Distributed Asynchronous Algorithm for Collaborative Multi-UAV Multi-Target Tracking

Nuno Pereira, Andrea Simonetto, and Coen de Visser

Abstract—The use of mobile sensor networks for multi-target tracking is an active research field. This paper describes the application of a novel two-layer relaxation for high-level control and coordination of UAVs to optimally track a set of moving targets considering a limited resources scenario. Using a single-best tracking criterion, the scheme deploys distributed algorithms constrained to ensure full target coverage at all times. Quadrotor and fixed-wing UAV dynamics are considered, using respectively a point-mass and a Dubins car dynamical model. Our novel two-layer relaxation divides the problem into a distributed assignment problem and many local single-agent tracking problems. The proposed distributed algorithm requires low bandwidth messages and solves the assignment problem with an asynchronous-unreliable communication protocol between only neighbour agents. The tracking problems are formulated as a SDP problem for a quadrotor, while for a non-holonomic fixed-wing dynamics, a sequential convex programming algorithm is added to solve the linearised SDP problem iteratively. The proposed two-layer algorithms allow the use of heterogeneous teams and emergency manoeuvres to deal with collision avoidance and agents’ refuelling are implemented. Analysis proves that our relaxation optimises the same functional as the initial centralised formulation at steady-state, thus they have the same global optimum in a static targets scenario. The proposed distributed algorithms have linear time complexity with the number of targets and their complexity does not depend on the number of agents. Results show that our algorithms perform reasonable well in a wide range of scenarios.

Keywords—Multi-target tracking, distributed robotics, multi-robot collaboration, semi-definite programming.

I. INTRODUCTION

TARGET tracking is an active research field with growing interest in recent years. Advances in miniaturization have made possible the development of small, light and cheap Unmanned Aerial Vehicles (UAVs). Their inexpensiveness, flexibility, highly scalability, and the very fact that they are unmanned make UAVs into highly desirable tracking and surveillance platforms. When used individually, UAVs have a relatively limited functionality. Groups of cooperating UAVs, also called “swarms”, potentially have a functionality that is greater than the sum of its parts. With a variety of military and civil applications such as surveillance, defence, security, reconnaissance, computer vision systems, monitoring, among others; performing target tracking with mobile sensor networks has become increasing significant. Comparing to the use of static sensors, the use of mobile sensors brings with it superior coverage and mobility. Also, their spatial distribution can change dynamically and adapt to changes in the environment or the targets’ movement. In contrast to large and expensive manned vehicles, small robots are ideal for target tracking applications, specially in hazardous environments.

However, creating an effective autonomous UAV swarm for surveillance and tracking is a challenging task. These robotic networks have limited on-board processing and communication resources, limiting the CPU time and the communication bandwidth. The problem of cooperative path planning of UAVs for target tracking is considered by implementing efficient communication protocols across the network. We propose to steer a group of UAVs to optimally track a larger set of dynamic targets, motivated by limited resources scenarios. Limited on-board processing and communication capabilities constrain the problem to a simple implementation without any central control unit, while guaranteeing full target coverage at all times. Due to the short endurance of most of these UAVs, refuelling strategies are added as an emergency manoeuvre together with a collision avoidance system. Also, non-holonomic vehicle dynamics are often needed to model fixed-wing UAVs bringing another challenge to the picture.

Centralised target-tracking is widely mentioned in the literature (single-target tracking [20],[19],[42],[40] and multi-target tracking [35],[1],[11],[39]). The problem of optimal trajectory generation for a team of heterogeneous robots moving in a plane and tracking a moving target by processing relative observations is proved to be Non-deterministic Polynomial-time hard (NP-hard) [42]. The performance criteria is defined in an A-optimal design fashion [37] using the Fisher Information Matrix (FIM) [40], and a relaxation is proposed to achieve linear complexity with the number of agents. The performance criterion is also defined using graph theory which seeks to maximise the visibility of the targets observation [11]. This problem is non-convex and sequentially linearised as a Semi-Definite Positive program (SDP), subjected to LMIs.

Distributed target-tracking is widely mentioned in the literature (single-target tracking [23],[41],[5],[24],[28] and multi-target tracking [29],[31],[30],[25],[16],[8],[36],[33]). The 2-
norm distance towards the targets is minimised in [3] to obtain the best image as possible spaced with desired fixed intervals. Assuming that sensing a target with one agent only is enough to track it, the problem is formalised as the maximisation of the time that each target is begin monitored by at least one sensor-agent in [29],[31]. Since this problem is proved to be NP-hard, a relaxation using force fields is considered. Minimising the average time duration between two consecutive observations for all targets is implemented in one of the few papers considering specifically a limited resources scenario [36], i.e., a number of agents lower than the number of targets $n \ll m$. The motion-planning is stated as a NP-hard optimisation-based problem solved with a gradient-based method, and a suboptimal approach is proposed to reduce largely the computational load. The Dubins car model is also used [36].

There are a number of other relevant works, related to field detection, formation control, area coverage and space partition. In field detection, the “targets” assume geometric forms instead of being mass-points, such as fire perimeters [7] or a distributed system along a spatial domain [37]. Formation control is another research topic relating UAVs and distributed control [18],[5]. The field of area coverage is motivated by environmental monitoring applications, where the goal is to cover a given area of interest [17]. Distributed implementations to solve coverage problems is mentioned in [6]. Ad hoc communication networks is a researching field that creates and maintains a network of mobile agents connected by wireless [34].

There is no standard formulation for these optimisation-based tracking problems among the literature. Different scenarios motivate different tracking goals formulated with different performance criteria. Regarding our formulation, the 2-norm distance of the agents towards the targets is a metric already used before [3]; and the assumption that sensing a target with one agent only is enough to track it was also considered [29],[31]. A limited resources scenario was considered before [36], however the authors assume that the agents move much faster than the targets: the agent-target assignment is fixed during the simulation. Regarding the agent models, the Dubins model describes a simple 2D constant speed agent with a bounded turning radius largely used as a kinematic model of fixed-wing UAVs [39],[36],[2]. Regarding the observation and estimation models, most of the mentioned literature [11],[36],[41],[24],[30],[28] assume perfect observation of the agents’ states. The position of the targets is also assumed to be known with precision [11],[30],[29],[31], motivating our perfect “estimation” within a given sensing range.

In this paper, we present a new distributed asynchronous algorithm for UAV target tracking that solves a general target tracking problem using local SDP solvers and using an asynchronous-unreliable communication between neighbour agents. We introduce a novel performance criterion assuming that sensing a target does not require more than one agent [29],[31], motivated by a limited resources scenario [36]. Our single best-estimation criterion optimises, for each target, the tracking performed by its closer agent. This approach extends all the work mentioned by considering the joint use of quadrotor and fixed-wing agents, allowing the implementation of our algorithm in heterogeneous swarms. This algorithm also extends the work of [36], by considering a more dynamic environment where the targets’ velocity is not neglectable along the optimisation. The main contribution is the novel two-layer approach to relax the centralised non-convex optimisation problem by solving distributed convex problems locally, decomposing it into an Assignment Problem (AP) and local Tracking Problems (TPs). We present a theorem proving that, under a static-targets scenario, the original and the relaxed two-layer problems optimise the same functional at steady-state, i.e., they have the same global optimum. Finally, we present our distributed asynchronous algorithm based on an asynchronous-unreliable communication between neighbour agents. The local problems are formulated as SDPs with linear complexity with the number of targets and no dependence with the number of agents.

