. To express this condition is necessary to define some triangles. If a "backtrack$_i$", with $i \geq 2$, algorithm is chosen, then, we have to define $N_T = i$-1 triangles that are created from $\boldsymbol{P}_1$ and $\boldsymbol{P}_j$,$\boldsymbol{C}_j^*$, $j = 2, \dots, i$. If the strategy "backtrack$_1$ is chosen, the exit condition is not present. The triangles defined in this way are summarized in the table (3.4).

T_{E_1}	$\boldsymbol{P}_1$	$\boldsymbol{P}_2$	$\boldsymbol{C}_2^*$
T_{E_2}	$\boldsymbol{P}_1$	$\boldsymbol{P}_3$	$\boldsymbol{C}_3^*$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$T_{E_{N_T}}$	$\boldsymbol{P}_1$	$\boldsymbol{P}_{N_T}$	$\boldsymbol{C}_{N_T}^*$

Table 3.4: An example of triangles computed during backtrack algorithm

At this point, a feasibility check has to be done to explore which triangles are free and consequently, which target can be reached by the leader drone. To do that, the same LP presented in subsection (3.4.3) is solved with the aforementioned triangles. The vector $\boldsymbol{\varrho} = \begin{bmatrix} \varrho_1 & \varrho_2 & \dots & \varrho_{N_T} \end{bmatrix}$ is filled with the output of the feasibility check, where ϱ_i is a boolean variable: if it is 1, it means that T_{E_i} is not free, if it is 0 it means that the triangle is free. The exit conditions are described in the figure (3.26)

Exit condition$_2$	**IF** $T = \{P_1, P_2, C_2^*\}$ free $\rightarrow$ **TRUE**
Exit condition$_3$	**IF** $T = \{P_1, P_2, C_2^*\}$ free $\vee$ $T = \{P_1, P_3, C_3^*\}$ free $\rightarrow$ **TRUE**
$\vdots$	$\vdots$
Exit condition$_{N_d}$	**IF** $T = \{P_1, P_2, C_2^*\}$ free $\vee \dots \vee$ $T = \{P_1, P_{N_d}, C_{N_d}^*\}$ free $\rightarrow$ **TRUE**

Figure 3.26: An overview of the exit conditions

When the exit condition is true, the backtrack algorithm is terminated in this way:

$$\boldsymbol{x}_{g_i} = \Gamma \left(\bar{t}(i)\right)^T, \quad \forall i = 1, \dots, N_d, \tag{3.67}$$

where $\bar{t}(i)$ is the index which represents the point on the path which fulfill all the conditions reported in Figures (3.25) and (3.26). An example of "backtrack$_2$" is depicted in Figure (3.27).

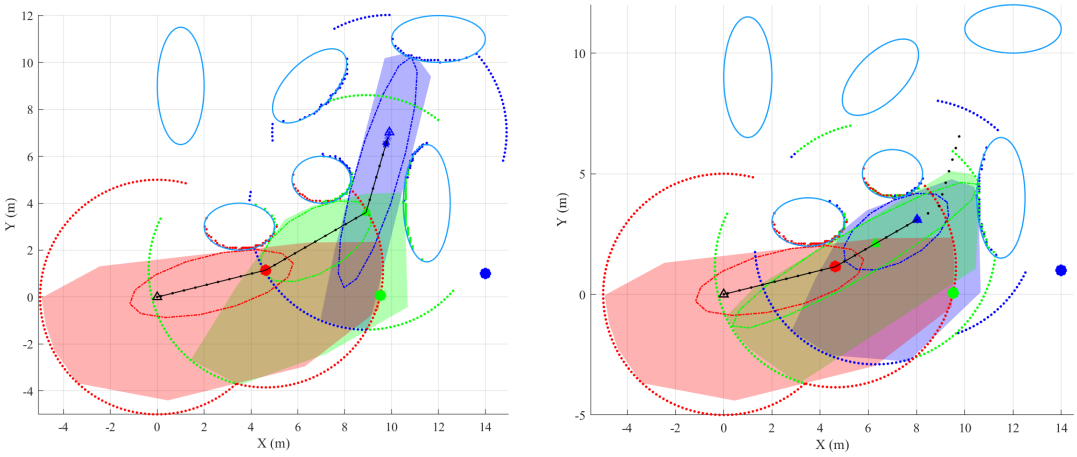

Figure 3.27: An example of "backtrack$_2$" strategy. The colored triangles represent the drones, while the black one is the ground station. The colored big dots are the new targets computed by the offline path planner. In blue are depicted the obstacles. On the left, it is represented the first iteration of the path following. On the right, the backtrack strategy terminates because the exit condition is respected, in fact, the triangle which has $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{C}_2^*$ as vertices, is free.

An example of "backtrack$_3$" is given in Figure 3.28.

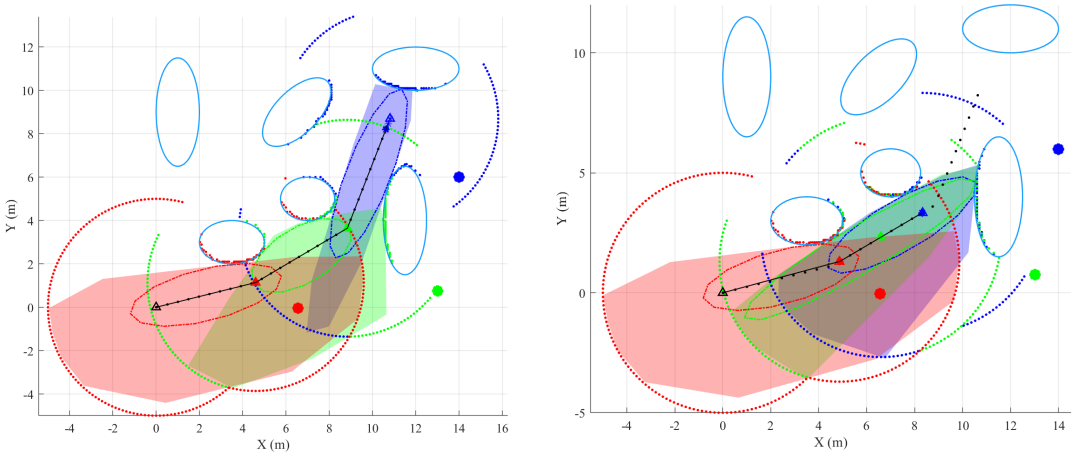

Figure 3.28: An example of "backtrack$_3$" strategy. The colored triangles represent the drones, while the black one is the ground station. The colored big dots are the new targets computed by the offline path planner. In blue are depicted the obstacles. On the left, it is represented the first iteration of the path following. On the right, the backtrack strategy terminates because the exit condition is respected, in fact, the triangle which has $\boldsymbol{P}_1, \boldsymbol{P}_3, \boldsymbol{C}_3^*$ as vertices, is free.

At this point, the backtrack algorithm is completed and a new path following algorithm is needed to bring the drones to their relative targets. The same steps of the subsection (3.4.2) are done. Considering $\boldsymbol{P}_0$ as the initial point of the path and $\boldsymbol{C}_{N_d}^*$ as the final

point, it is possible to compute matrix Γ and the vector $\boldsymbol{\iota}$:

$$\Gamma = \begin{bmatrix} P_0^x & P_0^y \\ P_0^x + \tau\cos(\Psi_0) & P_0^y \tau\sin(\Psi_0) \\ P_0^x + 2\tau\cos(\Psi_0) & P_0^y + 2\tau\sin(\Psi_0) \\ \vdots & \vdots \\ C_{N_d}^{*,x} + \mu_1\tau\cos(\Psi_1) & C_{N_d}^{*,y} + \mu_1\tau\sin(\Psi_1) \\ \vdots & \vdots \\ C_2^{*,x} + \mu_{N_d-1}\tau\cos(\Psi_{N_d-1}) & C_2^{*,y} + \mu_{N_d-1}\tau\sin(\Psi_{N_d-1}) \\ \vdots & \vdots \\ C_1^{*,x} & C_1^{*,y} \end{bmatrix}, \quad \boldsymbol{\iota} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \\ \vdots \\ N_d - 1 \\ \vdots \\ N_d - 1 \end{bmatrix}, \tag{3.68}$$

where Ψ_i, $i = 0, \ldots, N_d - 1$ corresponds to angle between tether i and x-axis, τ is a user-defined variable and μ_i is a coefficient which guarantees that all the points on the path are spaced by τ meters. The new vector $\boldsymbol{\iota}$ is constructed in such a way to connect the point composing the path to the tether they belong. An example of path creation with the vector $\boldsymbol{\iota}$ is depicted in Figure (3.29).

