TÉCNICO LISBOA

Distributed Two-Layer Algorithms for Collaborative Multi-UAV Multi-Target Tracking

Nuno Miguel Braga de Carvalho Vieira Pereira

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Examination Committee

Chairperson: Prof. João Miranda Lemos

Supervisor: Prof. Pedro Lima

Member of the Committee: Prof. João Xavier

October 2013

Acknowledgements

First and foremost, I would like to thank my supervisors dr.ir. Coen de Visser, dr.ir. Andrea Simonetto and dr.ir. Pedro Lima, for all the knowledge, feedback and support during these six months. They have motivated me and made possible to finish within the planned time. I would also like to offer special thanks to dr.ir. Andrea Simonetto for his patience teaching me how to write properly, for his time giving me constantly feedback and, most of all, for his guidance through my thoughts and ideas.

I would like to express my great appreciation to my housemates Konstantinos Dimitropoulos, Marta Bariain, Maaike Sybrandy and Claudia Poma, for being friends and family before housemates. They have made me rediscover a life in Delft.

I would also like to thank to all my friends, that in Delft, in Portugal, or elsewhere, gave me their tireless support along the way.

Additionally, I would like to thank to all my basketball teammates for helping me releasing all the stress accumulated, and my beach volleyball partners for giving me a summer blast in the Netherlands.

Finally, I would like to thank to my family, for understanding why my summer holidays and my time spent with them was so short. You mean the world to me.

Delft, 10th September 2013,
Nuno Pereira

Abstract

The use of mobile sensor networks for multi-target tracking is an active research field. This thesis 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.

Resumo

O uso de redes de sensores móveis para seguimento de múltiplos alvos é uma área de investigação actual. Esta tese introduz um novo método para o controlo e coordenação de UAVs, baseado na divisão do problema original em duas partes distintas. O objectivo é optimizar o seguimento de um conjunto de alvos móveis considerando um cenário de recursos limitados. Para cada alvo, este método optimiza o seu seguimento considerando apenas o agente mais próximo. Algoritmos distribuídos são implementados, assegurando que todos os alvos estão a ser seguidos durante toda a simulação. Quadrotores e UAV de asa-fixa são considerados usando, respectivamente, dinâmicas de massa-pontual e o modelo de carro *Dubins*. Este método divide o problema original num problema distribuído de atribuição alvos-agentes e em vários problemas locais de seguimento de alvos usando um só agente. O algoritmo distribuído proposto requer pouca largura de banda e resolve o problema de atribuição mesmo com um protocolo de comunicação assíncrono e não-fiável entre agentes vizinhos. Os problemas de seguimento locais para os quadrotores são formulados como SDP. Enquanto que para a dinâmica não-holonómica dos agentes de asa-fixa, um algoritmo de programação sequencial convexa é introduzido para resolver linearizações do problema original sob a forma de SDP. Os algoritmos propostos permitem o uso de equipas heterogéneas e manobras de emergência que evitam colisões e permitem o reabastecimento dos agentes. O método proposto optimiza a mesma função de custo que o problema original em regime estacionário, assim a aproximação deste método tem o mesmo mínimo global que o problema original se considerarmos alvos estáticos. Os algoritmos distribuídos propostos têm linear complexidade temporal com o número de alvos, e a sua complexidade não depende do número de agentes. Resultados mostram que os algoritmos introduzidos comportam-se relativamente bem em cenários distintos.

Palavras-chave

Seguimento de múltiplos alvos, robótica distribuída, colaboração entre robôs, programação semi-definida (SDP).

Contents

List of Figures

List of Tables

List of Acronyms

AP Assignment Problem

CPU Central Processing Unit

DOF Degrees-Of-Freedom

EKF Extended Kalman Filter

FIM Fisher Information Matrix

GPS General Positioning System

IMAV International Micro Air Vehicle Conference and Flight Competition

INS Inertial Navigation System

LASE Low Altitude Short Endurance

LMI Linear-Matrix Inequality

LPV Linear Parameter-Varying

MAV Micro Air Vehicle

MIMO Multi-Input Multi-Output

MPC Model Predictive Controller

NP-hard Non-deterministic Polynomial-time hard

RHC Receding Horizon Controller

RSS Received Signal Strength

SCP Sequential Convex Programming

SDP Semi-Definite Positive program

SISO Single-Input Single-Output

SOCP Second-Order Cone Program

SQP Sequential Quadratic Programming

TP Tracking Problem

UAV Unmanned Aerial Vehicle

List of Symbols

S	3D region of interest
N	North coordinate
E	East coordinate
h	Altitude
Q	Ground-plane of S
$\mathbf{g}$	Ground-plane coordinate of Q
n	Number of agents
$\mathbf{f}$	Agents' models
$\mathbf{f}_{\text{quad}}$	Quadrotor's model
$\mathbf{f}_{\text{wing}}$	Fixed-wing's model
$\mathbf{f}^i$	Agent i's model
$\mathbf{x}$	Agents' states
$\mathbf{x}^i$	Agent i's state
$\mathbf{x}_{\text{pos}}$	Agents' positions
$\mathbf{x}^i_{\text{pos}}$	Agent i's position
$\mathbf{x}^i_{\text{vel}}$	Quadrotor i's velocity
v^i_E, v^i_N, v^i_h	Quadrotor i's velocities
θ^i	Fixed-wing i's attitude
$\mathbf{u}$	Agents' control input
$\mathbf{u}^i$	Agent i's control input
$\Delta\mathbf{u}^i$	Agent i's control input change
f^i_E, f^i_N, f^i_h	Quadrotor i's control inputs
u^i	Fixed-wing i's control input
$\boldsymbol{\eta}$	Agents' model noise
$\boldsymbol{\eta}^i$	Agent i's model noise
m	Number of targets
$\mathbf{t}_{\text{pos}}$	Targets' positions
$\mathbf{t}^q_{\text{pos}}$	Target q's position
$\mathbf{t}_{\text{vel}}$	Targets' velocities
$\mathbf{t}^q_{\text{vel}}$	Target q's velocity
$\mathbf{h}$	Agents' sensing models
$\mathbf{h}^i$	Agent i's sensing model
$\boldsymbol{\zeta}$	Agents' sensing noise
$\boldsymbol{\zeta}^{i,q}$	Agent i's sensing noise towards the target q
$\mathbf{z}^{i,q}$	Estimation of the position of the target q by the agent i
J	Single-step performance criterion
$\widehat{\mathbf{x}}$	Agents' states prediction

$\widehat{\mathbf{x}}^i$	Agent i's state prediction
$\bar{J}$	Finite-horizon performance criterion
N	Fixed predictive-horizon
φ	Terminal cost
$\mathbf{c}$	Set of physical constraints
$\mathbf{c}_{\min}$	Physical constraints lower bound
$\mathbf{c}_{\max}$	Physical constraints upper bound
Δt	Time-step
$\mathbf{I}_p$	Identity matrix of size $p \times p$
$\mathbf{1}_{p\times r}$	Ones matrix of size $p \times r$
$\mathbf{0}_{p\times r}$	Null matrix of size $p \times r$
$Q^{i,q}$	Tracking quality of agent i towards target q
R^i	Sensing maximum range of agent i
$\mathcal{U}$	Expected target uncertainty
$\mathcal{C}$	Expected utility
β	Target uncertainty cost
ρ	Input/Utility cost
$\mathbf{\Sigma}$	Sensing noise covariance
$p(\mathbf{z})$	Estimation conditional probability function
$H_{\mathbf{z}}(p, \mathbf{x}, \mathbf{t})$	Entropy (Shannon information)
$\mathbf{o}$	Optimisation variables
a^q	Agent assigned to target q
$\mathcal{V}_n$	Set of n-partitions of Q
v^i	Tracking region of agent i
$\mathbf{y}^i$	Centre-point of the tracking region v^i
w^i	Power factor of agent i
$B(\mathbf{y}^i, \mathbf{y}^j)$	$(\mathbf{y}^i, \mathbf{y}^j)$-bisector half-space
$C(v^i)$	Generalized centroid of the subset v^i
$\phi(\mathbf{g})$	Density function
f	Performance Lipschitz function to compute the generalised centroid
$\mathbf{F}(\mathbf{x}, \mathbf{t})$	Fisher Information Matrix (FIM)
$\mathcal{H}(v, \mathbf{y})$	Energy functional
M	Lloyd Map
$\mathbf{X}$	Set of states of the agent along the predictive horizon N
$\mathbf{U}$	Set of inputs given to the agent along the predictive horizon N
$\boldsymbol{\theta}$	Set of attitudes of the agent along the predictive horizon N
$\boldsymbol{\theta_0}$	Fixed headings vector
α	Step-size of the SCP
τ	Trust-region of the SCP
ϵ	Stopping criterion of the SCP
p_{gossip}	Probability of communicating in the asynchronous-distributed scheme

Chapter 1

Introduction

Target tracking is an active research field with growing interest in the recent years. With a variety range 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 been increasing significantly.