This paper consists of nine sections organised as follows. Section I introduces the topic, provides an overview of the relevant literature and states the contributions of this work. Section II formulates our optimisation-based control problem and introduces the UAV models. Section III proposes a novel two-layer relaxation to solve the problem. Sections IV and V solve the two “layers”: the AP by implementing distributed communication schemes and the local quadrotor and fixed-wing TPs by solving SDPs. Section VI introduces the pseudo-code to implement on each UAV, proves exact convergence of our steady-state solution and introduces emergency manoeuvres (collision avoidance and agents refuelling). Section VII presents some simulation results and compares the different algorithms with respect to the agents performance. Section VIII concludes and mentions future research topics.

II. PROBLEM FORMULATION

Multi-UAV multi-target tracking can be seen as an active sensing problem [26]. Let the set of coordinates $(E, N, h)$ define a 3D convex space $S := [E, N, h]^\top \subset \mathbb{R}^3$. We define $(E, N)$ as the East and North plane coordinates respectively, and $h$ as the altitude coordinate with respect to a fixed ground-level (Fig. 1).

![Multi-UAV Multi-Target Tracking](image.png)

Let a group of $n$ UAVs have on-board sensors capable of tracking targets, represent our set of autonomous interacting
entities called agents. Each agent is defined by its unique number \( i = 1, \ldots, n \), and its state \( x_i \) which depends on the model of the agent \( i \) and includes its position \( x_{pos}^i = [E^i, N^i, h^i]^\top \in S \). The agents are modelled as discrete time dynamical systems

\[
x_{k+1}^i = \mathbf{f}^i(x_k^i, u_k^i, \eta_k^i), \quad i \in \{1, \ldots, n\},
\]

where \( \mathbf{f}^i \) is the non-linear model, \( x_i \) the state, \( u_i \) the control input, and \( \eta_i \) the system noise of the agent \( i \) at the time-step \( k \). We assume that on-board GPS/INS sensors are able to estimate the agents’ states perfectly. Two agent models are considered:

1) Quadrilater model: discrete point-mass dynamics

\[
\mathbf{f}^i_{\text{quad}}(x_k^i, u_k^i) = \begin{bmatrix} I_3 & 0_3 & I_3 \Delta t \\ 0_3 & 1 & 0_3 \\ I_3 & 0_3 & I_3 \Delta t \end{bmatrix} \begin{bmatrix} x_k^i \\ u_k^i \\ \eta_k^i \end{bmatrix},
\]

where \( x_i = [x_{pos}^i, x_{vel}^i]^\top \) is the state given by its position \( x_{pos}^i = [E^i, N^i, h^i]^\top \) and its velocity \( x_{vel}^i = [v_{E}^i, v_{N}^i, v_{h}^i]^\top \), and the input \( u_i = [f_{E}^i, f_{N}^i, f_{h}^i]^\top \) corresponds to force applied along the three axis. We constrain the quadrotores to a minimum flying altitude, a bounded velocity and force as follows.

\[
\begin{align*}
h^i &\geq h_{\text{min}} \\
-v_{\text{max}} &\leq v_{E}^i, v_{N}^i, v_{h}^i \leq v_{\text{max}}. \\
-f_{\text{max}} &\leq f_{E}^i, f_{N}^i, f_{h}^i \leq f_{\text{max}}.
\end{align*}
\]


\[
\mathbf{f}^i_{\text{wing}}(x_k^i, u_k^i) = \begin{bmatrix} E_{k}^i \\ N_{k}^i \\ \theta_{k}^i \end{bmatrix} = \begin{bmatrix} V_{M} \cos \theta_{k}^i \\ V_{M} \sin \theta_{k}^i \\ u_k^i \end{bmatrix} \Delta t,
\]

where \( x_i = [E^i, N^i, \theta]^\top \) is the state given by its 2D position \( (E^i, N^i) \) and orientation \( \theta \), and the scalar input \( u \) corresponds to the change of heading command. This model has a constant velocity \( V_{M} \) and a fixed-altitude \( h \) such that \( x_{pos}^i = [E^i, N^i, h]^\top \). We also constrain the fixed-wings with a maximum turning rate as follows.

\[
-u_{\text{max}} \leq u^i \leq u_{\text{max}}.
\]

Let a group of \( m \) ground vehicles represent our set of moving vehicles of interest called targets. Each target is defined by its unique number \( q = 1, \ldots, m \), and its ground position \( t_{pos}^q = [E^q, N^q, h^q = 0]^\top \in S \). The targets move according to a Random-walk discrete model

\[
t_{pos}^{q,k+1} = t_{pos}^{q,k} + t_{vel}^q \Delta t, \quad q \in \{1, \ldots, m\},
\]

where \( t_{pos}^{q,k} \) is the position, \( t_{vel}^{q,k} = [w_{E}^{q,k}, w_{N}^{q,k}, 0]^\top \) the velocity of the target \( q \) at the time-step \( k \), and \( \Delta t \) the discrete time-step; where \( w_{E}^{q,k}, w_{N}^{q,k} \) are zero mean Gaussian noises, bounded as

\[
||t_{vel}^{q,k}||^2 = (w_{E}^{q,k})^2 + (w_{N}^{q,k})^2 \leq (W_{\text{max}})^2, \quad q \in \{1, \ldots, m\}.
\]

Let the set of agents have sensors capable of tracking targets, modelled by

\[
z_k^q = h^i(x_k^i, t_{pos}^{q,k}, \zeta_k^q), \quad i \in \{1, \ldots, n\},
\]

where \( h^i \) is the sensing model of the agent \( i \); \( x_i^q \) the state of the agent \( i \) and \( t_{pos}^{q,k} \) is the position of the target \( q \); \( \zeta_k^q \) is the measurement vector, and \( \zeta_k^q \) the sensing noise of the agent \( i \) towards the target \( q \) at the time-step \( k \). We assume the following noisy sensor model within a limited sensing range \( R \)

\[
z_k^{i,q} = t_{pos}^{q,k} + \zeta_k^{i,q} ||x_i^q - t_{pos}^{q,k}||,
\]

where \( \zeta_k^{i,q} = [\zeta_k^{i,q}, \zeta_k^{i,q}, 0]^\top \), and \( \zeta_k^{i,q}, \zeta_k^{i,q} \) are zero mean Gaussian noises with standard deviation \( \xi \). However, our formulation optimises with respect to the sensed targets’ positions assuming \( z_k^{i,q} \approx t_{pos}^{q,k} \). However, the effect of having sensing noise with different boundaries is studied in Section VII.

Defining a sensing criterion, we introduce a performance criterion that measures how optimal a certain state and control law is. Typically in active sensing problems, the performance criterion is composed by two components, as follows [26].

\[
J_k(x_k^i, u_k^i) := \sum_j \beta ||U(x_k^j)|| + \sum_{l} \rho ||C^l(u_k^l)||,
\]

where \( U \) represents the \( j \) terms that measure the expected target uncertainty as a function of the states \( x_i \); and \( C \) represents the \( l \) terms that measure the expected utility-cost spent in tracking as a function of the inputs \( u_i \); the coefficients \( \beta \) and \( \rho \) weight the different impacts to achieve a scalar criterion \( J_k \).