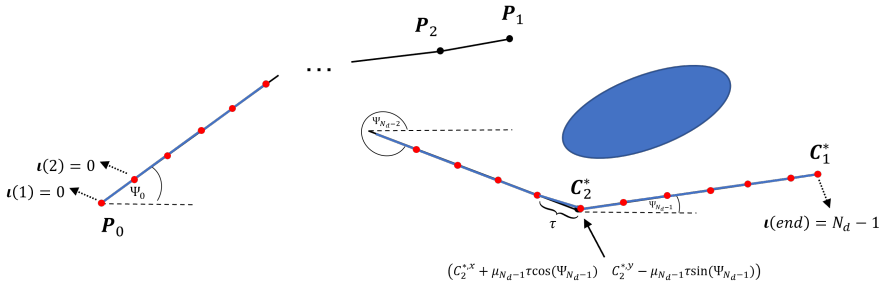

Figure 3.29: An example of path with vector $\boldsymbol{\iota}$

Now, to initialize the vector $\boldsymbol{t}$ after the backtrack phase, we have to recall the exit condition explained before. In fact, starting from them, the vector is initialized. As we know, if the triangle with vertices $\boldsymbol{P}_1$, $\boldsymbol{P}_i$ and $\boldsymbol{C}_i^*$ is considered free during the backtrack phase, it means that the target $\boldsymbol{C}_i^*$ is reachable for the leader drone. Furthermore, we just have to select from the vector $\boldsymbol{\iota}$ the index corresponding to the point of the path which comes before the target $\boldsymbol{C}_i^*$, which we call $\bar{\iota}$:

$$\boldsymbol{t} = \begin{bmatrix} \bar{\iota} \\ \bar{\iota} - 4 \\ \vdots \\ \bar{\iota} - 4(N_d - 1) \end{bmatrix}, \quad \boldsymbol{x}_{g_i} = \left[\Gamma(t(i))^T\right], \quad \forall i = 1, \ldots, N_d \tag{3.69}$$

An example of second path following after the backtrack phase is reported in Figure (3.30).

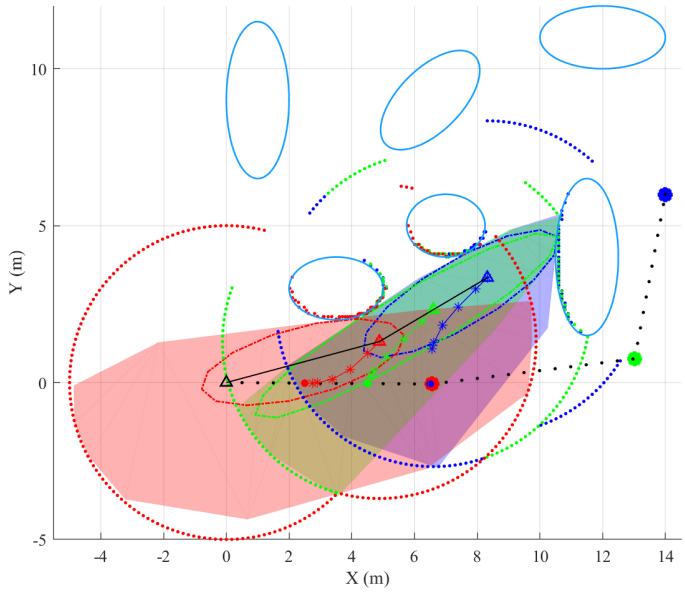

Figure 3.30: An example of path following after backtrack phase. The colored triangles represent the drones, while the black one is the ground station. The colored big dots are the new targets computed by the offline path planner. In blue are depicted the obstacles. As we can see the target $\boldsymbol{C}_3^*$ is considered reachable for the leader drone. In fact, the goal of the drones are assigned starting from $\bar{\imath}$, the index related to the point of the path which precedes the target $\boldsymbol{C}_1^*$.

Otherwise, if the backtrack phase is terminated for distance reason (see Figure 3.25)) and not with reference to exit condition, then, vector $\boldsymbol{t}$ and vector of goals are initialized in the same manner of the subsection (3.4.2):

$$\boldsymbol{t} = \begin{bmatrix} \bar{t} + 4(N_d - 1) \\ \vdots \\ \bar{t} + 4 \\ \bar{t} \end{bmatrix}, \boldsymbol{x}_{g_i} = \begin{bmatrix} \Gamma(t(i)) \end{bmatrix}^T, \quad \forall i = 1, \ldots, N_d \tag{3.70}$$

4 | Results

In this chapter the results obtained in numerical simulations are presented. The simulations are done with MATLAB and using different solvers. More in the details, the MATLAB function "fmincon", which is a Sequential Quadratic Programming (SQP) solver, is used to solve the non-linear offline path planner presented in section (3.2) (see (3.3)) and in section (3.5) (see (3.53)). In this solver, the gradient is calculated with Central finite Differences method (CD), to have a more precise value of the gradient, while BFGS (Broyden, Fletcher, Goldfarb, Shanno) method is used to calculate the hessian. This solver has many parameters which can be modified in order to obtain a different optimal solution. In fact it is possible to increase the maximum step length, especially when the target point of the leader is far away from the initial position of the drones. CPLEX solver [6] is used to solve the LP related to the optimal ellipse in section (3.3) (see (3.16)), the problem of obstacle check (3.4.3) (see (3.42)) and the problem of triangle method in subsection (3.5.1). In conclusion, YALMIP [9] with "fmincon" set as solver is used to solve the FHOCP seen in section (3.4) (see (3.32)) and in section (3.5). The script, every sampling time T_s, solves these problems and generates the new reference for the drones. Geometric parameter of ellipses used in environment are resumed in table (4.1).

Ellipse	$\boldsymbol{x_0}$ (m)	$\boldsymbol{y_0}$ (m)	$\boldsymbol{a}_e$	$\boldsymbol{b}_e$	$\boldsymbol{\alpha}$ (rad)
1	3.5	3	1.5	1	0
2	12	11	2	1	0
3	11.5	4	2.5	1	$\pi/2$
4	7	5	1	1.25	$\pi/2$
5	6.5	9	2	1	$\pi/4$
6	1	9	1	2.5	π

Table 4.1: Geometric properties of obstacles

where x_0, y_0 are (x, y) coordinate of the ellipse center, a, b are respectively the semi-major and semi-minor axes, α is the rotation angle with respect x axis. The parameter used in simulations can be resumed in the Table (4.2).

	Meaning	Value
$\boldsymbol{T_s}$	sampling time	0.5s
$\boldsymbol{N}$	prediction horizon of FHOCP	5
$\boldsymbol{d_{emergency}}$	offset used in polytope reduction	0.5m
$\boldsymbol{\bar{l}}$	maximum extension of the cable	8m
$\boldsymbol{\underline{l}}$	minimum extension of the cable	2m
$\boldsymbol{\underline{d}}$	minimum permitted distance between drones	1.5m
$\boldsymbol{\overline{V}}$	maximum velocity of drones	1m/s
$\boldsymbol{\overline{A}}$	maximum acceleration of drones	2m/s^2
$\boldsymbol{\delta}$	parameter used in offline path planner	1m
$\boldsymbol{\sigma}$	parameter used in offline path planner	1m
$\boldsymbol{v_1}$	increment used in follow strategy	2
$\boldsymbol{v_2}$	increment used in backtrack strategy	1

Table 4.2: Simulation parameters

4.1. Known environment

The simulations presented in this section are related to scenarios where all the obstacles are known by the drones, so the first offline optimisation problem finds an optimal configuration which is for this reason an obstacle free path. The initial condition of the drones and their respectively target are presented in Figure (4.1). The solver finds a set of admissible configuration starting from the target assigned to the leader drone. From that, the path following approach permits to create a discrete path to be followed. The solver computes at each T_s the optimal trajectory which the drones have to follow, considering the restricted polytope as feasible convex area. The first numerical simulation we present starts with the drones and the ground station in these positions:

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_1 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_2 = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_3 = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix},$$

while the targets assigned to the leader drone are:

$$\boldsymbol{P}_{target^1} = \begin{bmatrix} 11.5 \\ 8.5 \end{bmatrix}, \quad \boldsymbol{P}_{target^2} = \begin{bmatrix} 14 \\ 6 \end{bmatrix}$$

In Figure (4.2) are shown the optimal predicted trajectory found by the solver over the horizon N.

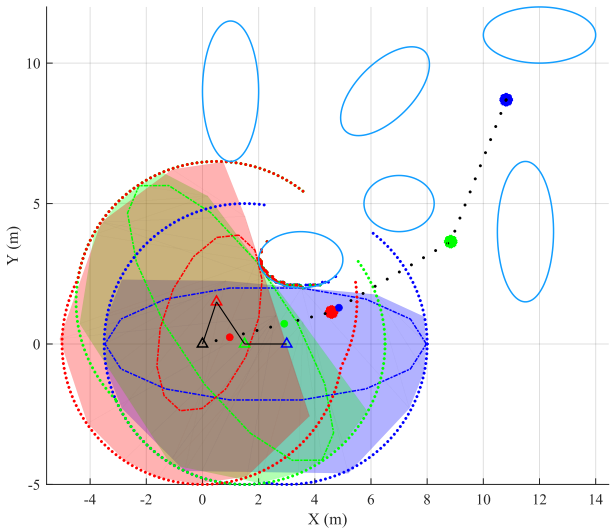

Figure 4.1: A representation of initial conditions of the drones, which are the colored triangles, the big colored dots represent the targets, while the black dots describe the path. The ellipses depicts the obstacles. The dashed lines are the discretized optimal ellipses, from whom are retrieved the colored areas which correspond to the convex approximation of the free space.