Contrary to the use of static sensors, the use of mobile sensors has superior coverage and mobility. They can cover larger areas without increasing their number. Also, their spatial distribution can change dynamically and adapt to changes in the environment or the targets' movement. While manned vehicles are large and expensive, the use of multiple small inexpensive robots is ideal for target tracking applications. Suitable also for hazardous environments, the use of robotic networks provides successful autonomous surveillance systems.

Unmanned Aerial Vehicles (UAVs) play an increasingly prominent role in these applications. Motivated by their manoeuvrability, the problem of UAVs cooperative path planning for target tracking is largely mentioned in the literature.

These mobile sensor networks are highly scalable, and they have redundancy to vehicle failures. Due to the short endurance of most of these UAVs, refuelling strategies are also implemented. However, these robotic networks have limitations on the on-board processing and communication resources. Therefore, a penalty in terms of CPU time and communication bandwidth has to be taken into account. Also, efficient communication protocols have to be implemented to ensure a common network goal.

1.1 Motivation and relevance

Tracking multiple dynamic targets with a limited mobile sensors network, carried by a group of UAVs, rises the tracking problem where the resources are limited. In a limited resources scenario, the number of agents can be significantly lower than the number of targets. Also, non-holonomic vehicle dynamics are often needed to model fixed-wings. Introducing non-holonomic constraints, they bring another challenge to the picture. On-board processing and communication protocols constrain the problem to be implemented in a simple and distributed way among the UAVs.

Facing these difficulties was attempted by the research community, due to the wide range of target-tracking scenarios where such an algorithm could be applied. Originally motivated by a proposal for the International Micro Air Vehicle Conference and Flight Competition (IMAV) 2014, we propose to steer a group of UAVs to optimally track a larger set of dynamic targets.

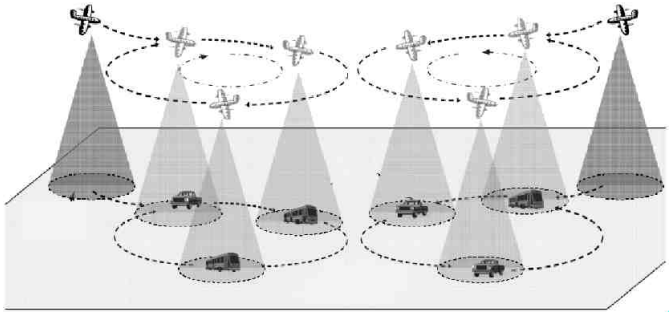

Figure 1.1: Multi-UAV Multi-Target Tracking (from Tang and Ozguner [51]).

1.2 Research goal and main contributions

Our goal is to develop target-tracking algorithms using a mobile sensors network composed by a group of UAVs. The proposed problem is to determine the control input for each UAV that optimise tracking multiple dynamic targets travelling inside a fixed area of interest in a complete autonomous way. The implementation should be distributed assuming no central control unit and allowing being scalable for large mobile sensor networks. The UAVs are able to communicate through a wireless protocol, allowing only local communication between sensors.

This problem can be split into different sub-goals that we seek to achieve:

- define an appropriate performance criteria and define the set of constraints that the algorithm needs to fulfil;
- implement a centralised algorithm and analyse its behaviour;
- distribute the problem among the agents, and convexify it in order to be implementable in real-time;
- discuss communication schemes between neighbour agents and design distributed algorithms based on local communication-only to solve the distributed problem;
- analyse the distributed solutions and compare with the original centralised one;
- deal with collision avoidance and slow changing number of UAVs.

A single-best estimation performance criterion is introduced, and a novel two-layer approach is proposed to solve the tracking problem. Two tracking problems are considered to allow collaboration between heterogeneous teams, i.e., the use of quadrotors and fixed-wings. Two distributed algorithms are designed based on local communication between neighbour vehicles.

The main contributions of this work are the following.

1. *Problem Formulation:* given a limited resources scenario, where we have fewer agents than targets, we assume that there is no need of sensing a target with more than one agent. This assumption motivates our single best-estimation criterion. Another contribution is the use of two different agent dynamics in the problem formulation: a quadrotor and a fixed-wing model.
2. *Novel two-layer approach:* with an initial centralised non-convex formulation that does not fulfil our goals, we propose a novel two-layer relaxation to solve the problem. This approach implements a first layer with a distributed Assignment Problem among the agents and a second local layer with single-agent Tracking Problems. Distribution and convexification is achieved with our novel formulation. A theoretical comparison between the two approaches is made, and by the means of a theorem, the optimal steady-state solution for static targets is proved to be the same.

3. *Final distributed algorithms:* we achieved a fully distributed implementation based on asynchronous-unreliable communication between neighbour agents. The tracking problems are formulated as Semi-Definite Positive programs achieving real-time for both agent models. The time complexity of our distributed algorithms does not depend on the number of agents and is linear with the number of targets.

1.3 Outline

This Master's Thesis consists of ten chapters organised as follows.

- Chapter 1 introduces and motivates the topic, proposes the research goals that we seek to achieve and states the main contributions of this project.
- Chapter 2 formulates our optimisation problem, discusses possible implementations and formally states our problem.
- Chapter 3 provides a brief survey on target-tracking, analyses the most important literature on the topic and other related work, and compares our formulation with others.
- Chapter 4 formalises two optimisation-based problems, for quadrotor and fixed-wing agents, and solves these problems using a naive centralised approach. Simulation results are a proof-of-concept of our formulation, but the current approach does not fully meet our goals: it is non-convex problem with a centralised implementation.
- Chapter 5 deals with the limitations of the naive centralised solution and proposes a novel two-layer approach. This approach relaxes and distributes our problem formulation to face the computational time and time complexity goals. This approach is divided into a Assignment Problem (AP) and local Tracking Problems (TPs).
- Chapter 6 formulates the assignment problem has a Voronoi partition problem. The problem is distributed and solved using two algorithms: a synchronous-distributed and an asynchronous-distributed algorithm. Convergence properties, simulation results and the communication protocol needed for each algorithm is discussed.
- Chapter 7 formulates the local tracking problems using a quadrotor or a fixed-wing agent model. The quadrotor's problem is formulated in a Semi-Definite Positive program (SDP) form, while the fixed-wing's, due to its non-holonomic dynamics, is implemented with a Sequential Convex Programming (SCP) algorithm which solves linearised SDPs iteratively.
- Chapter 8 introduces the pseudo-code to implement on each UAV the synchronous- and asynchronous-distributed solutions of our two-layer algorithms. The naive centralised and the two-layer approaches are compared, and the steady-state solution is proved to be the same. The use of heterogeneous teams is discussed and two emergency manoeuvres are introduced (collision avoidance and agents refuelling).
- Chapter 9 presents the simulation results and compares the different algorithms on the agents performance. Sensing and actuators noise is also introduced and their effects studied.
- Chapter 10 concludes the thesis and mentions future research topics.

Chapter 2

Problem formulation

Our multi-UAV multi-target tracking problem can be seen as an active sensing problem (Mihaylova et al. [36]). Let the set of coordinates $(E, N, h) \in \mathbb{R}^3$, define a 3D convex space $S := [E, N, h]^\top \subset \mathbb{R}^3$. We define $(E, N) \in \mathbb{R}^2$ as the East and North plane coordinates respectively, and $h \in \mathbb{R}$ as the altitude coordinate with respect to a fixed ground-level (Figure 2.1).

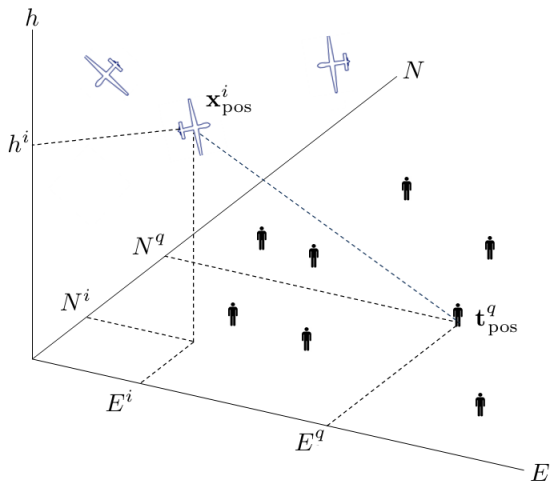

Figure 2.1: Multi-UAV Multi-Target Tracking.