In general, target tracking depends on many factors, however we define the tracking quality of the agent \( i \) towards the target \( q \) as the inverse of distance between them. Thus, to maximise the tracking quality, we minimise the distance agent-target \( ||x_{pos} - t_{pos}^{q}||^2 \). Motivated by real applications, we consider a limited-resources scenario where we have fewer agents than targets \( n \ll m \). Assuming that sensing a target does not require more than one agent to estimate the target’s position [29],[31], we formulate the problem using a single-best estimation Receding Horizon Controller (RHC) performance criterion, i.e., we want to minimise for each target the distance towards the closer agent

\[
J_k := \sum_{t=0}^{N-1} \sum_{q=1}^{m} \min_{i \in \{1, \ldots, n\}} \{ ||x_{pos}^{i,t+1} - t_{pos}^{q}||^2 \} +
\]

\[
+ \rho \sum_{i=1}^{n} \sum_{q=1}^{m} \min_{i \in \{1, \ldots, n\}} \{ ||x_{pos}^{i,t,N} - t_{pos}^{q}||^2 \},
\]

where \( N \) is the finite horizon, \( \rho \) the input control effort and \( \varphi \) the terminal horizon cost.

Constraining the problem to ensure full target coverage, we guarantee that each target has at least one agent capable of tracking it, i.e., each agent is within the sensing range of at least one agent \( ||x_{pos} - t_{pos}^{q}|| < R \). Using the Schur complement, we formulate the full target coverage constraint as a LMI constraint [32]

\[
\forall q \in \{1, \ldots, m\} \exists \sum_{i=1}^{n} \{ \mathbf{I} \}^\top \begin{bmatrix} (x_{pos} - t_{pos}^{q})^\top \cr (R^q)^2 \end{bmatrix} \geq 0.
\]
Given the single-best estimation criterion \( (10) \), the quadrotors \((2)-(3)\) or the fixed-wings \((4)-(5)\) dynamics and limitations, and the full target coverage constraint \((11)\); we define the problem of computing the control input \( \mathbf{u} := \{\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}\} \) of \( n \) agents to track \( m \) targets as

\[
\min_{\mathbf{u}} J_k(\mathbf{x}_k, \mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}) \quad \text{s.t.}
\begin{align*}
\text{quadrotors or fixed-wings dynamics and limitations,} \\
\text{full target coverage constraint.}
\end{align*}
\]

Typically, optimisation problems can be implemented with different schemes \([38]\). When the number of agents increase, solving the problem in a centralised way may become computationally intractable. Therefore, we implement a distributed optimisation where the problem is divided into \( n \) sub-problems to be solved in parallel by \( n \) decision-making agents without the need of a central unit. Also, cooperation among agents is implemented sharing as few information as possible between neighbour agents only.

III. A NOVEL TWO-LAYER RELAXATION

We first implement a centralised “naive” approach for both agent models based on a centralised optimisation scheme using the original non-convex performance criterion \((10)\). Non-convex problems are in general difficult to solve, a compromise between a very long computational time and finding the global optimum has to be made. We implemented the two problems with the MATLAB tool \texttt{fmincon}, which applies an interior point algorithm to solve non-convex problems. Regarding the time complexity with the number of agents, the MATLAB tool \texttt{fmincon} uses a Sequential Quadratic Programming (SQP) method. Due to its centralised formulation, the number of optimisation variables increases by \( 9N \) per quadrotor and by \( 4N \) per fixed-wing, being \( N \) the predictive horizon. Also, solving a non-convex problem is time consuming due to the need of using non-convex optimisation algorithms. Therefore, we conclude that this implementation is not implementable in real UAVs. In order to face the results obtained with this naive centralised implementation, we aim to relax the problem formulation to obtain a convex optimisation and distribute the problem over the agents. Our approach is decomposed into a distributed Assignment Problem (AP) and Local Tracking Problems (TPs).

![Fig. 2. Problem relaxation and distribution.](image)

In the current formulation, the point-wise minimum function solves the agent-target assignment in a centralised form coupled with the tracking problem. The positions of all the agents are compared in order to assign an agent to each target along the entire predictive horizon for every time-step. We relax the centralised formulation (Fig. 2), solving the agent-target assignment beforehand at each time-step, and steer the agents using a convex local tracking performance criterion.

In order to compute the AP beforehand, we will not compute the assignment along the predictive horizon using the positions \( \{\mathbf{x}_{k+t,pos}\}_{t=1}^N \), instead we will compute the assignment only once per time-step \( k \) using a single future agents positions \( \mathbf{x}_{F,pos} \) (Fig. 3). Using the variables \( \{\mathbf{x}_{F,pos}\}_{i=1}^n \), the global AP seeks to assign each target \( q \) to its closer agent:

\[
a^q = \arg \min_{i \in \{1, \ldots, n\}} \{||\mathbf{x}_{F,pos} - \mathbf{t}_{pos}^q||^2\}. \tag{12}
\]

![Fig. 3. Single future position computed beforehand.](image)

This AP relates to coverage problems \([13],[21]\) which seek to split a given area according to a certain performance criterion. Using coverage algorithms, the space is partitioned in \( n \) sectors, and each sector is assigned to one agent. Motivated by the fact that the local tracking problem steers the agents towards the targets’ centre-of-mass of each partition, we will define these future agents position \( \mathbf{x}_{F,pos} \) as the targets’ centre-of-mass. This definition guarantees that the steady-state solution of the original and the relaxed problems are indeed the same (Theorem 4).

To avoid a centralised entity to compute the target-assignment, we implement distributed partition algorithms to solve the AP based on communication between neighbour agents only \([6]\). Distributing the AP, we obtain our novel two-layer implementation (Fig. 2), described as follows.

1) Distributed Assignment Problem (AP) [Section IV]: implements a distributed scheme between neighbour agents which solves the agent-target assignment using the future agents position variable \( \mathbf{x}_{F,pos} \).

2) Local Tracking Problem (TP) [Section V]: implements \( n \) convex independent problems where each agent (quadrotor or fixed-wing) seeks to track optimally the set of targets inside its tracking region.

Regarding optimality and feasibility, we conclude that achieving the global optimum is not our first priority. Also, the global optimum is constantly changing due to the highly dynamic changing environment caused by the targets’ movement. Therefore, we are far more interested in obtaining fast and feasible algorithms.

IV. DISTRIBUTED ASSIGNMENT PROBLEM

The Assignment Problem assigns each target to one agent, such that each agent has a finite and defined number of target
to track in its local TP. The problem is formulated independently of the vehicle dynamics, allowing its implementation on heterogeneous UAVs swarms.

A. Global solution

As a first step, we will transform the AP into a future position’s partition problem which divides the ground-plane \( Q \) among the agents. An \( n \)-partition of \( Q \), denoted by \( (v^i)_{i=1}^n \), is an ordered collection of \( n \) closed subsets of \( Q \), with non-empty interiors, satisfying the properties:

(i) \( \bigcup_{i=1}^n v^i = Q \);
(ii) \( \forall i,j \in \{1,\ldots,n\}, i \neq j \) int \((v^i) \cap \text{int}(v^j) = \emptyset \).