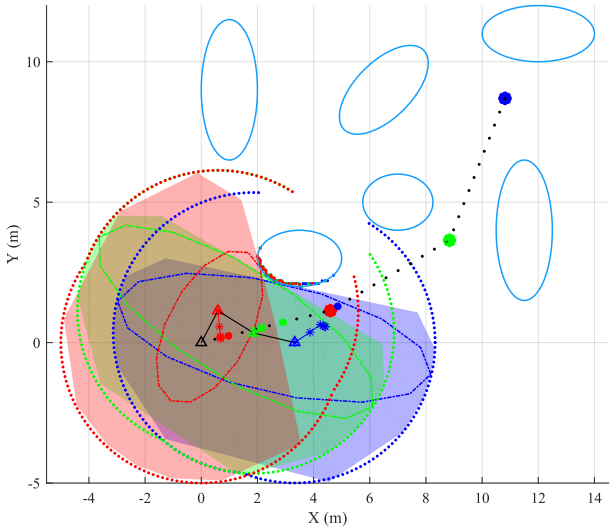

Figure 4.2: A representation of the planner in action. Big colored dots denote the targets $\boldsymbol{C}_i^*$, while the small colored dots are tracked goal of the drones $\boldsymbol{x}_{g_i}$, line with '*' are the predicted trajectory over the horizon N. Points represent the merged LiDAR readings while the colored surfaces are the polytopes. In blue the known obstacles.

After some iterations the targets are perfectly reached by the drones as we can see in Figure (4.3).

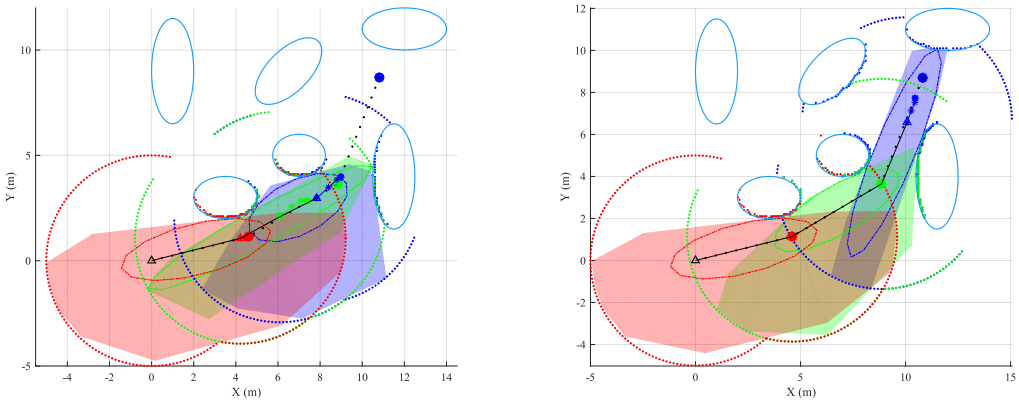

Figure 4.3: An example of path following during simulations. Big colored dots are the targets, denote the tracked goal of the drones.

In Figure (4.4) are depicted the plots related to the position of the drones during the simulation. Similarly, the constraints on maximum velocity are respected as well, as we can see in Figure (4.5). The trajectories and the final configuration reached after the first path following are shown in Figure (4.6).

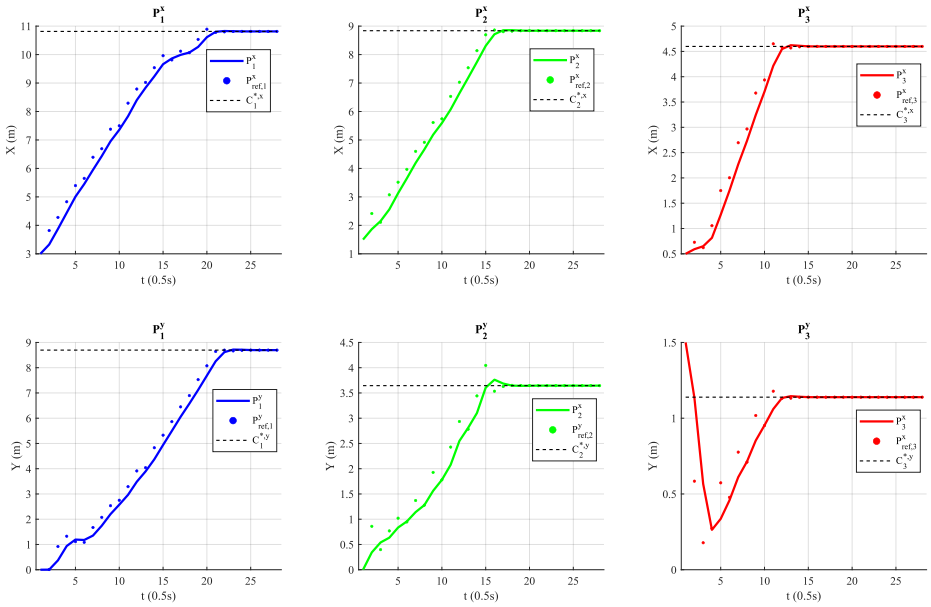

Figure 4.4: A representation of the position of the drones during the simulation. Dashed lines are the relative target, dots represent optimal input u^*.

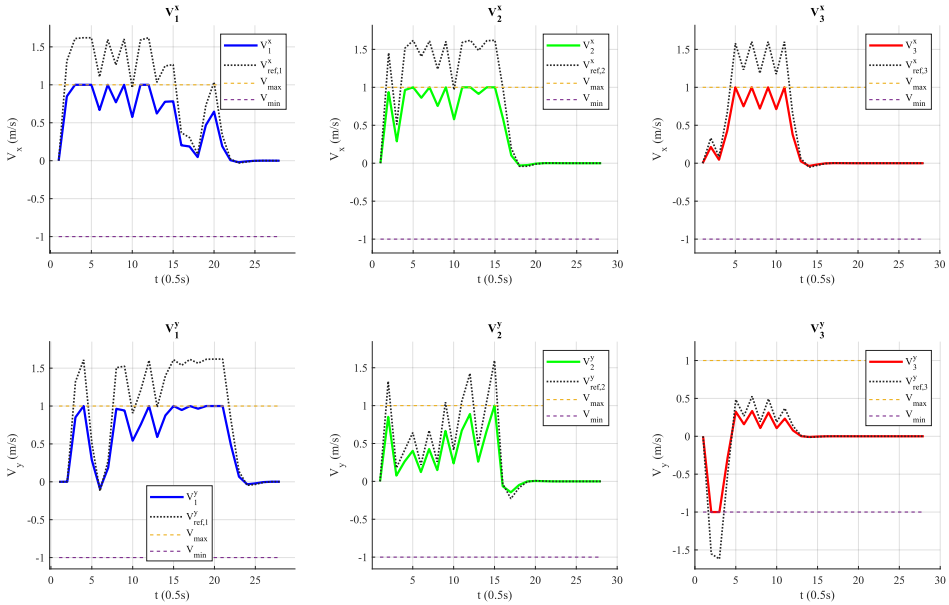

Figure 4.5: A representation of the velocities of the drones during the simulation. The black dashed line represent the reference velocity, while the dashed colored lines depict the boundaries.

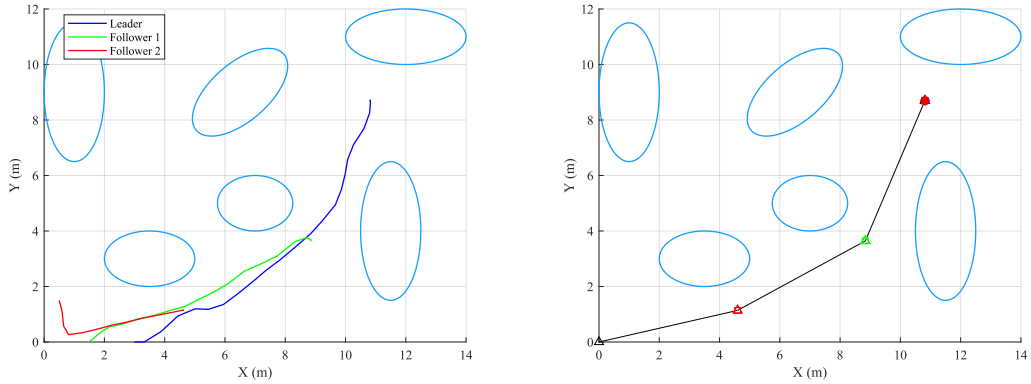

Figure 4.6: Representation of trajectories and final configuration. In blue are represented obstacles. On the left, the three colored lines represent the trajectories done by the drone during the simulation. On the right, it is depicted the final position of the drones, which are shown with colored triangles, while the ground station is the black one.