Let a group of n UAVs represent our set of autonomous interacting entities called *agents*. The entire set of agents constitutes our global system. These agents are modelled as *discrete time dynamical* systems

$$\mathbf{x}^i_{k+1} = \mathbf{f}^i(\mathbf{x}^i_k, \mathbf{u}^i_k, \boldsymbol{\eta}^i_k), \quad i \in \{1, ..., n\}, \tag{2.1}$$

where $\mathbf{f}^i$ is the non-linear model of the agent i; $\mathbf{x}^i_k$ is the state, $\mathbf{u}^i_k$ the control input, and $\boldsymbol{\eta}^i_k$ the system noise of the agent i at the time-step k. The position of the agent i, $\mathbf{x}^i_{k,\text{pos}} = [E^i_k, N^i_k, h^i_k]^\top \in S$, is included in the state $\mathbf{x}^i_k$ which depends on the model of the agent i.

Let a group of m ground vehicles represent our set of moving targets of interest called *targets*. The set of targets moves according to the following dynamics

$$\mathbf{t}^q_{k+1,\text{pos}} = \mathbf{t}^q_{k,\text{pos}} + \mathbf{t}^q_{k,\text{vel}} \Delta t, \quad q \in \{1, ..., m\}, \tag{2.2}$$

where $\mathbf{t}^q_{k,\text{pos}} = [E^q_k, N^q_k, h^q_k]^\top \in S$ is the position, $\mathbf{t}^q_{k,\text{vel}}$ is the velocity of the target q at the time-step k, and Δt is the discrete time-step. Let the set of agents have *sensors* capable of sensing the targets. These

sensors are modelled by

$$\mathbf{z}_k^{i,q} = \mathbf{h}^i(\mathbf{x}_k^i, \mathbf{t}_{k,\text{pos}}^q, \boldsymbol{\zeta}_k^{i,q}), \quad i \in \{1, ..., n\}, \tag{2.3}$$

where $\mathbf{h}^i$ is the sensing model of the agent i; $\mathbf{x}_k^i$ the state of the agent i and $\mathbf{t}_{k,\text{pos}}^q$ is the position of the target q; $\mathbf{z}_k^{i,q}$ is the measurement vector, and $\boldsymbol{\zeta}_k^{i,q}$ the sensing noise of the agent i towards the target q at the time-step k. Depending on $\mathbf{h}, \mathbf{x}, \mathbf{t}$, and the noise level/model, the agents are able to estimate the target position more or less accurately.

Our goal is to determine the control input $\mathbf{u}^i$ *for each agent* i *so that we optimise a certain metric based on the target estimation quality.*

Defining a sensing criterion, we introduce a performance criterion that measures how optimal is a certain state and control law. Typically in active sensing problems, the performance criterion is composed by two components: minimising the target estimation uncertainty and minimising the control effort. By minimising the uncertainty, we maximise the information gathered about the targets. The second term represents the utility, i.e., the cost or energy spent to steer the agents. Therefore, the quality criterion can be described by the minimisation of the cost function

$$J_k(\mathbf{x}_k, \mathbf{u}_k) := \sum_j \beta^j \mathcal{U}^j(\mathbf{x}_k) + \sum_l \rho^l \mathcal{C}^l(\mathbf{u}_k), \tag{2.4}$$

where $\mathcal{U}$ represents the j terms that measure the *expected target uncertainty*, and $\mathcal{C}$ represents the l terms that measure the *expected utility-cost* spent in tracking; the coefficients β^j and ρ^l weight the different impacts to achieve a scalar criterion J_k. The terms $\mathcal{U}$ and $\mathcal{C}$ are, respectively, functions of the systems' states $\mathbf{x}_k$ and inputs $\mathbf{u}_k$. The states are defined by the vector of all the agents' states $\mathbf{x}_k = [\mathbf{x}_k^1, ..., \mathbf{x}_k^n]^\top$, and the inputs given by all the agents' inputs $\mathbf{u}_k = [\mathbf{u}_k^1, ..., \mathbf{u}_k^n]^\top$. The cost-function (2.4) is also called the single-step performance criterion.

The performance criterion can be also defined using a Receding Horizon Controller (RHC) which seeks to minimise the single-step ahead performance over a fixed finite-horizon $N < \infty$ (Borrelli et al. [8]). The resulting controller applies only the first optimal control-input, and at the next time-step, the same finite-horizon problem is solved over a shifted horizon. Using a prediction of the future states

$$\widehat{\mathbf{x}}_{k+t|k}^i = \mathbf{f}^i(\widehat{\mathbf{x}}_{k+t-1|k}^i, \mathbf{u}_{k+t-1}^i), \quad i \in \{1, ..., n\}, \quad t \in \{1, ..., N\},$$

where $\widehat{\mathbf{x}}_{k+t|k}^i$ is the prediction t time-steps ahead of the state of the agent i, and we assume to know the current agents' positions $\widehat{\mathbf{x}}_{k|k}^i = \mathbf{x}_k^i$, we can define a finite-horizon performance criterion

$$\bar{J}_k(\mathbf{x}_k, \mathbf{u}_k, ..., \mathbf{u}_{k+N-1}) := \sum_{t=0}^{t=N-1} \left[J_k(\widehat{\mathbf{x}}_{k+t+1|k}, \mathbf{u}_{k+t}) \right] + \varphi(\widehat{\mathbf{x}}_{k+N|k}), \tag{2.5}$$

where $\widehat{\mathbf{x}}_{k+t+1|k}$ is the prediction $t+1$ time-steps ahead of the system states, $\mathbf{u}_{k+t}$ is the control input given to the system at time $k+t$, J_k is the single-step performance criterion given by Equation (2.4), and $\varphi(\widehat{\mathbf{x}}_{k+N|k})$ represents an "extra" contribution of the terminal state $\mathbf{x}_{k+N}$ to the cost function, also known as terminal cost. When chosen properly, the terminal cost is highly effective providing stronger guarantees to the predictive control law (Murray [37]).

The minimisation of the performance criterion is typically constrained, i.e., both the system and the inputs are usually bounded. These bounds can be generally described under a *set of constraints*

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{x}_k, \mathbf{u}_k, ..., \mathbf{u}_{k+N-1}) \leq \mathbf{c}_{\max} \tag{2.6}$$

where $\mathbf{c}$ represents the physical variables constrained by the bounds $\mathbf{c}_{\min}, \mathbf{c}_{\max}$.

2.1 Problem definition

We are interested in a multi-UAV multi-target tracking problem where the resources are limited, i.e., the number of agents at our disposal is lower than the number of targets to be sensed $n \ll m$, and where sensing a target with one agent only is enough to estimate its position. Considering a sensing region S, a set of n agents that seek to sense m targets, we define our problem as follows.

2.1.1 Performance criterion

In general, the tracking of a certain target q by an agent i depends on many factors, such as distance, weather conditions, sensors, shape of the target, etc. However, assuming that the target estimation quality is inversely proportional to the distance agent-target, we define the tracking quality as follows.

Definition 1 (Tracking Quality)**.** *The tracking quality of the agent i towards the target q is given by the inverse of distance between them*

$$Q^{i,q} := \left(||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right)^{-1} = \left((\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}})^\top (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}) \right)^{-1}.$$

Our performance criterion is derived here step-by-step. First we note that to maximise the tracking quality (Definition 1), we seek to minimise the distance agent-target $||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2$. Starting with the performance criterion

$$J := \sum_{q=1}^{m} \left(\sum_{i=1}^{n} ||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right),$$

we define the optimal tracking quality of all agents over all targets. This formulation will obviously bring all agents towards the same position. The agents will tend to move where the distance towards all the targets is lower, i.e., the "set of targets' centroid".

In our formulation we consider a limited-resources scenario where we have fewer agents than targets $n \ll m$. This assumption is motivated by realistic scenarios where we cannot afford to have more than one flying UAV to sense only one target of interest. Thus, we would like to have a problem formulated in such a way that the UAVs naturally split themselves to cover different groups of targets. Instead of having the agents trying to gather information about all the targets. We want to divide the targets in groups where each group is tracked by a single agent.

As said, we are assuming that one agent is capable of gathering enough information about a target. Otherwise we would need more than one agent for tracking each target which may not be feasible in a limited-resources scenario. Note that this assumption is not too restrictive with the current technology (Parker [44] and Parker and Emmons [46]), one UAV with a on-board camera and a ground-map is able to roughly estimate the position of a moving target. Therefore, we want each target to be sensed only by one agent, as close as possible, to obtain the single best sensing quality. Formulating the performance criterion

$$J' := \sum_{q=1}^{m} \left(\min_{i \in \{1,\dots,n\}} \{ ||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \} \right),$$

we represent the goal of obtaining the single best tracking of each target from the closer agent, regardless the sensing obtained from other further agents.