Also, \( (v^i)_{i=1}^n \subseteq V_n \), where \( V_n \) denotes the set of \( n \)-partitions of \( Q \). Let \( i = \{1,\ldots,n\} \) denote the unique number of the set of agents, we attribute a one-to-one correspondence between the agents and the components of the \( n \)-partition \( (v^i)_{i=1}^n \). We refer to \( v^i \) as the tracking region of the agent \( i \), i.e., the task of each agent \( i \) is to track the targets placed inside the subset \( v^i \subseteq Q \). Among all the possible \( n \)-partitions \( (v^i)_{i=1}^n \subseteq V_n \) of \( Q \), we define the Voronoi \( n \)-partition \( (v^i(y))_{i=1}^n \) of \( Q \) by the set of points \( (y^i)_{i=1}^n \) as follows.

\[
v^i(y) := \{ g \in Q | \forall j \in \{1,\ldots,n\}, j \neq i \| g - y^i \| \leq \| g - y^j \| \}, \tag{13}
\]

where \( y^i \) represents the set of points of the compact set \( v^i \subseteq Q \) by the unique minimum

\[
C(v^i) := \arg \min_{y^i} \left\{ \int_{v^i} f(||y - g||)\phi(g)dg \right\}, \tag{14}
\]

where \( y^i \) is any point in \( v^i \), \( \phi(g) : Q \rightarrow R^+ \) a bounded integrable positive density function, and \( f \) a performance Lipschitz function \( f : R^+ \rightarrow R^+ \). Defining the performance function as the 2-norm Euclidean distance \( f(g) = \|g\|^2 \), the centroid is called the centre-of-mass [6] and defining the target density function \( \phi(g) \)

\[
\phi(g) = \sum_{q=1}^m \delta(g - t^q_{\text{pos}}), \tag{15}
\]

where \( \delta \) represents the Dirac function. The centre-of-mass of \( v^i \subseteq Q \) avoids the numerical computation of integrations in (14), becoming as follows.

\[
C(v^i) = \frac{1}{m^i} \sum_{t^q_{\text{pos}} \in v^i} t^q_{\text{pos}}, \tag{16}
\]

where \( m^i \) is the number of elements of the set \( \{t^q_{\text{pos}}\}_{q=1}^m \subseteq v^i \), i.e., the number of targets inside the tracking region \( v^i \).

We propose a so-called “centring and partitioning” Lloyd algorithm [21] to compute the future positions \( X_{\text{F, pos}} \). The Lloyd’s “centring and partitioning” algorithm chooses the partition \( v \) and the centre-points \( y \) that minimise the following energy functional

\[
\mathcal{H}(v, y) := \sum_{i=1}^n \int_{v^i} f(||y^i - g||)\phi(g)dg, \tag{17}
\]

which using the density function (15) and the 2-norm performance function becomes

\[
\mathcal{H}(v, y) = \sum_{i=1}^n \left( \sum_{t^q_{\text{pos}} \in v^i} ||y^i - t^q_{\text{pos}}||^2 \right). \tag{18}
\]

The Lloyd algorithm motivated many research efforts in later years. Despite the successful application in a large number of applications, the global convergence of this algorithm is proved only by a very few known conditions [13]. Note that the “centring and partitioning” is an alternating variable algorithm, i.e., we do not minimise for the optimisation variables \( (v, y) \) together, we first optimise for \( y \) and then for \( v \).

To study the global convergence of the Lloyd algorithm, we first define the Lloyd map [12] by \( M = C \circ V \), where \( C \) denotes the map that matches the partition \( (v^i(y))_{i=1}^n \) to the mass centroids \( (C(v^i) y^i)_{i=1}^n \) of \( Q \). The Voronoi partition \( (v^i(y))_{i=1}^n \) of the set of points \( \{y^i\}_{i=1}^n \). Then we recall the following results.

**Proposition 1** (Fixed point of the Lloyd Map [13]). Any limit point \( y = (y^i)_{i=1}^n \) of the Lloyd algorithm is a fixed point of the Lloyd map \( M \) and a critical point of the energy functional \( \mathcal{H} \).

**Theorem 1** (Unique fixed point [13]). If a given fixed point \( y = (y^i)_{i=1}^n \) is unique, then the Lloyd algorithm converges globally.

**Theorem 2** (Finite set of fixed points [13]). If the iterations in the Lloyd algorithm stay in a compact set, where the Lloyd map \( M \) is continuous, then the algorithm is globally convergent to a critical point of \( \mathcal{H} \).

**Proposition 2** (Energy functional \( \mathcal{H}(v, y) \) [6]). Considering any partition \( (v^i)_{i=1}^n \subseteq V_n \) and any set of points \( \{y^i\}_{i=1}^n \subseteq Q \), the following energy functional properties hold

\[
\begin{align*}
\mathcal{H}(\upsilon_{\text{Voronoi}}(y), y) &\leq \mathcal{H}(v, y), \\
\mathcal{H}(v, C(v)) &\leq \mathcal{H}(v, y),
\end{align*}
\]

where \( \upsilon_{\text{Voronoi}}(y) \) represents the Voronoi partition of \( Q \) by \( y \) and \( C(v) \) corresponds to the centre-of-mass of the subsets \( (v^i)_{i=1}^n \subseteq Q \).

Our Lloyd map has typically more than a single fixed point, as we will see in the simulation results. Therefore, only local optimality of the Lloyd algorithm is ensured. However, considering moving targets scenarios, converging to the global
optimum is not our first priority. Also, the so-called “centring and partition” algorithm, i.e., dividing the space using Voronoi partitions and centring the partitions in the centre-of-mass points, is monotonically non-increasing with the energy functional (18).

B. Distributed solutions

We distribute the proposed Lloyd algorithm into distributed AP units using the work done in [6], presenting a “centring and partitioning” algorithm relying on an asynchronous-unreliable communication between neighbour agents.

Introducing a Delaunay graph [10] with the node set \((y^1)_i=1^n\) and with edges in \((y^1, y^j)\) if and only if the agents \((i, j)\) are neighbours. By neighbours we consider a pair of agents \((i, j)\) with tracking regions \(v^i, v^j \in Q\) that satisfy

\[
\partial v^i \cap \partial v^j \neq \emptyset,
\]

and considering the \((y^1, y^j)\)-bisector Voronoi half-space of two distinct points \((y^1, y^j)\) given by

\[
B(y^1, y^j) := \{g \in \mathbb{Q} | ||g - y^1|| \leq ||g - y^j||\},
\]

we use the tracking regions update between two agents as follows

\[
v^i_{k+1} = (v^i_k \cup v^j_k) \cap B(C(v^i_k), C(v^j_k)), \tag{22a}
\]

\[
v^j_{k+1} = (v^j_k \cup v^i_k) \cap B(C(v^j_k), C(v^i_k)). \tag{22b}
\]

We define the asynchronous-distributed algorithm (Gossip [6]) based on direct-neighbours communication. The communication protocol can be asynchronous and failures may occur without compromising the performance (Fig. 4).

**Algorithm 1** Asynchronous-distributed coverage.

- for all \(k \in \mathbb{N}\), each agent \(i\) maintains in memory a tracking region \(v^i_k \subset Q\)
- given \((v^1_0, ..., v^n_0)\) an initial \(n\)-partition of \(Q\)

repeat

1. select a randomly a pair of neighbours \((i, j)\)
2. each agent computes its centroid \(C(v^i_k), C(v^j_k)\) (16)
3. agent \(i\) communicates \(C(v^i_k)\) to agent \(j\) and \(v^j_k\)
4. update tracking regions \(v^i_{k+1}, v^j_{k+1}\)

until the simulation is stopped.

To study the convergence of the asynchronous-distributed algorithm, we recall the following result [6]

**Theorem 3** (Convergence under persistent gossip communication [6]), Considering the Lloyd map which is a stochastic process. If \(M\) is a randomly persistent stochastic process, then the gossip coverage algorithm converges almost surely to the set of fixed points of the Lloyd map \(M\).