At this point, when the first path following is concluded, the second offline path planner receives in input the second target assigned to the leader drone and is able to find a new optimal obstacle free configuration $\boldsymbol{C}^*$ for all the drones as we can see in Figure (4.7). Backtrack type is chosen by the strategy algorithm, more in the details backtrack$_3$ strategy. In fact, the solution of offline path planner consists on moving the three drones. Every T_s, the algorithm checks when it is possible to stop the strategy. The backtrack

strategy is depicted in Figure (4.8).

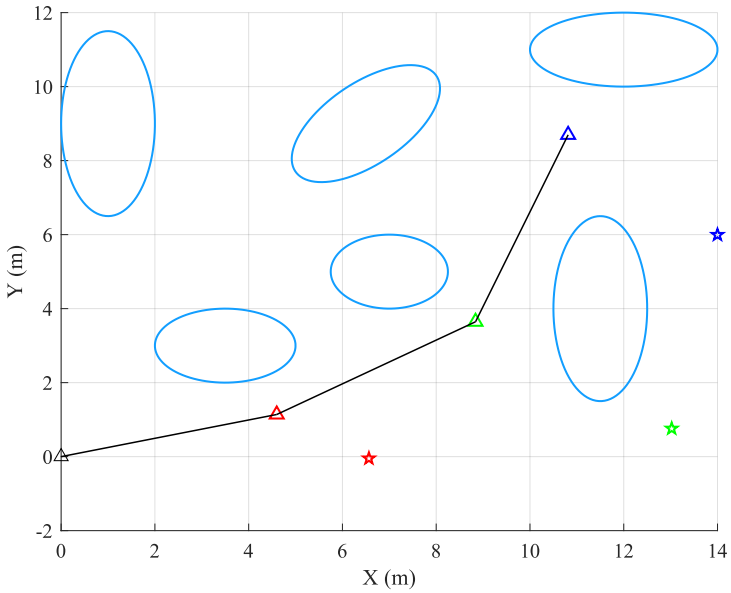

Figure 4.7: A representation of second offline optimisation. The three stars represent the new target of the drones, while the triangles depict their actual position, except for the black one which is the ground station. In blue are shown the obstacles.

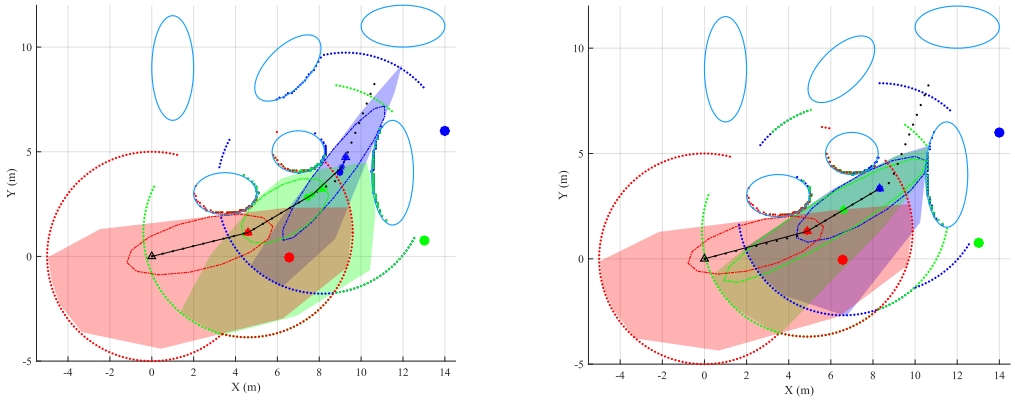

Figure 4.8: A representation of backtrack strategy. The triangles represent the actual drone position, while the big colored dots represent their new target. The points are the LiDAR merged readings $L_{i,i+1}(k)$, while the colored surfaces are the polytopes $D_j(k)$. On the left, a plot of the simulation, on the right, the target $\boldsymbol{C}_1^*$ is reachable by the leader drone, so the exit condition is true and the backtrack phase is terminated.

The first iteration of the second path following is illustrated in the Figure (4.9). As we can see, since the target $\boldsymbol{C}_3^*$ is reachable for the leader, it means that the goal of the

leader, $\boldsymbol{x}_{g_1}$, is set as the point of the path which comes before the target. The other goals are simply scaled starting from that one, as we know from the theory.

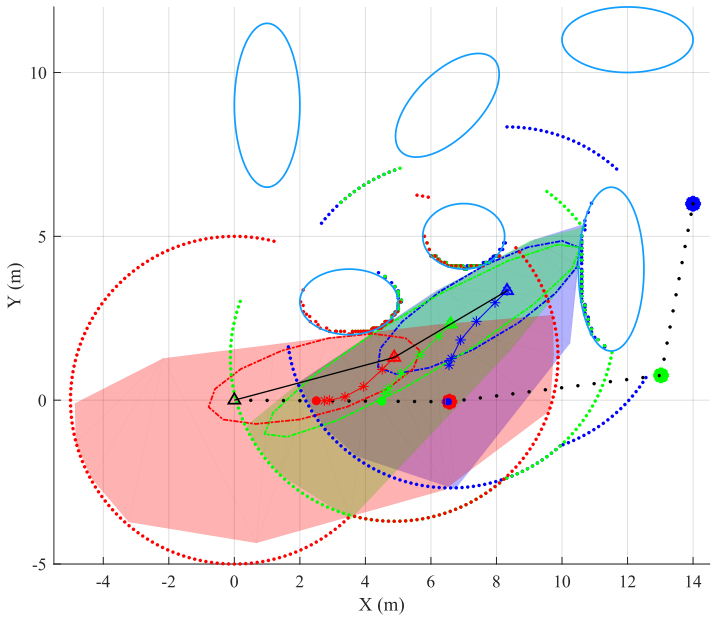

Figure 4.9: A representation of the second path following. The drones moves toward their goals, which are depicted as small colored dots, while small black dots represent the path to be followed. Big dots denote the tracked optimal configuration $\boldsymbol{C}^*$, line with $'*'$ are the predicted trajectories calculated over the horizon N.

The second path following approach finishes after some iterations, in fact the prefixed targets are perfectly reached by the drones. The Figure (4.10) shows this process.

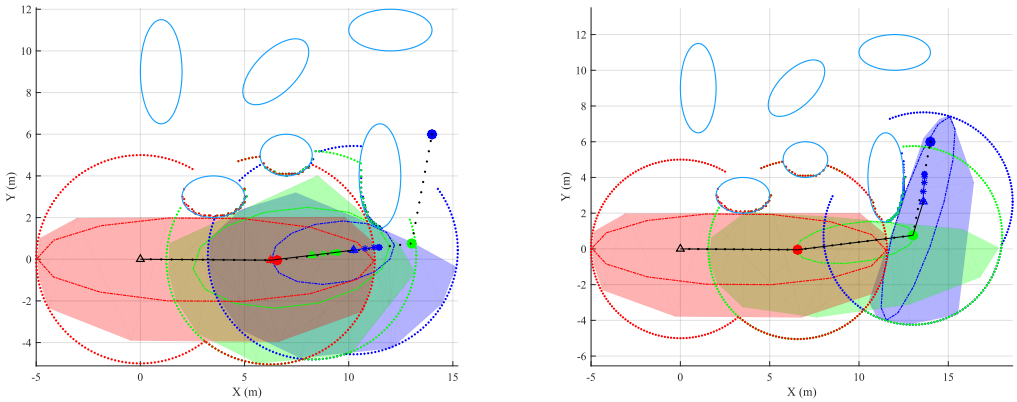

Figure 4.10: A representation of the second path following

The Figure (4.11) depicts drones position during backtrack and second path following,

while Figure (4.12) represent how the velocity changes during the simulation. The constraints on velocity as it can be seen are totally respected. Trajectories and final configuration reached by drones are reported in Figure (4.13)

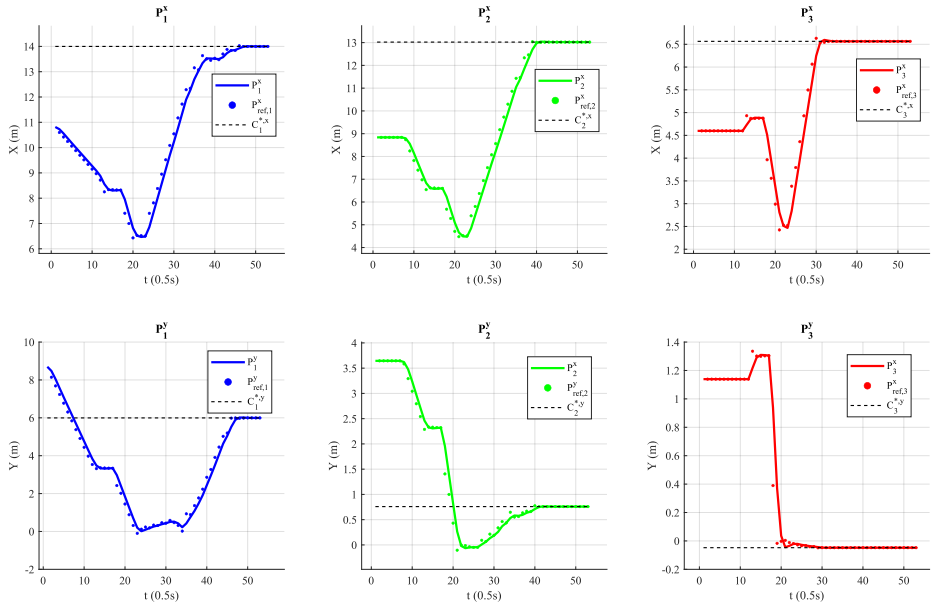

Figure 4.11: A representation of the position of the drones during the simulation. Dashed lines are the relative target, dots represent optimal input u^*.