Then, adding a balanced input effort weight, we propose

$$J'' := \sum_{q=1}^{m} \left(\min_{i \in \{1,\dots,n\}} \{||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} \right) + \rho \sum_{i=1}^{n} ||\mathbf{u}^i||^2,$$

where the factor ρ represent the effort of the *utility-cost* with respect to the *target uncertainty*. This control-effort factor helps stabilizing the optimisation-based algorithm. Thus, with a target uncertainty term $\mathcal{U}$ given by the distance towards the closer agent with unit-weights $\beta^j = 1, \;\; j = \{1, ..., m\}$, and with a utility cost term $\mathcal{C}$ given by the 2-norm of the input vector with constant-weights $\rho^l = \rho, \;\; l = \{1, ..., n\}$ (Equation (2.4)); we are able to formulate our single-step performance criterion.

Definition 2 (Single-step Performance Criterion)**.** *Giving a set of agents with positions* $\mathbf{x}^i_{k,\text{pos}}, \;\; i = 1, ..., n$, *and a set of targets with positions* $\mathbf{t}^q_{k,\text{pos}}, \;\; q = 1, ..., m$, *at the current time-step* k, *we formulate the following performance criterion*

$$J_k(\mathbf{x}_k, \mathbf{u}_k) := \sum_{q=1}^{m} \left(\min_{i \in \{1,\dots,n\}} \{||\mathbf{x}^i_{k,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right) + \rho \sum_{i=1}^{n} ||\mathbf{u}^i_k||^2, \tag{2.7}$$

where the first term seeks to obtain the single best tracking from the closer agent, and the second term ρ *adds a balanced input control effort.*

Note that, with a single step ahead controller, if the inputs at time k do not make changes in the agents' positions at time $k+1$, a trivial local minimum is obtained with null inputs. Therefore, we need to formulate the performance criterion with a predictive control horizon. Using the single-step performance criterion of Definition 2 over a finite predictive horizon of N steps, and adding a terminal horizon cost, we can define the N-steps performance criterion (Equation (2.5))

Definition 3 (N-steps Performance Criterion)**.** *Giving a set of agents with positions* $\mathbf{x}^i_{k,\text{pos}}, \;\; i = 1, ..., n$, *and a set of targets with positions* $\mathbf{t}^q_{k,\text{pos}}, \;\; q = 1, ..., m$, *at the current time-step* k, *a prediction model given by the agents' dynamics, and a finite predictive horizon of* N *steps, we formulate the following performance criterion*

$$\begin{aligned} \bar{J}_k(\mathbf{x}_k, \mathbf{u}_k, ..., \mathbf{u}_{k+N-1}) := \sum_{t=0}^{N-1} \Bigg[\sum_{q=1}^{m} \left(\min_{i \in \{1,\dots,n\}} \{||\widehat{\mathbf{x}}^i_{k+t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right) + \rho \sum_{i=1}^{n} ||\mathbf{u}^i_{k+t}||^2 \Bigg] + \\ + \varphi \sum_{q=1}^{m} \left(\min_{i \in \{1,\dots,n\}} \{||\widehat{\mathbf{x}}^i_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right), \end{aligned} \tag{2.8}$$

where the first term seeks to obtain the single best tracking from the closer agent, the second term ρ *adds a balanced input control effort into the cost function, and the third term* φ *adds a terminal horizon cost. Note that, as the targets movement is random, we guide the agents along the time horizon* N *with respect to the current position of the targets* $\mathbf{t}^q_{k,\text{pos}}, \;\; q = 1, ..., m,$.

The N-steps Performance Criterion (2.8) will be our performance criterion.

2.1.2 Agents

A set of UAVs, here called agents, are considered.

Definition 4 (Agents)**.** *Giving a space* S, *the agents are the set of* n *UAVs carrying out sensors capable of tracking targets. Each agent is defined by its state* $\mathbf{x}^i$, *which includes its position* $\mathbf{x}^i_{\text{pos}} = [E^i, N^i, h^i]^\top \in S$, *and* i *is its unique number* $i = 1, ..., n$.

Our focus is on the high-level control and coordination of UAVs. Therefore, we will consider basic agent models in order to have simple path planning problems to solve, and leave inner loop controllers deal with the full vehicle dynamics. Two agent models (Equation (2.1)) will be used:

1. *Quadrotor model*: defined with a discrete double integrator unit point-mass dynamics

$$\begin{aligned} \mathbf{x}^i_{k+1} &\equiv \begin{bmatrix} \mathbf{x}^i_{k+1,\text{pos}} \\ \mathbf{x}^i_{k+1,\text{vel}} \end{bmatrix} = \mathbf{f}^i_{\text{quad}}(\mathbf{x}^i_k, \mathbf{u}^i_k) \equiv \mathbf{A} \begin{bmatrix} \mathbf{x}^i_{k,\text{pos}} \\ \mathbf{x}^i_{k,\text{vel}} \end{bmatrix} + \mathbf{B}\mathbf{u}^i_k, \quad i \in \{1, ..., n\}, \\ \mathbf{A} &= \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3 \Delta t \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times 3} \\ \mathbf{I}_3 \Delta t \end{bmatrix}, \end{aligned} \tag{2.9}$$

where $\mathbf{x}^i$ is the state of the agent i given by its position $\mathbf{x}^i_{\text{pos}}$ and its velocity $\mathbf{x}^i_{\text{vel}} = [v^i_E, v^i_N, v^i_h]^\top$, and the input $\mathbf{u}^i = [f^i_E, f^i_N, f^i_h]^\top$ of the agent i corresponds to the force applied along the three axis. We consider the following limitations (2.10) on the quadrotor's dynamics, meaning that the quadrotors have a minimum flying altitude, bounded velocity and force (Figure 2.2).

$$\begin{cases} h^i \geq h_{\min} \\ -v_{\max} \leq v^i_E, v^i_N, v^i_h \leq v_{\max} \\ -f_{\max} \leq f^i_E, f^i_N, f^i_h \leq f_{\max} \end{cases}, \quad i = \{1, ..., n\}. \tag{2.10}$$

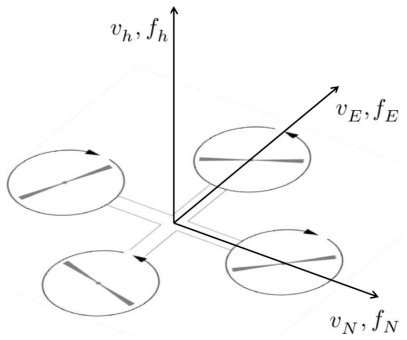

Figure 2.2: Quadrotor model.

2. *Fixed-wing model*: defined with fixed-altitude discrete *Dubins* car model (Dubins [20]):

$$\mathbf{x}^i_{k+1} \equiv \begin{bmatrix} E^i_{k+1} \\ N^i_{k+1} \\ \theta^i_{k+1} \end{bmatrix} = \mathbf{f}^i_{\text{wing}}(\mathbf{x}^i_k, u^i_k) \equiv \begin{bmatrix} E^i_k \\ N^i_k \\ \theta^i_k \end{bmatrix} + \begin{bmatrix} V_M \cos\theta^i_k \\ V_M \sin\theta^i_k \\ u^i_k \end{bmatrix} \Delta t, \quad i \in \{1, ..., n\}, \tag{2.11}$$

where $\mathbf{x}^i$ is the state of the agent i given by its 2D position $[E^i, N^i]^\top$ and orientation θ^i, and the scalar input u^i of the agent i corresponds to the change of heading command. These agents fly with constant velocity V_M and with a fixed-altitude $\mathbf{x}^i_{\text{pos}} = [E^i, N^i, \bar{h}]^\top, i \in \{1, ..., n\}$. We also consider constraints (2.12) on the turning rate, meaning that the fixed-wings have a maximum turning rate (Figure 2.3).

$$-u_{\max} \leq u^i \leq u_{\max}, \quad i = \{1, ..., n\}. \tag{2.12}$$

We assume to have perfect observation on the agents' state.

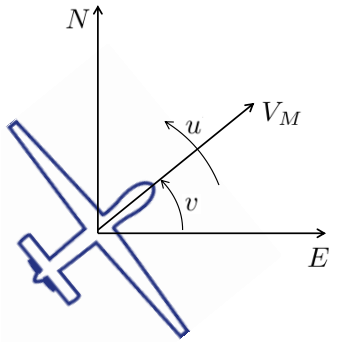

Figure 2.3: Fixed-wing model.

Assumption 1 (Perfect agent state knowledge)**.**

We assume that on-board sensors are able to estimate the agents' states perfectly. Currently, coupling the General Positioning System (GPS) with the Inertial Navigation System (INS), constitutes a GPS/INS sensor which is able to provide accurately the state of UAV systems. Therefore, we consider to have the current agent state included in the system's output.

2.1.3 Targets

A set of moving ground vehicles, here called targets, are considered

Definition 5 (Targets)**.** *Giving a space S, the targets are the set of m moving ground vehicles which we seek to track. Each target is defined by its ground position $\mathbf{t}^q_{\text{pos}} = [E^q, N^q, h^q = 0]^\top \in S$, and q is its unique number $q = 1, ..., m$.*

Two types of targets (Equation (2.2)) will be considered:

1. *Static targets*: targets that do not move with time

$$\mathbf{t}^q_{k+1,\text{pos}} = \mathbf{t}^q_{k,\text{pos}}, \quad q \in \{1, ..., m\}, \tag{2.13}$$

   where $\mathbf{t}^q_{\text{pos}}$ is the position of the target q with no altitude $h^q = 0$.