Using a random process to select the pair of communication neighbour agents, the asynchronous-distributed algorithm converges almost surely \(^2\) to a local minimum. Thus, the global optimality is not ensured. The asynchronous-distributed algorithm is implementable in an asynchronous, unreliable and delayed communication, with only a random pair of neighbour agents communicating per time-step. However, due to having only one communication per time-step, the asynchronous-distributed coverage algorithm converges slower than other distributed synchrony algorithms. Depending on the random movement, even with the same initial conditions, this algorithm can end up in different local optima [32].

V. LOCAL TRACKING PROBLEM

After assigning the set of targets to be tracked by each agent, we implement the local TPs. Starting with the quadrotor case, we define the problem as a SDP which deals with the optimisation of a linear cost function subjected to linear equality constraints and to Linear-Matrix Inequalities (LMIs). For the fixed-wing case, once the problem is non-convex due to the non-holonomic dynamics, we linearise it to obtain a SDP solved iteratively with a Sequential Convex Programming (SCP) algorithm.

The performance criterion (10) becomes quadratic (and convex) for the single agent case

\[
\mathcal{J}_k := \sum_{t=0}^{N-1} \sum_{q=1}^m ||x_{k+t+1, \text{pos}} - t_{k, \text{pos}}^q||^2 + \rho ||u_{k+t}||^2 + \varphi \sum_{q=1}^m ||x_{k+N, \text{pos}} - t_{k, \text{pos}}^q||^2 \equiv x^\top Q x + c^\top x,
\]

and it can be reformulated as with an epigraph form moving the quadratic terms to Linear-Matrix Inequality (LMI) constraints

\[
\min \mathcal{J}_k = c^\top x + \beta \quad \text{s.t.} \quad \begin{bmatrix} Q & X \\ X^\top & \beta \end{bmatrix} \succeq 0, \beta \geq 0.
\]

The full target coverage constraint is guaranteed considering the next possible targets’ positions \(t_{k+1, \text{pos}}^q\). We define the worst case scenario (Fig. 5) such that we guarantee to be feasible for any future positions if we fulfill the constraints for

\(^2\)Regarding the convergence of stochastic processes, “almost surely” convergence means that the process with converge with probability one.
A. Quadrotor SDP problem

Considering a Random-walk targets model, the worst-case scenario is derived [32] and formulated as follows.

\[ t_{k+1,\text{pos}}^q = t_{k,\text{pos}}^q + \frac{W_{\max}}{\|t_{k,\text{pos}}^q - x_{k,\text{pos}}\|}(t_{k,\text{pos}}^q - x_{k,\text{pos}}) \Delta t, \quad (23) \]

and using the full target coverage constraint (11) it is implemented in a LMI form.

A. Quadrotor SDP problem

Given the optimisation variables \( U = u_t, X = x_{t+1,\text{pos}} = [E_{t+1,1}, N_{t+1}]^T, \theta = [\theta_{t+1}] \) stacked as \( o = [u_t, x_{t+1,\text{pos}}, x_{t+1,\text{vel}}]^T, t = \{k, \ldots, k + N - 1\} \), we define the problem

\[
\min_o \bar{J}(o) = c^T X_{\text{pos}} + \beta \text{ s.t.} \quad \begin{align*}
A_{\text{eq}} o &= b_{\text{eq}}, A_{\text{ineq}} o \preceq b_{\text{ineq}}, \\
Q_{\text{X}}^{-1} X_{\text{pos}} &\geq \beta X, \\
X_{\text{pos}}^\top &\beta X \\
Q_{\text{U}}^{-1} U &\geq \beta U, \\
U^\top &\beta U, \\
M^q &\geq 0, \\
q &= \{1, \ldots, m\},
\end{align*}
\]

where

\[
\begin{align*}
\beta &= \beta_X + \beta_U, \\
M^q &= \begin{bmatrix}
I_3 \\
(x_{k+2,\text{pos}} - t_{k+1,\text{pos}}^q)^\top \\
R^2
\end{bmatrix}
\end{align*}
\]

such that the vehicle dynamics dependence is implicitly coded in the constraints by introducing the matrices \( A_{\text{eq}}, b_{\text{eq}}, A_{\text{ineq}}, b_{\text{ineq}}, A_{\text{eq2}}(\theta_0), b_{\text{eq2}}(\theta_0) \) and considering the agents’ states as optimisation variables. The derivation of the matrices \( c, Q_{\text{X}}, Q_{\text{U}}, A_{\text{eq}}, b_{\text{eq}}, A_{\text{ineq}}, b_{\text{ineq}}, A_{\text{eq2}}(\theta_0), b_{\text{eq2}}(\theta_0) \) can be found in [32]. The parameters of the problem are the predictive horizon \( N \), the performance weighting factors \( \rho, \varphi \), the quadrotors’ specifications \( h_{\min}, v_{\max}, f_{\max} \), the targets’ maximum velocity \( W_{\max} \), the sampling period \( \Delta t \) and the quadrotors’ maximum sensing range \( R \).

Note that this problem is a convex problem (defined with a convex cost function subjected to convex inequality constraints and affine equality constraints). The time complexity of this problem can be derived analytically, once SDP problems use interior-point methods, being given by \( O(\max\{\rho^3, \varphi^2\}) \) [27], where \( \rho = 9N \) represents the number of optimisation variables and \( \varphi = 24N + 8m + 2 \) the number of constraints. Therefore the time complexity of this problem depend mainly on the predictive horizon \( N \) but it also increases linearly with the number of targets \( O(m) \).

B. Fixed-wing SCP to solve linearised SDP problems

The fixed-wing dynamics can not be formulated with only linear equality and inequality constraints due to the non-holonomic dynamics. Feedback linearisation inversion [18], non-holonomic transformation [15] and Sequential Convex Programming (SCP) [4],[9] are techniques used to deal with non-linear constraints. We define a SCP where the linearised SDPs are solved iteratively. Using the first Taylor expansion of the trigonometric functions cosine and sine, we linearise the fixed-wing dynamics around an initial-headings vector \( \theta_0 \).

Given the optimisation variables \( U = u_t, X = x_{t+1,\text{vel}} = [E_{t+1,1}, N_{t+1}]^T, \theta = [\theta_{t+1}] \) stacked as \( o = [u_t, x_{t+1,\text{vel}}, x_{t+1,\text{vel}}]^T, t = \{k, \ldots, k + N - 1\} \), and using a fixed headings vector \( \theta_0 = [\theta_{0,k+1}, \ldots, \theta_{0,k+N-1}]^T \), we define the problem

\[
\min_o \bar{J}(o) = c^T X + \beta \text{ s.t.} \quad \begin{align*}
A_{\text{eq}} o &= b_{\text{eq}}, A_{\text{ineq}} o \preceq b_{\text{ineq}}, A_{\text{eq2}}(\theta_0) o &= b_{\text{eq2}}(\theta_0), \\
Q_{X}^{-1} X &\geq 0, \beta X \geq 0, \\
Q_{U}^{-1} U &\geq 0, \beta U \geq 0, \\
M^q &\geq 0, q = \{1, \ldots, m\},
\end{align*}
\]

where

\[
\beta = \beta_X + \beta_U, \\
M^q = \begin{bmatrix}
(I_3) \\
(x_{k+2,\text{vel}} - t_{k+1,\text{vel}}^q)^\top \\
R^2
\end{bmatrix}
\]

such that the vehicle dynamics dependence is implicitly coded in the constraints by introducing the matrices \( A_{\text{eq}}, b_{\text{eq}}, A_{\text{ineq}}, b_{\text{ineq}}, A_{\text{eq2}}(\theta_0), b_{\text{eq2}}(\theta_0) \) and considering the agents’ states as optimisation variables. The derivation of the matrices \( c, Q_{X}, Q_{U}, A_{\text{eq}}, b_{\text{eq}}, A_{\text{ineq}}, b_{\text{ineq}}, A_{\text{eq2}}(\theta_0), b_{\text{eq2}}(\theta_0) \) can be found in [32]. The parameters of the problem are the predictive horizon \( N \), the performance weighting factors \( \rho, \varphi \), the fixed-wings’ specifications \( V_{\max}, v_{\max}, h \), the targets’ maximum velocity \( W_{\max} \), the sampling period \( \Delta t \) and the fixed-wings’ maximum sensing range \( R \).