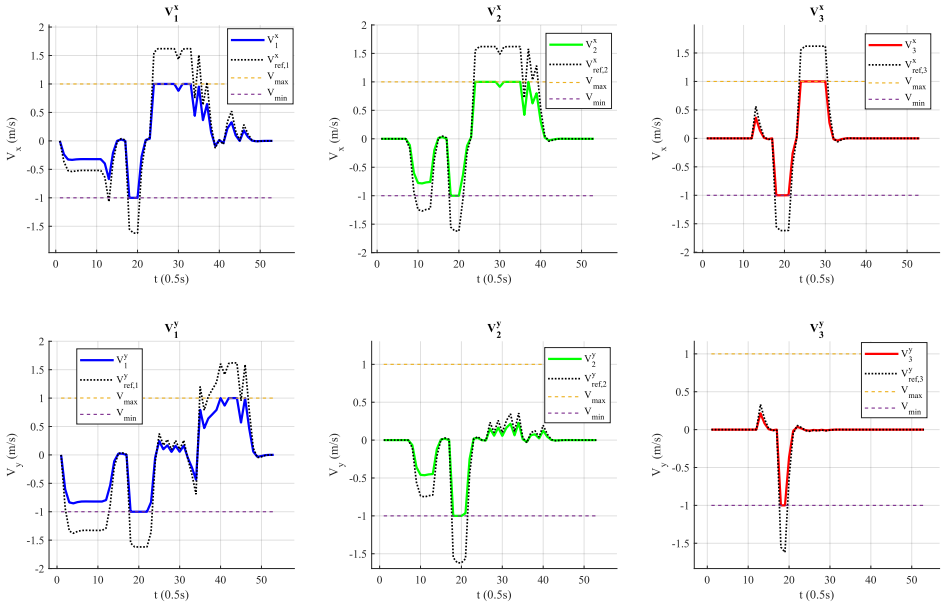

Figure 4.12: A representation of the velocities of the drones during the simulation. Colored dashed lines correspond to the boundaries, while colored dots are the velocity references.

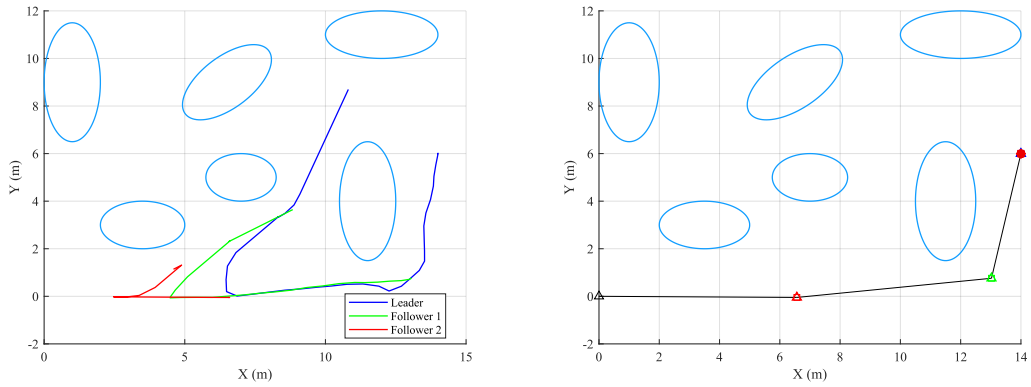

Figure 4.13: Representation of trajectories and final configuration of the drones involving backtrack strategy and second path following step. In blue are represented obstacles. On the left, the three colored lines represent the trajectories done by the drone during the simulation. On the right, it is depicted the final position of the drones, which are shown with colored triangles, while the ground station is the black one.

The constraints related to the distance between drones as it can be seen in Figure (4.14) are respected as well. In fact, the minimum distance is sat to be $\underline{d} = 1.5m$ (see Table (4.2)). The computational time of each iteration (figure 4.15) is quite low, in fact in all the simulation the mean value is about $0.2s$.

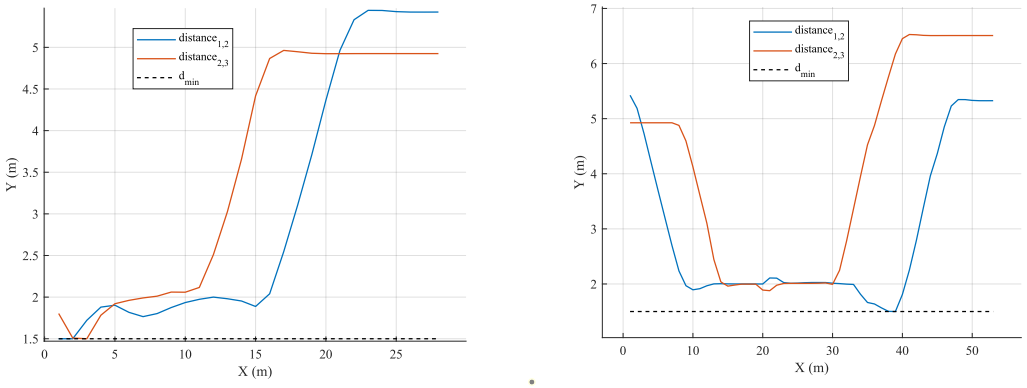

Figure 4.14: Representation of distance between drones during the simulation. On the left, the distance in the first path following, while on the right, the distance in backtrack and second path following phases.

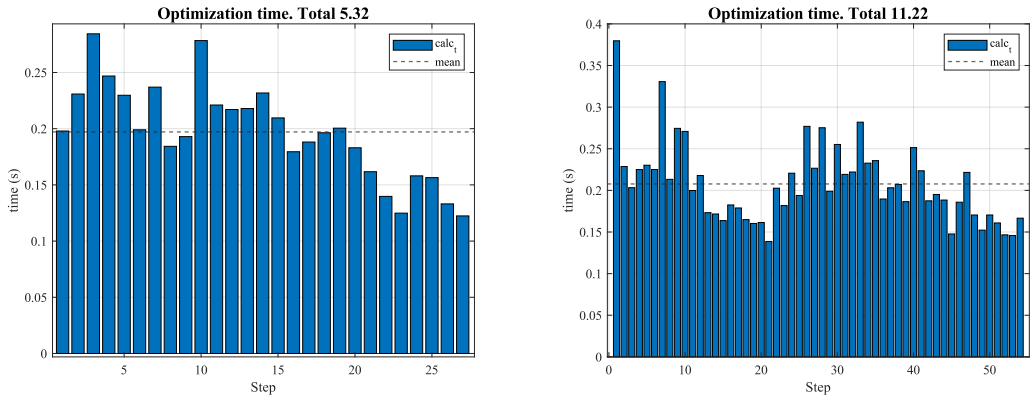

Figure 4.15: A representation of the computational time at each iteration. On the left, optimisation time of the first path following, while on the right computational time of backtrack and second path following. In blue are reported the time of each iteration, dashed line represents the mean value.

4.2. Unknown obstacles

The second set of simulations presented concern unknown obstacles in the environment which are not known a priori by offline optimisation algorithm. It means that the optimal target points are not found taking into account these obstacles. In this example the function which has to check unknown obstacles is tested. The initial conditions of the drones are different from the ones seen in the previous example:

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_1 = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_2 = \begin{bmatrix} -1.5 \\ 1.5 \end{bmatrix}, \quad \boldsymbol{P}_3 = \begin{bmatrix} -3.5 \\ 2 \end{bmatrix},$$

while the targets assigned to the leader drone are:

$$\boldsymbol{P}_{target^1} = \begin{bmatrix} 1 \\ 14 \end{bmatrix}, \quad \boldsymbol{P}_{target^2} = \begin{bmatrix} 10 \\ 15 \end{bmatrix}$$

The first path following is executed and the drones reach their targets even if an unknown obstacle is present near the target of the leader drone. In the Figure (4.16) are shown the initial conditions of the drones.