2. *Random-walk targets*: targets moving with a random-walk model

$$\mathbf{t}^q_{k+1,\text{pos}} = \mathbf{t}^q_{k,\text{pos}} + \mathbf{t}^q_{k,\text{vel}}\Delta t, \quad q \in \{1, ..., m\}, \tag{2.14}$$

   where $\mathbf{t}^q_{\text{pos}}$ is the position of the target q with no altitude $h^q = 0$, and the velocity is given by $\mathbf{t}^q_{\text{vel}} = [w^q_E, w^q_N, 0]^\top$, where w^q_E, w^q_N are zero mean bounded Gaussian noises, bounded as

$$||\mathbf{t}^q_{\text{vel}}||^2 = (w^q_E)^2 + (w^q_N)^2 \leq (W_{\max})^2, \quad q \in \{1, ..., m\}.$$

Regarding the targets' observation, we consider the noisy sensing model (2.3):

$$\mathbf{z}^{i,q}_k = \mathbf{t}^q_{k,\text{pos}} + \boldsymbol{\zeta}^{i,q}_k ||\mathbf{x}^i_{k,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||, \quad \forall_{i \in \{1,...,n\}} : ||\mathbf{x}^i_{k,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}|| \leq R^i, \tag{2.15}$$

where $\boldsymbol{\zeta}^{i,q}_k = [\boldsymbol{\zeta}^{i,q}_{k,E}, \boldsymbol{\zeta}^{i,q}_{k,N}, 0]^\top$, and $\boldsymbol{\zeta}^{i,q}_{k,E}, \boldsymbol{\zeta}^{i,q}_{k,N}$ are zero mean bounded Gaussian noises.

Therefore, by minimising the distance agent-target $||\mathbf{x}^i_{k,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||$ with our performance criterion (2.8), we optimise the tracking quality of our sensing. However, we assume the targets' positions to be roughly given by the set of measurements within the limited sensing range R^i

$$\mathbf{z}^{i,q}_k \approx \mathbf{t}^q_{k,\text{pos}}, \quad \forall_{i\in\{1,\dots,n\}} : ||\mathbf{x}^i_{k,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}|| \leq R^i, \tag{2.16}$$

such that our formulation optimises with respect to the measured targets' positions. In Chapter 9, we show the results of this assumption by considering different boundaries in the noisy sensing model (2.15).

2.1.4 Full target coverage

The minimisation of the performance criterion is constrained (Equation (2.6)) to ensure full target coverage. We want to guarantee that each target q has at least one agent capable of tracking it, i.e., each target q is at least within the sensing range of one agent. [1] Considering sensors with a maximum range, an agent i is only capable of sensing a target q within a certain range $||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}|| \leq R^i$. Therefore we can define the full target coverage constraint as

$$\forall_{q\in\{1,\dots,m\}} \exists_{i\in\{1,\dots,n\}} : ||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \leq (R^i)^2. \tag{2.17}$$

Using the *Schur* complement of a matrix block, the full target coverage constraint can be written as a positive semi-definite matrix constraint, which will be useful in Chapter 7. First we define the Schur complement

Definition 6 (Schur Complement)**.** *Let the real square matrix* $\mathbf{M}$ *be of the form*

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{bmatrix}, \tag{2.18}$$

with $\mathbf{A}, \mathbf{C}$ *real square matrices, and* $\mathbf{A}$ *invertible. We define* $\mathbf{S}$ *as*

$$\mathbf{S} = \mathbf{C} - \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B}, \tag{2.19}$$

and we call $\mathbf{S}$ *the Schur complement of* $\mathbf{A}$ *in* $\mathbf{M}$*.*

And we recall the proposition using this complement

Proposition 1 (Schur Complement)**.** *Considering a matrix* $\mathbf{M}$ *of the form* (2.18) *and its Schur complement* (2.19) $\mathbf{S}$*, if* $\mathbf{A}$ *is a invertible positive definite matrix* $\mathbf{A} \succ 0$*, then* $\mathbf{M} \succeq 0 \Leftrightarrow S \succeq 0$*.*

Then, defining the matrix block

$$\mathbf{M} = \begin{bmatrix} \mathbf{I}_3 & (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}) \\ (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}})^\top & (R^i)^2 \end{bmatrix},$$

and having $\mathbf{I}_3^{-1} = \mathbf{I}_3 \succ 0$, from the Proposition 1 we conclude

$$\mathbf{M} \succeq 0 \Leftrightarrow S = (R^i)^2 - (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}})^\top (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}) \geq 0 \Leftrightarrow ||\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \leq (R^i)^2,$$

allowing us to rewrite the full target constraint (2.17), as a semi-definite positive matrix constraint.

[1] We implicitly assume that at $t = 0$ we have full target coverage. Otherwise, the agents should be initially spread around the plane coordinates $(E, N) \in Q$ until the full target coverage constraint is satisfied.

Definition 7 (Full target coverage constraint)**.** *Giving a set of agents with positions* $\mathbf{x}^i_{k,\text{pos}},\ \ i = 1, ..., n$ *and a set of targets with positions* $\mathbf{t}^q_{k,\text{pos}},\ \ q = 1, ..., m$*, we formulate the full target constraint as*

$$\forall_{q\in\{1,...,m\}}\exists_{i\in\{1,...,n\}} : \mathbf{M} = \begin{bmatrix} \mathbf{I}_3 & (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}}) \\ (\mathbf{x}^i_{\text{pos}} - \mathbf{t}^q_{\text{pos}})^\top & (R^i)^2 \end{bmatrix} \succeq 0. \tag{2.20}$$

2.2 Optimisation problem

We can finally state our target-tracking optimisation problem embodying our goals.

Problem 1 (Target tracking optimisation problem)**.**

Given the performance criterion $\bar{J}_k$ (2.8)*, either the quadrotors dynamics and limitations* (2.9)-(2.10)*, or the fixed-wings dynamics and limitations* (2.11)-(2.12)*, and the full target coverage constraint* (2.20)*; we define the problem of computing the control input* $\mathbf{u} := \{\mathbf{u}_k, ..., \mathbf{u}_{k+N-1}\}$ *of* n *agents to track* m *targets inside the sensing region* S *as*

$$\min_{\mathbf{u}}\ \bar{J}_k(\mathbf{x}_k, \mathbf{u}_k, ..., \mathbf{u}_{k+N-1})\ \text{ s.t.}$$
$$\begin{cases} \text{agent dynamics and limitations (quadrotors or fixed-wings),} \\ \text{full target coverage constraint.} \end{cases}$$

2.3 Distributed implementation

Typically, optimisation problems can be implemented in different ways as (see Figure 2.4 from Yang and Johansson [53])

- *Centralized optimisation*: the problem is posed in a centralised way, where a vector of global variables is chosen by a centralised entity to minimise a global performance criterion. In this case, the agents will be fully steered from a fixed-ground base at each time step.
- *Decomposed optimisation*: the problem is divided into sub-problems but a centralised unit is still needed. In this case, the agents can be steered locally however part of the optimisation problem has to be solved centrally. This central unit needs to communicate with all the agents at each time step.
- *Distributed optimisation*: the problem is divided into n sub-problems to be solved by n decision-making agents. In this case, the optimisation problem is completely solved locally, i.e., there is no need for a central unit. Communication schemes are defined to share information among the agents (cooperation).
- *Non-cooperative optimisation*: the problem is divided into n sub-problems to be solved totally independently by n decision-making agents. Each agent optimises its own criterion regardless the others. No communication strategies are used, i.e., there is no cooperation.

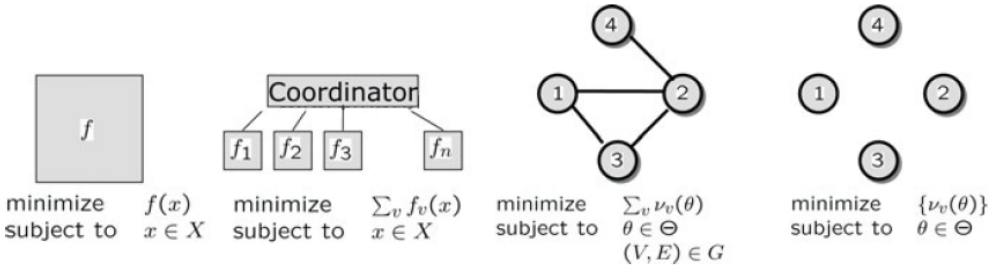

Figure 2.4: Centralized, Decomposed, Distributed and Non-Cooperative optimisation schemes (from Yang and Johansson [53]).