To guarantee a certain linearisation precision we bound the solution using a trust-region method around the initial point \( (\theta^*(\theta_0) - \theta_0) \in T \). Then, adding one extra constraint

\[
|\theta_{k+t}^*(\theta_0) - \theta_{0,k+t}| \leq \tau, \quad t = \{1, N - 1\}, \quad (26)
\]

we implement a trust-region SCP to solve the local fixed-wing Tracking Problem.

The performance obtained solving the non-linear non-convex problem and the linearised SCP is quite similar [32]. However our SDP formulation is convex and has a time complexity, based on the SDP, given by \( O(\max\{\rho^3, \varphi^2\}) \), where \( \rho = 4N \) is the number of optimisation variables, \( c = 15N + 8m + 2 \) the number of constraints, and \( i \) the number of SDP iterations. The dependence is linear with the number of maximum iterations \( O(i_{\text{max}}) \), and we achieve also for the
Algorithm 2 Sequential Convex Programming (SCP).
- for every time-step \( k \) initiate using the current heading \( \theta_{0,0} := 1_{(N-1)\times 1} \otimes \theta_k \)
- given a step-size \( \alpha \) and a trust-region \( \tau \)
- given a stopping criterion \( \epsilon, i_{\text{max}} \)
\[ \text{repeat} \]
1. Find the optimal \( \theta^*(\theta_{0,1}) \) solving the linearised SDP
2. Update \( \theta_{0,i+1} = (1-\alpha)\theta_{0,i} + \alpha\theta^*_i(\theta_{0,i}) \)
3. \( i = i + 1 \)
\[ \text{until} \]
one of the stopping criterion is satisfied
\[ i = i_{\text{max}} \lor ||\theta_{0,1} - \theta^*_{i-1}(\theta_{0,i-1})|| < \epsilon. \]

Algorithm 3 Asynchronous-distributed two-layer of agent \( i \).
\[ \text{repeat} \]
1. Input: \( \mathbf{x}_i^k, \{\mathbf{t}_q^k\}_{q=1}^m \in v_i^k \), neighbours \( j = \{1, ..., n_i\} \)
2. Lloyd process: jump with probability \( (1-p_{\text{gossip}}) \) to \( 7_a \), pick a random neighbour \( j \)
3. Compute centroid: \( C(v_i^k) \)
4. Communicate to neighbour: \( C(v_j^k) \)
5. Receive from neighbour: \( C(v_i^k) \)
6. Update tracking region: \( v_i^{k+1} \) using \( C(v_j^k) \)
7_a. Quadrotor TP: solve the SDP problem
7_b. Fixed-wing TP: run the SCP algorithm solving linearised SDP problems
8. Steer the agent: apply first control variable \( u_k \)
\[ \text{until} \]
the simulation is stopped.

fixed-wings’ case linear time complexity with the number of targets \( O(m) \).

VI. DISTRIBUTED TWO-LAYER ALGORITHM

We define the final agent asynchronous-distributed algorithm for our two-layer approach, equivalence between our novel two-layer approach and the centralised approach is proved under some conditions, and the implementation with heterogeneous teams and emergency manoeuvres is also discussed.

A local randomly persistent stochastic process is introduced, a so-called Lloyd process, to decide whether one agent should “gossip” with one neighbour. Each agent \( i \) communicates with probability \( p_{\text{gossip}} = 1/n \), and selects a random neighbour \( j \) in case it decides to communicate. This asynchronous-distributed algorithm can be implemented in an unreliable communication network; as long as this process is randomly persistent the convergence guarantees are fulfilled.

The distributed two-layer algorithm receives three inputs:
1) the current state of the agent \( \mathbf{x}_i^k \) obtained using a GPS/INS sensor,
2) the current position of all the targets inside the tracking region \( \{\mathbf{t}_q^k\}_{q=1}^m \) \( i \) within the agent’s sensing range \( R^2 \),
3) the set of neighbours \( j = \{1, ..., n_i\} \).

Fig. 6. Asynchronous-distributed two-layer algorithm of Agent \( i \).

This implementation allows the use of different UAV models working together. Therefore, using the two models introduced, the presented algorithms can be implemented in teams with both quadrotor and fixed-wing agents. Regarding the computational power on-board, it is required for each UAV to have a SDP solver. SDP can be solved relatively efficiently even with low capacity processors. Regarding the communication protocol, the synchronous algorithm needs a synchronous-reliable protocol between all neighbour agents every time-step, while the asynchronous algorithm is implementable in an asynchronous-unreliable protocol where only one pair of neighbour agents communicate in average at each time-step. The required bandwidth is only two coordinates \((E,N)\) per message. The time complexity is linear with number of targets \( O(m) \), and it does not depend on the number of agents \( n \).

Considering the set of \( n \) agents placed at \( \mathbf{x}_{0,\text{pos}} \), and the set of \( m \) targets placed at \( \mathbf{t}_{0,\text{pos}} \) inside the tracking-plane of interest \( Q \). Our distributed two-layer implementations have to be initialised with an initial Voronoi \( n \)-partition of \( Q \) by \( v_{0,\text{pos}} \), where each agent \( i \) is initialised with the tracking region \( v_i^0(\mathbf{x}_{0,\text{pos}}) \subset Q \).

A. Two-layer solution analysis

We have presented a two-layer relaxation for our centralised multi-target multi-UAV tracking problem. Considering one agent only, it is obvious that both formulations lead to exactly the same solution. Considering static targets, we define the steady-state solution as the final positions to which the quadrotors converge to. Note that for the fixed-wings’ case, there are not steady-state positions because the agents do not stop moving, instead they tend to move in circles around the steady-state solution.

Theorem 4 (Steady-state solution for static targets scenario). Under a static-targets scenario, and considering the original and the relaxed two-layer problems, both approaches optimise the same functional at steady-state, thus they have the same steady-state global optimum.