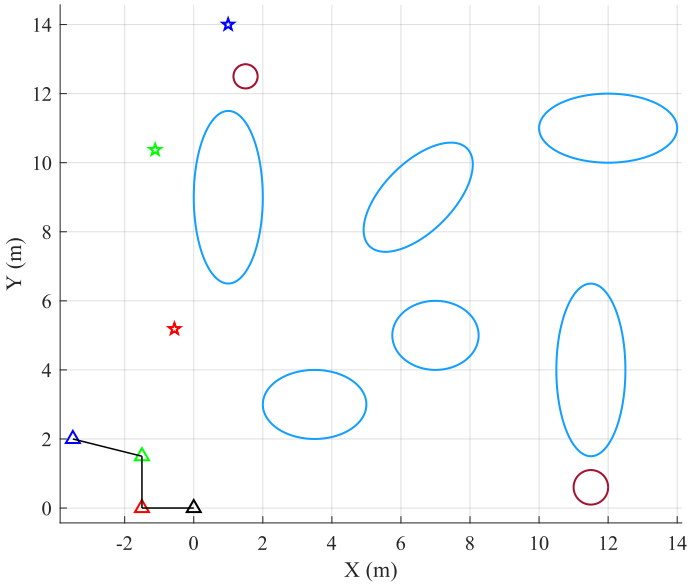

Figure 4.16: Representation of initial conditions of the drones with unknown obstacles. In blue are represented known obstacles, while in red the unknown ones. Stars are the optimal configuration found by first offline optimisation, while triangles are the drones initial position, except for the black one, which is the ground station.

The path following approach brings the drones to their targets $\boldsymbol{C}_i^*$ as we can see in Figure (4.17).

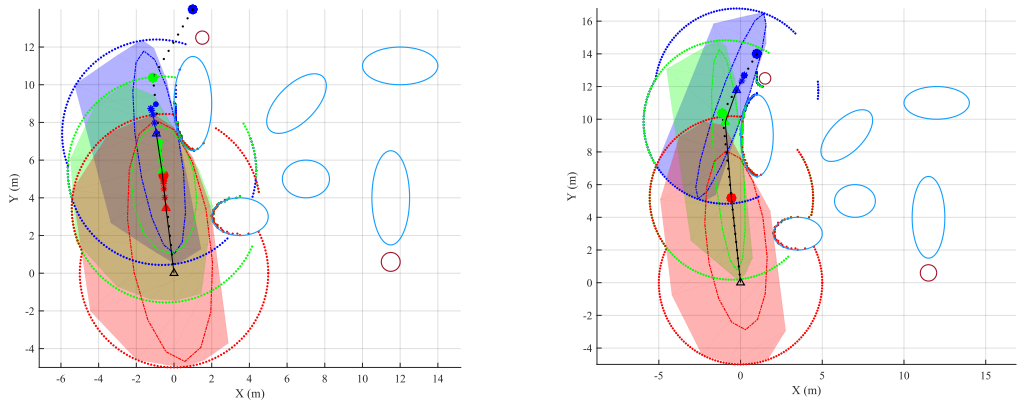

Figure 4.17: A representation of path following with unknown obstacles. In blue are represented known obstacles, while in red the unknown ones. Big colored dots are the targets of the drones, while small colored dots represent their goals.

The first path following lasts some iterations, when it is terminated the targets are perfectly reached by the drones as shown in Figure (4.18).

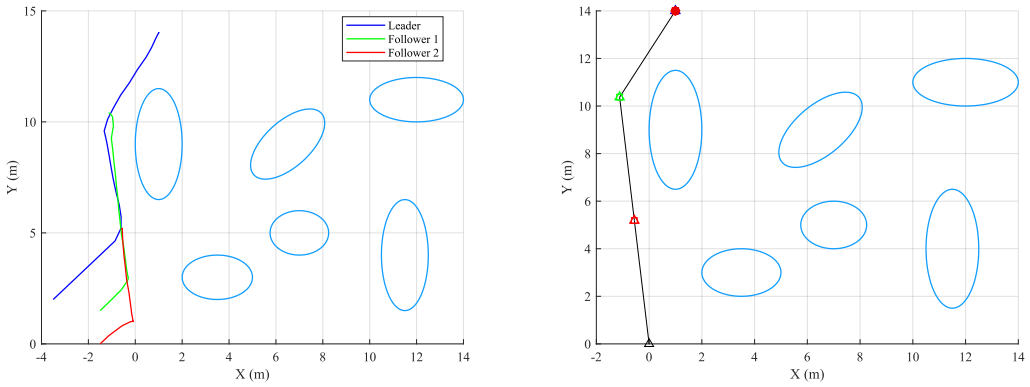

Figure 4.18: A representation of trajectories and final configuration after path following with unknown obstacles. In blue are represented known obstacles, while in red the unknown ones. Big colored dots are the targets of the drones, while small colored dots represent their goals.

Now, the second offline planner, starting from the actual position of the drones, finds a new optimal configuration $\boldsymbol{C}^*$, as it is shown in Figure (4.19).

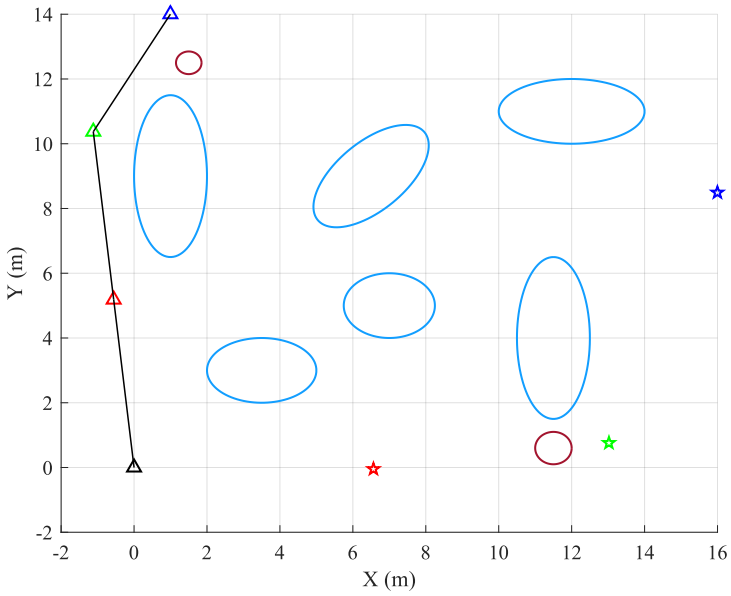

Figure 4.19: A representation of second offline optimisation with unknown obstacles. The stars represent the targets of the drones, while the triangles depict their actual position, excluding the black one, which represents the ground station. In blue are shown known obstacles, while in red unknown ones.

The red obstacle, on the right of the figure, which is unknown for the offline algorithm, is on the path. Even in this case the solution involves three drones moved, in fact, the strategy backtracking$_3$ is chosen, since the obstacles does not permit any other option.

This strategy is subsequently terminated once the exit condition is satisfied: drone_3 is close to the ground station. In the Figure (4.20) it is depicted the aforementioned strategy.

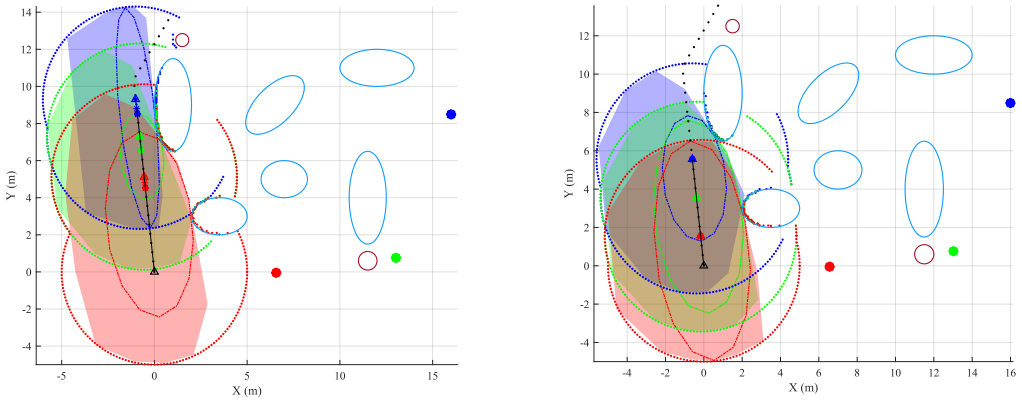

Figure 4.20: A representation of backtrack strategy with unknown obstacles.

Once the backtrack approach is terminated, the second path following starts. As expected, at a certain point during the simulation the leader drone encounters the unknown obstacle on the path. It is at this point that the function which checks obstacles, is activated. Automatically the actual position of the drones becomes their targets: $\boldsymbol{x}_{g_i} = P_i$, $i = 1, \ldots, 3$. In fact, after some iterations, the drones stop at that point. This process is is depicted in the Figure (4.21).