On one hand, when the number of agents increase, solving the problem in a centralised way may become computationally intractable. Also, assuming a limited communication network, a decomposed scheme needs constant communication of all agents with a centralised unit (or all-to-all communications) which is hardly implementable in real applications. On the other hand, non-cooperative optimisation hardly assures a optimal solution for global tracking goals. Thus, we want to solve the problem using a distributed layout.

To implement our optimisation-based algorithm, we aim for a *distributed optimisation* where each agent makes decisions communicating with other agents to cooperatively strive for the global good. The global tracking problem is divided in n sub-problems which are solved in parallel and locally by each agent, improving the scalability to large mobile sensor networks. Also, we seek to implement algorithms based on as few communication as possible and between neighbour agents only.

2.4 Concluding Remarks

To summarise, we state our problem as follows.

How to guide a group of UAVs to optimally track a set of moving ground targets. Considering a limited resources scenario, where the number of agents is lower than the number of targets, we seek to obtain the single best estimation of all targets. We consider a target estimation quality inversely proportional to the distance agent-target. Assuming sensors with a maximum sensing range, we aim to ensure full target coverage at all times. Assuming a limited communication network, we want to implement distributed algorithms with as few communication as possible between neighbour agents only.

Chapter 3

Related work

An overall understanding of how the research community have addressed and formulated the problem is given. The tracking goal is defined in different ways which lead to different performance criteria, different agent models are considered, and the target estimation differs within the literature. Also, the literature can be divided according to the tracking application: single- or multi-target tracking, with a centralised or a distributed implementation.

3.1 Performance criteria

The performance criterion (2.4) typically weights the *expected target uncertainty* $\mathcal{U}$ of the system and the *expected utility-cost* $\mathcal{C}$ to move the agents.

Expected target uncertainty

The quality of the target estimation is often related to the *Fisher* information or the *Shannon* information (Grocholsky et al. [22]).

The Fisher Information Matrix (Definition 8) is based on the covariance matrix of the estimated position probability distribution which is a measure of the estimation uncertainty (Zhao et al. [55]).

Definition 8 (Fisher Information Matrix (FIM))**.** *Giving n sensors with models* (2.3)

$$\mathbf{z}^i = \mathbf{h}^i(\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}}) + \boldsymbol{\zeta}^i, \quad i \in \{1, ..., n\},$$

where $\mathbf{z}^i$ *denotes the measurement of the sensor* i*, the function* $\mathbf{h}^i(\mathbf{x}^i, \mathbf{t}_{\text{pos}})$ *models the sensor* i*,* $\mathbf{x}^i_{\text{pos}}$ *is the position of the sensor* i*,* $\mathbf{t}_{\text{pos}}$ *is the position of the target, and* $\boldsymbol{\zeta}^i$ *is the zero-mean Gaussian noise with covariance* $\boldsymbol{\Sigma}^i$*. The FIM is expressed as*

$$\mathbf{F}(\mathbf{x}, \mathbf{t}) := \sum_{i=1}^{n} \left(\frac{\partial \mathbf{h}^i}{\partial \mathbf{t}_{\text{pos}}} \right)^{\top} (\boldsymbol{\Sigma}^i)^{-1} \left(\frac{\partial \mathbf{h}^i}{\partial \mathbf{t}_{\text{pos}}} \right).$$

Remark 1 (Sensor Models)**.**

Typically the sensor models used are (Zhao et al. [55])

- *range-only:* $\mathbf{h}^i(\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}}) = ||\mathbf{t}_{\text{pos}} - \mathbf{x}^i_{\text{pos}}||$.

- *bearing-only:* $\mathbf{h}^i(\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}}) = \frac{\mathbf{t}_{\text{pos}} - \mathbf{x}^i_{\text{pos}}}{||\mathbf{t}_{\text{pos}} - \mathbf{x}^i_{\text{pos}}||}$.
- *range-and-bearing:* $\mathbf{h}^i(\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}}) = \mathbf{t}_{\text{pos}} - \mathbf{x}^i_{\text{pos}}$.

Since the target uncertainty metric $\mathcal{U}$ is defined as a scalar function, and no scalar function can capture all aspects of the FIM matrix, different functions are proposed according to the application (Mihaylova et al. [36], Uciński [52]):

- *D-optimal design*: steers the agents to the positions $\mathbf{x}$ that minimise the determinant $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = -\det(\mathbf{F}(\mathbf{x}, \mathbf{t}))$, or its logarithm $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = -\log(\det(\mathbf{F}(\mathbf{x}, \mathbf{t})))$. This metric is designed to minimise the volume of the confidence ellipsoid of the estimates.
- *A-optimal design*: steers the agents to the positions $\mathbf{x}$ that minimise the trace $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = tr(\mathbf{F}(\mathbf{x}, \mathbf{t})^{-1})$ which suppresses the variance of the estimations.
- *L-optimal design*: steers the agents to the positions $\mathbf{x}$ that minimise the weighted trace $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = tr(\mathbf{W}\mathbf{F}(\mathbf{x}, \mathbf{t})^{-1})$. This criterion is a generalisation of the previous one and a special case can be seen in De Geeter et al. [16].
- *Sensitivity criterion*: steers the agents to the positions $\mathbf{x}$ that minimise the trace $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = -tr(\mathbf{F}(\mathbf{x}, \mathbf{t}))$. This metric represents a simplified version of the *A*-optimal criterion.
- *E-optimal design*: steers the agents to the positions $\mathbf{x}$ that minimise the maximum eigenvalue $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = \lambda_{\max}(\mathbf{F}(\mathbf{x}, \mathbf{t})^{-1})$, minimising the length of the largest axis of the ellipsoid.
- *Zhao, et al.'s criterion*: steers the agents to the positions $\mathbf{x}$ that minimise the metric $J(\mathbf{F}(\mathbf{x}, \mathbf{t})) = ||\mathbf{F}(\mathbf{x}, \mathbf{t}) - \bar{\lambda}\mathbf{I}||^2$, where $\bar{\lambda} = 1/d \sum_{i=1}^{d} \lambda_i$, $\{\lambda_i\}_{i=1}^{d}$ denotes the eigenvalues of the FIM $\mathbf{F}(\mathbf{x}, \mathbf{t})$, and $\mathbf{I}$ is the identity matrix. This objective function has close connection to the *D-optimal design* and it has a easier analytical tractability (Zhao et al. [55]).

The Shannon information is based on the probability density function, which models the observations made by the agents. This metric describes the probability of obtaining a particular observation $\mathbf{z}^i$ given the positions $\mathbf{x}^i_{\text{pos}}$ and $\mathbf{t}_{\text{pos}}$. The concept of Shannon information or Entropy (Definition 9) measures the inherent uncertainty of a distribution representing another measurement of the uncertainty of a state estimation (Mathews and Durrant-Whyte [34]).

Definition 9 (Entropy)**.** *Giving n sensors with observations modelled by the conditional density*

$$p(\mathbf{z}^i|\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}}), \quad i \in \{1, ..., n\},$$

the entropy is defined as the negative of the expectation of the distribution $p(\mathbf{z}^i|\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}})$

$$H_{\mathbf{z}}(p, \mathbf{x}, \mathbf{t}) := -E_{\mathbf{z}}[\log(p(\mathbf{z}^i|\mathbf{x}^i_{\text{pos}}, \mathbf{t}_{\text{pos}}))],$$

where $E[.]$ *denotes the expected value.*

Different performance criteria based on the Shannon information are used, being the minimisation of the entropy the most common (Mihaylova et al. [36]):

- *entropy criterion*: steers the agents to the positions $\mathbf{x}$ that minimise the entropy of the posterior distribution $J(p, \mathbf{x}, \mathbf{t}) = H_{\mathbf{z}}(p, \mathbf{x}, \mathbf{t})$. This metric is designed to minimise the compactness of a distribution.

Information-based performance criteria like the ones presented here are typically non-convex in the optimisation variables, i.e., agents' positions and control variables; and very often Non-deterministic Polynomial-time hard (NP-hard) (Zhou and Roumeliotis [58],Zhou and Roumeliotis [59]). [1] This class of problems is extremely computational demanding, it has lead Researchers to develop heuristics to solve them.

Expected utility-cost

The cost $\mathcal{C}$ measures the energy spent to move the agents. This factor is a function of the inputs $\mathbf{u}$, typically formulated as:

- *absolute input*: minimises the weighted sum of the absolute value of the inputs given to the system $J(\mathbf{u}_k) = \sum_{i=1}^{n}(\rho^i||\mathbf{u}_k^i||^2)$.
- *relative input*: minimises the weighted sum of the relative inputs given to the system $J(\Delta\mathbf{u}_k) = \sum_{i=1}^{n}(\rho^i||\Delta\mathbf{u}_k^i||^2)$, where $\Delta\mathbf{u}_k^i = \mathbf{u}_k^i - \mathbf{u}_{k-1}^i$ is the changing in the input.