Proof: The centralised steady-state solution is given by the minimum
\[ \min_{\mathbf{x}} J_{ss} = \sum_{q=1}^m \left( \min_{i=1,...,n} \left\{ ||\mathbf{x}_i^{ss,\text{pos}} - \mathbf{t}_q^{\text{pos}}||^2 \right\} \right), \]
where \( \mathbf{x}_i^{ss,\text{pos}} \) represents the steady-state position of the agent \( i \) and \( \mathbf{t}_q^{\text{pos}} \) the position of the static target \( q \). Using the Voronoi \( n \)-partition \( v(\mathbf{x}_{ss,\text{pos}}) \), we reformulate the steady-state solution as
\[ \min_{\mathbf{x}} J_{ss} = \sum_{i=1}^n \left( \sum_{\mathbf{t}_q^{\text{pos}} \in v(\mathbf{x}_{ss,\text{pos}})} ||\mathbf{x}_i^{ss,\text{pos}} - \mathbf{t}_q^{\text{pos}}||^2 \right). \]
The two-layer relaxation steady-state positions are given by the local TP solutions

$$\min_{x_{ss,\text{pos}}} J_{ss} = \sum_{t_{ss,\text{pos}}^i \in v^i} \left( ||x_{ss,\text{pos}}^i - t_{ss,\text{pos}}^i||^2 \right).$$  (28)

where the steady-state partition \((v^i)_{i=1}^n\) is given by the minimum of the energy functional

$$\min_{v,y} H(v,y) = \sum_{i=1}^n \left( \sum_{t_{ss,\text{pos}}^i \in v^i} ||y^i - t_{ss,\text{pos}}^i||^2 \right).$$  (29)

which is obtained considering the local TP solutions as the centres \(y\) (28)

$$\min_{v,x} H(v,x_{ss,\text{pos}}) = \sum_{i=1}^n \left( \sum_{t_{ss,\text{pos}}^i \in v^i} ||x_{ss,\text{pos}}^i - t_{ss,\text{pos}}^i||^2 \right).$$  (30)

which is equivalent to the centralised solution (27).

Note that in general, we can not ensure that our relaxed two-layer algorithm finds the steady-state solution (global optimum) because the asynchronous-distributed coverage algorithm often stops in local minima.

Considering moving targets, there is no steady-state solution as the agents are constantly adapting to the targets’ positions. However, our relaxation only differs from the original formulation in the partition \((v^i)_{i=1}^n\)

1) original partition: every time step, \(N\) Voronoi \(n\)-partitions are computed centrally along the predictive horizon, dividing the set of targets by their closest agent considering every future agents’ positions \(x_{k+1,\text{pos}}\).

2) two-layer partition: every agent has its own tracking region updated by communication between neighbour agents. These partitions are fixed along the predictive horizon and tend to the minimum of the energy functional using a “centring and partition” algorithm with centres \(y\). Thus the slower the targets move, the closer the local TP solutions tend to these centres and more similar are both implementations.

B. Emergency manoeuvres

Two manoeuvres are added to our algorithm. When two agents \((i,j)\) are in danger of colliding \(||x^i - x^j|| < R_{CA}\), the collision avoidance algorithm starts. Note that this situation is unlikely to occur because each agent has its own tracking region and its position tends to be inside this region.

UAVs are typically flying agents with a short endurance and intelligence to return for refuelling has been considered [7]. The refuelling problem is basically dealing with a slow dynamic changing number of agents. When one agent needs to refuel, it leaves its mission and the other agents compensate.

Algorithm 4 Collision avoidance.

- choose the agent \(i\) with the highest altitude \(h_i \geq h_j\) (or choose randomly if they have the same altitude)

repeat

1. Guide the agents in the \(E,N\) plane normally using the two-layer algorithms
2. if \(h_i < h_j + R_{CA}\), then increase the altitude of the agent \(i\) towards \(h_j + R_{CA}\), else maintain the altitude \(h_i\) of the agent \(i\)
3. Maintain the altitude \(h_j\) of the agent \(j\)

until \(||x^i - x^j|| \geq R_{CA}\)

- for the fixed-wings’ case, bring the agent \(i\) back to the fixed altitude \(h_i = \hat{h}\)

Similarly, when one agent returns from refuelling, the agents rearrange themselves.

Algorithm 5 Agent departure.

- given a departure agent \(j\)
  - pick randomly one of the neighbours (agent \(i\)) of the departing agent \(j\)
  - rearrange the tracking region \(v^j_{k+1} = v^j_k \cup v^j\)
  - remove the agent \(j\) from the set of agents \(n_{k+1} = n_k - 1\).

Algorithm 6 Agent arrival.

- given an arrival agent \(j\)
  - pick randomly one agent (agent \(i\))
  - guide the new agent \(j\) to the tracking region \(v_i\)
  - split the tracking region into two
    \(v^j_{k+1} = v^j_k \cup B(C(x^j_k), C(x^j_k))\)
    \(v^j_k = v^j_k \cup B(C(x^j_k), C(x^j_k))\)
  - add agent \(j\) to the set of agents \(n_{k+1} = n_k + 1\).

VII. ALGORITHM RESULTS

The performance obtained with three target models are shown: nearly static targets, slow moving targets (slower than the agents) and fast moving targets (faster than the agents); respectively \(W_{\text{max}} = 0.01, 1, 3\) in Figures 7-8, 9-10 and 11-12. Both quadrotor and fixed-wing models are simulated separately (see agents path in Figures 13-14). The full parameters are presented in Table I and the computational time in Table II.

The original centralised non-convex formulation is simulated using the MATLAB toolbox \texttt{fmincon}. The distributed two-layer algorithm is implemented with a synchronous and asynchronous communication, and using the SDP solver SeDuMi from the MATLAB toolbox \texttt{YALMIP}[22].

Note that our two-layer approach distributes the problem among the agents and the computational time for computing each TP is spent locally by each agent. Regarding feasibility, the algorithm constraints are fulfilled during all the simulations for both the centralised and the distributed algorithms. The worst-case scenario guarantees feasibility for the local tracking problems as long as there is a feasible solution.
Table I. Final simulations parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td>finite horizon</td>
</tr>
<tr>
<td>n, m</td>
<td>10, 100</td>
<td>number of agents, targets</td>
</tr>
<tr>
<td>$\rho, \sigma$</td>
<td>0.01, 0.01</td>
<td>performance weighting factors</td>
</tr>
<tr>
<td>$t_{\text{min}}, t_{\text{max}}$, $f_{\text{MAX}}$</td>
<td>10, 2, 2</td>
<td>quadrotors’ specifications</td>
</tr>
<tr>
<td>$h, V_d, u_{\text{MAX}}$</td>
<td>10, 2, 0.8</td>
<td>fixed-wings’ specifications</td>
</tr>
<tr>
<td>$v_{\text{MAX}}$, $f_{\text{E}}, \alpha, \epsilon$</td>
<td>2, 0.7, 0.2, 0.01</td>
<td>different targets maximum velocities</td>
</tr>
<tr>
<td>$R$</td>
<td>40</td>
<td>agents’ sensing range</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>1</td>
<td>time-step (in seconds)</td>
</tr>
<tr>
<td>$k$</td>
<td>60</td>
<td>number of time-steps of the simulation</td>
</tr>
</tbody>
</table>

Fig. 7. Static targets: quadrotors.
Fig. 8. Static targets: fixed-wings.
Fig. 9. Slow targets: quadrotors.
Fig. 10. Slow targets: fixed-wings.
Fig. 11. Fast targets: quadrotors.
Fig. 12. Fast targets: fixed-wings.
Fig. 13. Slow targets quadrotors path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.
Fig. 14. Slow targets fixed-wings path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 10.

Regarding the steady-state solutions, the centralised algorithm finds the best solution while the two-layer relaxation finds a nearby optima. Regarding the agent models, the fixed-wings have worse results than the quadrotors as they tend to oscillate around the quadrotors steady-state solutions instead of converging (Figure 8). This is clearly caused by the constrained non-holonomic dynamics of these agents. Nevertheless they exhibit the same qualitative results regarding the different algorithms. Regarding the distributed algorithm, it is able to optimise with an asynchronous, unreliable, and delayed communication [6] without significantly compromise its performance.