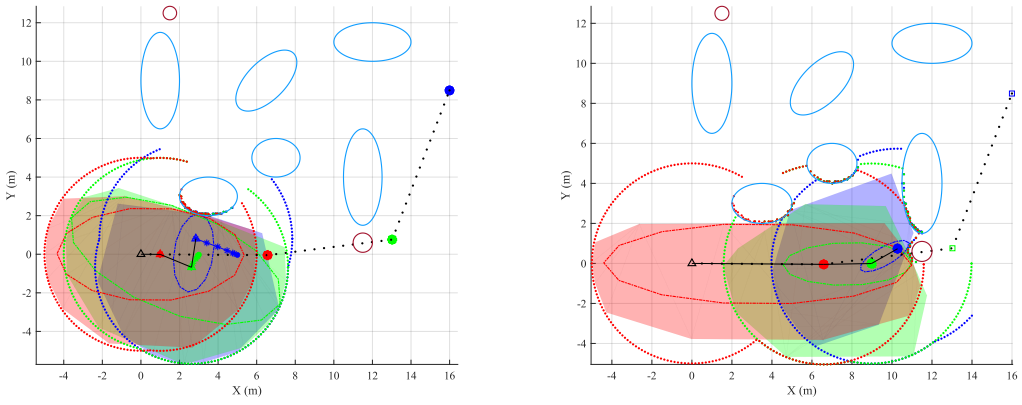

Figure 4.21: A representation of second path following. The stars represent the targets of the drones found by offline optimisation, while big dots depict their actual targets, imposed by obstacle check function. In red are shown unknown obstacles, while in blue the known ones.

In the second numerical simulation that we present the positions of the drones and the

target assigned to the leader are:

$$\boldsymbol{P}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_1 = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}, \quad \boldsymbol{P}_2 = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}, \quad \boldsymbol{P}_3 = \begin{bmatrix} 3 \\ 0 \end{bmatrix},$$

while the targets assigned to the leader drone are:

$$\boldsymbol{P}_{target^1} = \begin{bmatrix} 11 \\ 8.5 \end{bmatrix}, \quad \boldsymbol{P}_{target^2} = \begin{bmatrix} 14 \\ 6 \end{bmatrix}$$

As we can see from the Figure (4.22), the unknown obstacle is not on the path, it only occludes the range of sensors. As a consequence, the convex approximation of the free space is relatively smaller with respect to the same example proposed in "obstacle free" version, in section 4.1.

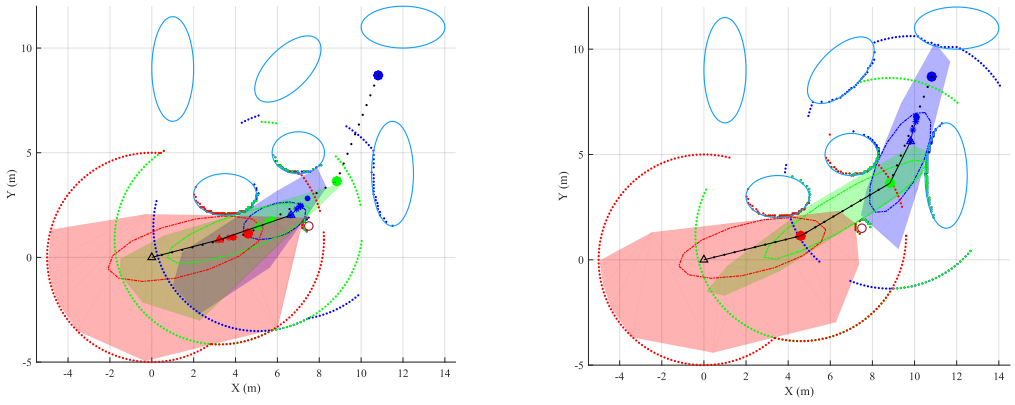

Figure 4.22: A representation of second path following. The stars represent the targets of the drones found by offline optimisation, while big dots depict their actual targets, imposed by obstacle check function. In red are shown unknown obstacles, while in blue the known ones.

The path following approach terminates whenever the targets are reached by the drones. The offline planner finds the same three optimal configuration of the "obstacle free" example. The result is reported in the Figure (4.23).

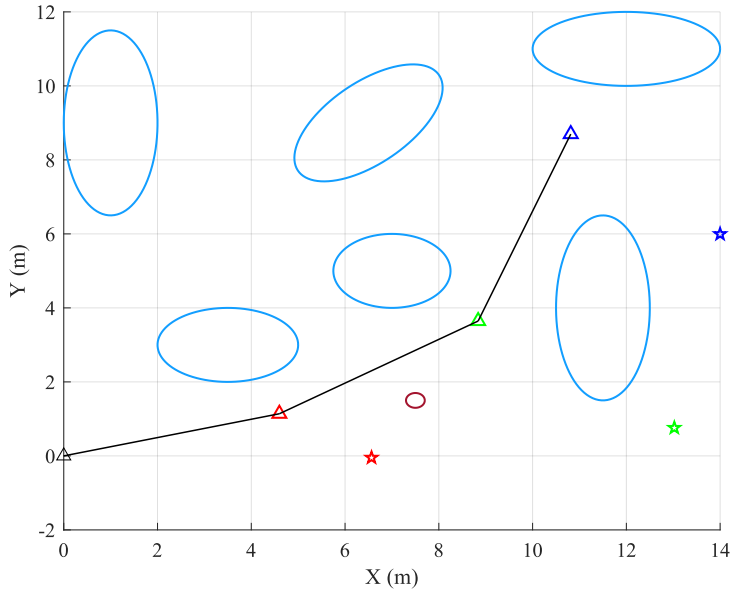

Figure 4.23: A representation of second offline optimisation. The three stars represent the new target of the drones, while the triangles depict their actual position, except for the black one which is the ground station. In blue are shown the obstacles.

Since the environment is full of obstacles, the unique proper strategy which can be chosen is the backtrack$_3$ one. The backtrack approach in this case terminates because the target $\boldsymbol{C}_3^*$ can be reached by the drone. This situation is depicted in Figure (4.24).

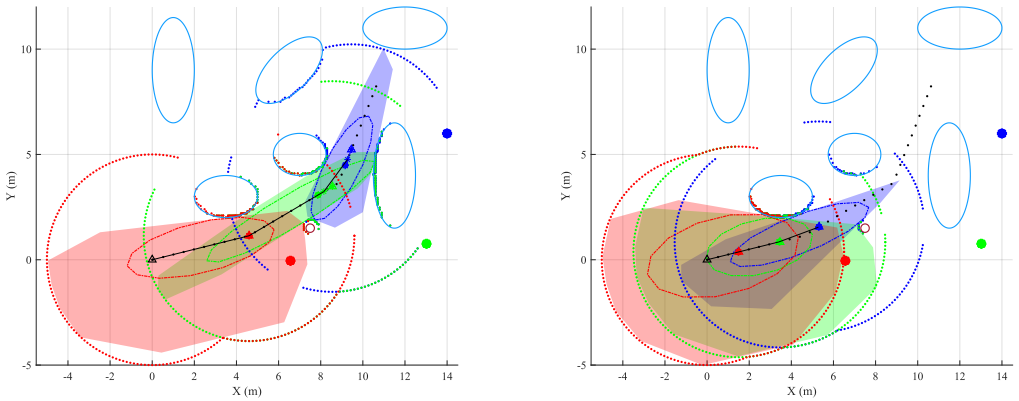

Figure 4.24: A representation of backtrack strategy with unknown obstacles. On the left, the leader drone and drone$_2$ backtrack towards drone$_3$. On the right, the algorithm is terminated because the exit condition is met.

Finally, the online path following algorithm has the aim to bring the drones to their respective target, nonetheless the unknown obstacle. In the Figure (4.25) it is reported this situation.

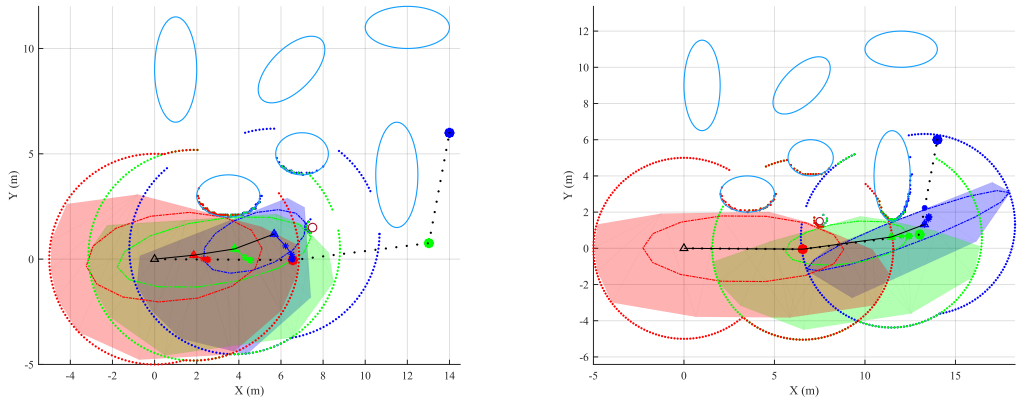

Figure 4.25: A representation of the second path following algorithm with unknown obstacles. On the left, it can be seen that the goal of the leader is assigned as the point which comes before the target $\boldsymbol{C}_3^*$. On the right, the algorithm continues working until all the targets are reached.