3.2 Agent models

Different agent models (2.1) are used to insert the vehicle dynamics in the optimisation-based algorithm. Apart from the quadrotor model (Point-mass model (2.9)) and the fixed-wing model (*Dubins* model (2.11)), other models often used are:

- *Way-points dynamics*, single integrator dynamics:

$$\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \mathbf{u}_k^i, \quad i \in \{1, ..., n\},$$

  where $\mathbf{x}^i$ is the position and $\mathbf{u}^i$ is the relative way-point of the agent i. The goal is to find optimal way-points. Note that if the input is constrained, it represents having agents with limited velocity.

  This formulation is generally suitable for any vehicle which has an efficient low level controller to move between way-points. However, it does not take into account the vehicle dynamics, for example the attitude of the agents is not considered. Moving forwards or backwards has the same control cost which may not be the most suitable for fixed-wing UAVs.

- *Linear dynamics*, Multi-Input Multi-Output (MIMO) linear systems:

$$\mathbf{x}_{k+1}^i = \mathbf{A}\mathbf{x}_k^i + \mathbf{B}\mathbf{u}_k^i, \quad i \in \{1, ..., n\},$$

  where $\mathbf{x}^i$ is the position and $\mathbf{u}^i$ the input of the agent i, $\mathbf{A}$ a state-matrix and $\mathbf{B}$ a input-matrix.

  This dynamics is a generalization of the Point-mass dynamics (2.9) described by a MIMO linear system.

- *Linear varying dynamics*, Linear Parameter-Varying (LPV) systems:

$$\mathbf{x}_{k+1}^i = \mathbf{A}(\boldsymbol{\gamma})\mathbf{x}_k^i + \mathbf{B}(\boldsymbol{\gamma})\mathbf{u}_k^i, \quad i \in \{1, ..., n\},$$

[1] In computational complexity theory, Non-deterministic Polynomial-time hard (NP-hard) is a class of problems which are, informally, "at least as hard as the hardest problems in NP". Therefore, a problem is NP-hard if solving it in polynomial time would make possible to solve all NP problems in polynomial time. The abbreviation NP refers to Non-deterministic polynomial time problems which are accepted to have polynomial time using a non-deterministic Turing machine.

where $\mathbf{x}^i$ is the position and $\mathbf{u}^i$ the input of the agent i, $\boldsymbol{\gamma}$ are the parameters of the system, $\mathbf{A}$ and $\mathbf{B}$ are the state-matrix and the input-matrix given as a function of these parameters.

LPV systems are suitable to approximate non-linear systems. The non-linear model is formulated as a parametrized linear system, where the parametrization is dependent on the states (Bruzelius [10]). Here, a flying-wing can be described parametrizing the state and input matrices as a function of the UAV states.

- *A non-linear 3D dynamics*, using the continuous 6 Degrees-Of-Freedom (DOF) fixed-wing aircraft model presented in Adami and Zhu [1], converted to a discrete-time model using a basic forward difference to express the continuous derivative $\dot{x} \approx \frac{x(k+1)-x(k)}{\Delta t}$:

$$\begin{bmatrix} \mathbf{x}^i_{k+1} \\ \mathbf{v}^i_{k+1} \\ \boldsymbol{\gamma}^i_{k+1} \\ \boldsymbol{\omega}^i_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^i_k \\ \mathbf{v}^i_k \\ \boldsymbol{\gamma}^i_k \\ \boldsymbol{\omega}^i_k \end{bmatrix} + \left(\begin{bmatrix} \mathbf{A}_1(\boldsymbol{\gamma}^i_k)\mathbf{v}^i_k \\ \mathbf{A}_2(\boldsymbol{\omega}^i_k)\mathbf{v}^i_k \\ \mathbf{A}_3(\boldsymbol{\gamma}^i_k)\boldsymbol{\omega}^i_k \\ \mathbf{A}_4(\boldsymbol{\omega}^i_k) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{B}_2 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{B}_4 \end{bmatrix} \begin{bmatrix} \mathbf{f}^i_k \\ \boldsymbol{\tau}^i_k \end{bmatrix} \right) \Delta t, \quad i \in \{1, ..., n\}$$

  where $[\mathbf{x}^i, \mathbf{v}^i, \boldsymbol{\gamma}^i, \boldsymbol{\omega}^i]^\top$ is the state of the agent i ($\mathbf{x}^i$ the position, $\mathbf{v}^i$ the velocity, $\boldsymbol{\gamma}^i$ the angular position and $\boldsymbol{\omega}^i$ the angular velocity), $[\mathbf{f}^i, \boldsymbol{\tau}^i]^\top$ the input given to the agent i ($\mathbf{f}^i$ the force and $\boldsymbol{\tau}^i$ the torque), and the matrices $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4, \mathbf{B}_2, \mathbf{B}_4$ defined in Adami and Zhu [1] as a function of the moments of inertia $I_{xx}, I_{xz}, I_{yy}, I_{yz}$ of the UAV.

  This complex model describes in detail any fixed-wing agent, adding the "manoeuvrability" of the UAV to the optimisation algorithm. However, controlling non-linear systems is not a trivial task. Steering the agents using precise non-linear models increase largely the computational load of the tracking problem. Therefore, for real implementations, we assume to have faster inner loops with low-level controllers to control the UAVs, and a path planning outer loop for target-tracking.

3.3 Estimation

Throughout our work, we assume to estimate the position of the targets exactly (with the exception of the Chapter 8 where we add sensing noise). However, filtering strategies are implemented in the literature to estimate the targets' state from a set of noisy measurements. Extended Kalman Filter (EKF) is usually used for estimating the targets' position and update its estimation (Mathews and Durrant-Whyte [34]). However, these strategies assume a centralised fusion centre to compute and update the estimation. For distributed implementations, other strategies are used

- *Distributed Kalman Filter*: first introduced in Olfati-Saber [40] and used for tracking applications in Olfati-Saber [41], this algorithm relies on embedded consensus filters that dynamically compute averages of sensor data from all sensors in a distributed way. The sensor data and the covariance data pass by Low-Pass and Band-Pass consensus filters respectively. Then, both informations go to Micro Kalman Filter iterations implemented by each agent.

- *Distributed Particle Filter*: implemented and tested in Sheng et al. [47], it uses a low dimensional Gaussian mixture model to approximate local sufficient beliefs and provide a robust tracking performance with reduced communication. The estimation is proved to converge almost surely to the centralised Bayesian estimation. A recent survey on different approaches and algorithms can be found in Hlinka et al. [24].

- *Decentralised Data Fusion*: derived in Manyika and Durrant-Whyte [32] and applied in Grocholsky et al. [22], Ong et al. [43], this approach allows fusing information in a fully distributed approach

where each sensor seeks to maximise the information gathered without knowledge of other decision makers. The network of sensors communicates the state estimations using information measurements.

3.4 Literature review

The literature is divided over the single- and multi-target tracking, with centralised or distributed implementations. Furthermore other relevant related work is analysed.

3.4.1 Centralized single-target tracking

In LaValle et al. [29], the minimisation of the distance travelled by a single agent is considered, while maintaining visibility towards a target. This visibility is introduced as a soft constraint into the cost-function. A finite horizon RHC is implemented to compute the distance travelled while maintaining the target inside a visible range. In terms of sensing-models, omnidirectional cameras are used and considered to be capable of sensing a target inside a given visibility range. The agents dynamics are considered single integrator dynamics and the observation of the state of the agents is perfect. The target is moving and no estimation strategies are implemented. Two schemes are considered: when we can predict the movement of the target and when it is completely random. Visibility-based motion planning is introduced and the problem is considered to be generally a NP-hard problem. Finally, real simulations are considered.

In LaValle et al. [28], the problem of searching for one moving target in a maze is introduced and proved to be NP-hard. The goal is to compute the minimum number of agents needed to find a single target inside a constrained 2D environment. The algorithm seeks to compute trajectories for a set of agents in order to cover the area without missing the target. The searching trajectories are computed *a priori* and a mathematical formulation is used to prove whether there is a chance of the target to escape the agents' radar or not. Omnidirectional cameras are used and considered to be capable to sense a target inside a given visibility range. The agents also have a single integrator dynamics and their states' observation is perfect. Only one target is considered and it is moving the best possible way to avoid being sensed by the agents. Visibility-based motion planning is introduced and an exhaustive set of maze examples are tested to prove the formulation developed.