For the moving targets’ case, the performance obtained using our novel two-layer approach is quite similar to the centralised solution. The asynchronous-distributed algorithm converges slower which affects its performance more than in the static targets’ case. However, after some iterations the performance is less compromised and the asynchronous-distributed algorithm obtains very similar performance to the synchronous-distributed one.

Regarding the computational time, the centralised problem takes minutes to compute the solution for each time-step, while using our two-layer relaxation the TP optimisation is solved within a second for the quadrotor and the fixed-wing case, using a scenario with $n = 10$ agents, $m = 100$ targets and $N = 3$ predictive horizon. Thus, for real implementations, we need $n$ machines, one for each of the $n$ agents, which will take around 0.8-0.95 seconds to compute the path planning for the respective agent at each time step. The only communication required is sending/receiving two coordinates $C_E(v^t), C_N(v^t)$ between neighbour agents.

Considering our two-layer asynchronous-distributed algorithm for a slow targets scenario ($W_{\text{MAX}} = 1$) using quadrotors, we consider the noisy sensing model (8) and we also consider imperfect actuators with the steering model

$$u_k^i = u_k^i + \sigma_k^i,$$

where $\sigma_k^i = [\sigma_k^i, E, \sigma_k^i, N, \sigma_k^i, h]^T$ and $\sigma_k^i, E, \sigma_k^i, N, \sigma_k^i, h$ are zero mean Gaussian noises with standard deviation $\mu$. The effect of different sensing and actuating noises can be seen, respectively, in Figures 15 and 16.

Feasibility is accomplished during the entire simulation for the different observing noises, but not in the presence of actuating noise. The qualitative results are maintained with observing noise and the performance decreases as expected.

Table II. Simulation time in seconds per time-step.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadrotors</td>
<td>146 seconds</td>
<td>0.82 seconds</td>
<td>0.79 seconds</td>
</tr>
<tr>
<td>Fixed-wings</td>
<td>94 seconds</td>
<td>0.94 seconds</td>
<td>0.93 seconds</td>
</tr>
</tbody>
</table>

Regarding the steady-state solutions, the centralised algorithm finds the best solution while the two-layer relaxation finds a nearby optima. Regarding the agent models, the fixed-wings have worse results than the quadrotors as they tend to oscillate around the quadrotors steady-state solutions instead of converging (Figure 8). This is clearly caused by the constrained non-holonomic dynamics of these agents. Nevertheless they exhibit the same qualitative results regarding the different algorithms. Regarding the distributed algorithm, it is able to optimise with an asynchronous, unreliable, and delayed communication [6] without significantly compromise its performance.

For the moving targets’ case, the performance obtained using our novel two-layer approach is quite similar to the centralised solution. The asynchronous-distributed algorithm converges slower which affects its performance more than in the static targets’ case. However, after some iterations the performance is less compromised and the asynchronous-distributed algorithm obtains very similar performance to the synchronous-distributed one.

Regarding the computational time, the centralised problem takes minutes to compute the solution for each time-step, while using our two-layer relaxation the TP optimisation is solved within a second for the quadrotor and the fixed-wing case, using a scenario with $n = 10$ agents, $m = 100$ targets and $N = 3$ predictive horizon. Thus, for real implementations, we need $n$ machines, one for each of the $n$ agents, which will take around 0.8-0.95 seconds to compute the path planning for the respective agent at each time step. The only communication required is sending/receiving two coordinates $C_E(v^t), C_N(v^t)$ between neighbour agents.

Considering our two-layer asynchronous-distributed algorithm for a slow targets scenario ($W_{\text{MAX}} = 1$) using quadrotors, we consider the noisy sensing model (8) and we also consider imperfect actuators with the steering model

$$u_k^i = u_k^i + \sigma_k^i,$$

where $\sigma_k^i = [\sigma_k^i, E, \sigma_k^i, N, \sigma_k^i, h]^T$ and $\sigma_k^i, E, \sigma_k^i, N, \sigma_k^i, h$ are zero mean Gaussian noises with standard deviation $\mu$. The effect of different sensing and actuating noises can be seen, respectively, in Figures 15 and 16.

Feasibility is accomplished during the entire simulation for the different observing noises, but not in the presence of actuating noise. The qualitative results are maintained with observing noise and the performance decreases as expected.
with the noise increase. However, for actuating noise, even with small standard deviation error the performance decrease is not neglectable. The proposed algorithm needs precise actuators while it is more robust to sensing noise.

Regarding the emergency manoeuvres, we simulate the agents’ collision avoidance algorithm using fixed-wings (Figure 17) and the agents’ refuelling algorithm using quadrotors (Figure 18).

To test the agents’ collision avoidance algorithm, once the two agents stabilise around the centre of their tracking regions, we switch their tracking regions. Therefore, the first agent goes towards the second agent position and vice-versa. The collision is avoided by agent 1 which increases its altitude and after the desired separation is replaced, the agent 1 returns to its fixed-altitude $h$. The agents’ refuelling performance decreases largely when one of the three agents departs to refuel. Then, the remaining two agents reorganise themselves, lowering the cost-function $J$. When the agent returns from refuelling, another transitory phase brings the performance criterion back to the initial performance achieved before the refuelling manoeuvre at $k \approx 20 - 40$.

VIII. CONCLUSION AND FUTURE WORK

In this work we address the topic of distributed optimisation for collaborative multi-UAV multi-target tracking. The desired UAV behaviour is stated based on limited resources scenario, where the number of agents is significantly lower than the number of targets, and formulated by a single-best estimation criterion.

A novel two-layer relaxation is introduced to distribute and devise our original centralised problem into a Assignment Problem and Tracking Problems to be solved locally. First, our distributed-approach is based only in a local asynchronous-unreliable communication protocol between neighbour agents. Second, the tracking problem is convexified bringing convergence guarantees where persistent feasibility is ensured using a worst-case scenario prediction. Also, the local TPs can be solved with SDP solvers, allowing a faster implementation with linear complexity with the number of targets and no dependence on the number of agents. Third, splitting the centralised problem into an AP-TPs formulation allows the completely separation of the assignment and tracking tasks. Thus, the problems can be improved/tested separately, and the assignment task can even run with a slower rate.

Finally, the solutions of both approaches are theoretically studied and proved to be highly related. The steady-state solution of our relaxation is proved to be exact, i.e., optimal with respect with the original formulation. Also, two emergency manoeuvres are added to our algorithm, in order to deal with collision avoidance between the agents and with agents’ refuelling.

This work opens and suggests some challenges for future research. An implementation of our pseudo-code on real UAVs is a must. Another research field arises considering scenarios with obstacles which constrain the line-of-sight from the agents towards the targets. Considering the non-holonomic fixed-wings dynamics, the convexification/linearisation of the problem could be done with different techniques that are worth of deeper study. Also using fixed-wings, the assignment problem can be adapted to these vehicle dynamics, for instance considering higher weights on the targets “in-front” of the agent when computing the Voronoi centroids might improve the overall fixed-wings performance. Heterogeneous teams are considered in this work, simulations using different agent model combinations can be simulated to determine which UAV combinations bring better for each scenario. Finally, in most of the real applications, the targets are vehicles with a certain dynamics. Target model estimation techniques might be added to the algorithm, noticing that this information would need to be transmitted between neighbour agents.

REFERENCES


\[6\] F. Bullo, R. Carli, and P. Frasca. Gossip coverage control for robotic