As we can see, the aforementioned path following algorithm is working even if an unknown obstacles is present. Since the focus in this section is related to the simulation environment provided with unknown obstacles, figure related to the drones position and velocity, computational time and distance constraints have been omitted. They are very similar to the previous example in section (4.1). All the constraints are satisfied and the computational time is low, with a mean value about $0.2s$.

5 | Conclusions and future developments

An approach to navigate systems of tethered quadcopters in a partially known environment has been presented, where an offline optimization problem computes the optimal configuration to reach a target considering the known obstacles. A real-time MPC algorithm allows to reach the desired configuration, involving some backtrack process if needed. The algorithm is able to stop the chain of drones if an insurmountable obstacle is present or to decide which is the better strategy which the system has to follow. A novel approach to approximate the free-space where the drones can move with a convex polytope able to guarantee that the vehicles and tethers remain in an obstacle-free area has been used. In addiction, a strategy able to guarantee the existence of a feasible problem at each time step and consequently guarantee obstacle avoidance is reported. The current research aim to extend this approach with a 3D scenario, including uncertainty and model mismatch quantification, or to implement this approach on a real system. The other possibility is to distribute the problem between the different vehicles and include dynamic obstacles with a proper algorithm of obstacle avoidance.

Bibliography

[1] M. Bolognini and L. Fagiano. Lidar-based navigation of tethered drone formations in an unknown environment. *IFAC-PapersOnLine*, 53(2):9426–9431, 2020.

[2] M. N. Boukoberine, Z. Zhou, and M. Benbouzid. Power supply architectures for drones-a review. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 5826–5831. IEEE, 2019.

[3] M. Caruso, P. Gallina, and S. Seriani. On the modelling of tethered mobile robots as redundant manipulators. *Robotics*, 10(2):81, 2021.

[4] E. Casella, A. Collin, D. Harris, S. Ferse, S. Bejarano, V. Parravicini, J. L. Hench, and A. Rovere. Mapping coral reefs using consumer-grade drones and structure from motion photogrammetry techniques. *Coral Reefs*, 36(1):269–275, 2017.

[5] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su. Uav path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 47(6):1407–1420, 2016.

[6] I. I. Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[7] L. Fagiano. Systems of tethered multicopters: modeling and control design. *IFAC-PapersOnLine*, 50(1):4610–4615, 2017.

[8] S. Formentin and M. Lovera. Flatness-based control of a quadrotor helicopter via feedforward linearization. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6171–6176. IEEE, 2011.

[9] J. Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

[10] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen. Autonomous uav navigation using reinforcement learning. *arXiv preprint arXiv:1801.05086*, 2018.

[11] D. Saccani and L. Fagiano. Autonomous uav navigation in an unknown environment via multi-trajectory model predictive control. In *2021 European Control Conference (ECC)*, pages 1577–1582. IEEE, 2021.

[12] M. Tognon. Attitude and tension control of a tethered formation of aerial vehicles. 2014.

[13] J. Unger, M. Reich, and C. Heipke. Uav-based photogrammetry: monitoring of a building zone. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 40 (2014)*, 40(5):601–606, 2014.

[14] C. Zammit and E.-J. Van Kampen. Comparison between a* and rrt algorithms for uav path planning. In *2018 AIAA guidance, navigation, and control conference*, page 1846, 2018.

List of Figures

List of Tables

List of Symbols

Variable	Description
N_d	Number of drones
ϕ, ϑ, ψ	Roll, Pitch and Yaw angles
$\boldsymbol{p}^g, \dot{\boldsymbol{p}}^g$	Position and Velocity vector of the drones
$\overline{m}_d$	Mass of the vehicle
m_w	Weight of the winch and stored cable
m_t	Mass of the extended cable
b, d	Lift and Drag coefficients
Ω	Rotational speed of the rotor
L_f	Lift force
T	Drag force
a	Distance rotor-centre of mass of the drone
p, q, r	Angular velocities of drone (local coordinates)
R^T	Rotation matrix (local-global coordinates)
$\boldsymbol{F}$	Vector of forces (global reference frame)
g	Gravity acceleration
I_x, I_y, I_z	Angular mass of the drone
J_p	Moment of inertia of the propeller
d_w	Distance between drone and winch
$\theta, \dot{\theta}, \ddot{\theta}$	Position, Velocity, Acceleration of the winch
r_e	External radius of the winch
$\overline{m}_w$	Mass of the winch without tether wounded
ρ_t	Unitary mass of the tether per length
l_t	Overall length of tether
J_w	Moment of inertia of the winch
r_{i}	Internal radius of the winch

Variable	Description
B_w	Viscous friction coefficient
u_w	Winch torque
e_t	Elongation of the tether
K_t	Stiffness of the tether
$\boldsymbol{F}_t$	Vector of forces of the tether
$\boldsymbol{P}_{ref}$	Vector of references
$\boldsymbol{x}$	States vector
$\boldsymbol{u}$	Inputs vector
$\boldsymbol{P}$	Vector of the position of the drone
$\boldsymbol{V}$	Vector of the velocity of the drone
$\boldsymbol{A}$	Vector of acceleration of the drone
k_{vel}, k_{pos}	Gains of the model of the drone
T_s	Sampling time
A_{dt}, B_{dt}, C_{dt}	Matrices of the full system in discrete-time
P_{xy}	Selection matrix
$\boldsymbol{\gamma}$	Vector of measurements of LiDAR
k	Discrete-time variable
N_r	Number of measurements of LiDAR sensor
$\boldsymbol{\varphi}$	Vector of directions of LiDAR sensor
H	Shape matrix of obstacles
O	Set of the obstacles
χ_c	Center coordinates of the obstacle
N_O	Number of obstacles
φ_M	Horizontal angle
$\boldsymbol{C}$	Vector of configuration of the system
$\boldsymbol{S}_{free}$	Free space
$\bar{l}, \underline{l}$	Maximum, Minimum tether length
$\boldsymbol{P}_{target}$	Vector of the target
$\boldsymbol{C}^*$	Vector of optimal configuration of the system
X, Y	Equation of rotated ellipse
x_0, y_0	Center coordinates of ellipse
α	rotation angle of the ellipse

Variable	Description
δ	Distance between drones position and ellipses
a_e, b_e	Semi-major,semi-minor axis of ellipse
σ	Distance between obstacle and tether
d_{min}	Distance between tether and ellipse
d_c	Distance between ellipse center and tether
d_e	Distance between discretized point and ellipse
N_{pe}	Number of discretized points
N_e	Number of constraints with perpendicular method
L	Matrix containing LiDAR readings
L_d	Overlapping measurements
$L_{i,i+1}$	Merged LiDAR readings
k_1, k_2	Optimization variables optimal ellipse problem
N_p	Number of discretized points of the ellipse
D	Convex under-approximation of free space
ε	Variable used in polytope expansion
N_d	Number of convex polytope
$d_{emergengy}$	Distance used in polytope reduction
N	Prediction horizon
Q, T_u	Weighting matrix used in FHOCP
J	Cost function of FHOCP
τ	Distance used in path following approach
μ	Coefficient used in path following approach
Ψ	Angle between tether and x axis
$\boldsymbol{t}$	Vector used in path following to assign drones goal
γ	Matrix defining the points of the path
$\bar{t}$	Variable used in path following to assign drones goal
υ	Increment used in path following
$L_{d_{i,i+1}}$	Matrix of LiDAR readings used in LP
λ	Perturbation variable used in obstacle check function
$\overline{A}, \overline{V}$	Maximum acceleration and velocity of the drone
$\underline{d}$	Minimum distance between drones
N_F	Number of free drones

Variable	Description
G	Set containing position and target of drones
T	Triangle used in triangle method
$\boldsymbol{tr}$	Vector used for triangle analysis
ϵ	Variable used in backtrack strategy
N_T	Number of triangles
$\boldsymbol{\iota}$	Vector of indices used in path following

Ringraziamenti

Ringrazio il prof. Lorenzo Mario Fagiano, per avermi dato l'opportunità di svolgere la tesi in un ambito così stimolante quale l'ottimizzazione numerica e i sistemi di droni interconnessi.
Ringrazio i miei correlatori Danilo Saccani e Michele Bolognini, per avermi aiutato e supportato durante questi mesi e per essere stati sempre disponibili e gentili nei miei confronti. Vi auguro il meglio per il vostro futuro professionale.
Ringrazio la mia famiglia, che mi è sempre stata saldamente vicina durante il percorso universitario e mi ha aiutato a superare i momenti di difficoltà.
Ringrazio infine i miei amici e la mia ragazza Valentina, i quali mi hanno spronato e mi hanno concesso attimi di relax e di svago. Grazie a loro ho affrontato questo percorso con più leggerezza e tranquillità.