In Zhou and Roumeliotis [59], 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 NP-hard. The performance criteria is defined in an A-optimal design fashion using the FIM. The problem is solved analytically for one agent only. Then, using cyclic coordinate one-step-ahead descent method, they solve a relaxation of the general NP-hard problem in a way to achieve linear complexity with the number of agents. This algorithm is called the non-linear Gauss-Seidel algorithm. Regarding the sensors, they consider combinations of range-only, bearing-only and range-and-bearing sensors. Single integrator dynamics are considered and the position of the agents is assumed to be known with precision. The targets are dynamic and they move with a random walk. The targets' position is estimated using a recursive centralised EKF.

In Zhao et al. [55], the performance criterion is given by

$$J(\mathbf{F}(\mathbf{x},\mathbf{t})) = ||\mathbf{F}(\mathbf{x},\mathbf{t}) - \bar{\lambda}\mathbf{I}||^2,$$

where $\bar{\lambda}$ given by the average of the eigenvalues $\lambda_i, i = \{1, ..., d\}$ of $\mathbf{F}(\mathbf{x},\mathbf{t})$

$$\bar{\lambda} = \frac{1}{d}\sum_{i=1}^{d}\lambda_i.$$

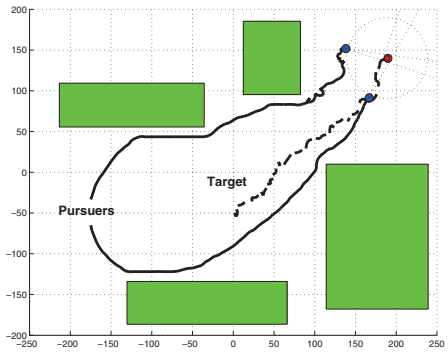

Figure 3.1: Tracking a moving target in a cluttered environment (from Spletzer and Taylor [50]).

This non-convex performance criterion is proved to have closer connections to the D-optimal design formulation, however it simplifies the problem in 3D cases where the determinant of the FIM is hardly analytically tractable. Necessary and sufficient optimal placement conditions are obtained analytically. Regarding the sensors: range-only, bearing-only and Received Signal Strength (RSS) sensors are considered separately. Again, agents dynamics are considered as single integrator dynamics with perfect observations. The targets are static and no estimation is assumed, therefore their position is assumed to be known. Simulations are made using a centralised gradient control law. A gradient control law implements the optimal placement obtained analytically. Distribution implementations are only possible by dividing the group of agents into several sub-teams with non-arbitrary size.

3.4.2 Centralized multi-target tracking

In Spletzer and Taylor [50], the non-convex performance criteria is based on the Shannon information and it captures how the expected error in the estimation varies with the different robot configurations. Particle filtering is used and a gradient-based single-step optimisation is implemented. Different sensors combinations are considered and allowed in their work. 2^{nd} order 2D vehicle dynamics are used. Also, obstacles are considered to constraint the agents' movement (see Figure 3.1). The targets move according to a random walk and estimation strategies are mentioned. Incorporating a prediction model of the targets dynamics is considered. An interesting simulation is developed using two target models: one "known" model to compute algorithm predictions and a more realistic model to simulate the targets' movement. Target-agent assignments and heterogeneous sensor models are implicitly encoded in the algorithm, although not tested.

In Bakhtari and Benhabib [2], the non-convex problem of maximising the combined visibility of all sensors to target recognition is addressed. They use a finite control horizon to predict, for all possible next targets' and obstacles' placement, where the sensors should be placed to guarantee line-of-sight towards every target (see Figure 3.2). Cameras moving with bounded velocity along fixed lines are considered. The targets and obstacles have a random movement. Regarding the sensors, they use bearing-only sensors with a limited bearing range, i.e., the camera is only able to track the targets inside a pre-defined bearing range and subjected to having line-of-sight towards the target. The level of recognition of a certain target is defined as a function of the quality of the sensing-image, which depends mainly on the camera orientation. Also the minimum number of cameras needed to obtain a certain quality is studied for different scenarios.

In Derenick et al. [17], the performance criterion is defined using graph theory. It seeks to maximise

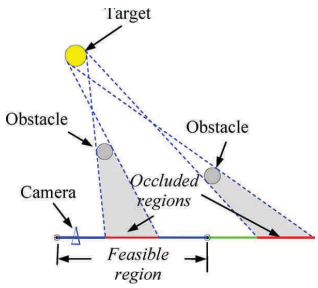

Figure 3.2: Feasible sensor regions towards a target (from Bakhtari and Benhabib [2]).

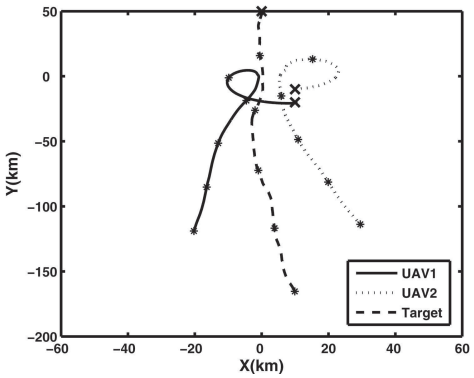

Figure 3.3: Fixed-wing tracking behaviour (from Zhan et al. [54]).

the visibility of observation the targets

$$J(\mathbf{x}) = -\lambda_2(\mathbf{L_V}(\mathbf{x})),$$

where λ_2 the second smallest eigenvalue of the state-dependent Laplacian $\mathbf{L_V}(\mathbf{x})$ of the visibility graph $\mathbf{G_V}(\mathbf{x})$. The problem is non-convex and sequentially linearised as a SDP, subjected to linear matrix inequalities. The constraints ensure full target coverage and minimal network connectivity across the agents. A Second-Order Cone Program (SOCP) relaxation was considered for large teams. The SOCP is easier and more efficient to solve, being therefore more suitable for large teams implementation. Regarding the models, they consider range-and-bearing sensors, single integrator agent dynamics and static or moving targets. The position of the agents and of the targets is assumed to be known with precision (perfect estimation). A constraint where each target has to be followed by at least k agents is implemented, the so-called k-coverage constraint.

In Zhan et al. [54], the metric is defined as L-optimal design performance criterion. The problem is non-convex and NP-hard in general (Zhou and Roumeliotis [58],Zhou and Roumeliotis [59]), thus it is relaxed to a sequential optimisation approach to have an easier tractability. One-step-ahead gradient optimisation is introduced to steer only the heading of the UAVs. Radar sensors (range-only) with Doppler effects, targets moving with constant velocity, and basic 2D fixed-wing dynamics are considered. An EKF is also introduced to estimate the targets' position. Simulations are obtained using the *Dubins* constant-velocity 2D fixed-wing model (see Figure 3.3). Results of the error in the position and velocity of the targets are obtained and the heading control input is analysed.

3.4.3 Distributed single-target tracking

In Martínez and Bullo [33], the global performance criteria is formulated in a D-optimal design fashion. An one-step ahead gradient descent algorithm is implemented, with computed steepest descent rules. Only range sensors are mentioned in this work. 2D single integrator dynamics are considered to move the set of agents. However, the agents movement is considered to be constrained to the static boundary $\partial\mathcal{Q}$ of a compact convex set $\mathcal{Q}$ (see Figure 3.4). The agents' position is assumed to be known with precision. The targets are moving, and an EKF is used to compute the targets' position estimation. However, for this estimation, a centralised fusion centre is needed. The control input is computed by each agent in a distributed way based on detection of the clockwise and counter-clockwise agents along the allowed trajectory $\partial\mathcal{Q}$ (see Figure 3.5). Despite having good results without requiring much communication, it constraints the agents movement to a one dimensional translation on a static boundary.

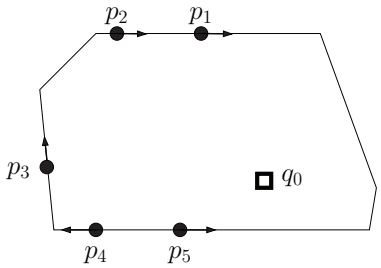

Figure 3.4: Sensors placed along a static boundary surrounding the target (from Martínez and Bullo [33]).

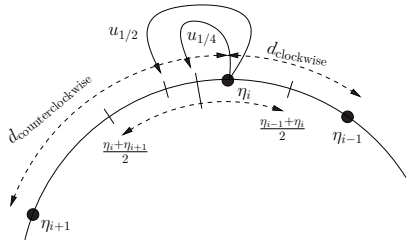

Figure 3.5: Computing next position communicating with the direct neighbours (from Martínez and Bullo [33]).

In Zhou and Roumeliotis [56], the performance criterion is based on the Shannon information

$$J(p(\mathbf{z})) = \text{tr}(p(\mathbf{z})),$$

where tr(.) is the trace of a given matrix, which is in general NP-hard (Zhou and Roumeliotis [58]).It is reformulated to the minimisation of the maximum eigenvalue

$$J(p(\mathbf{z})) = \lambda_1(p(\mathbf{z})),$$

which is identical to minimising the 2-norm of the sum of a given set of 2D vectors. For this particular problem the two criteria result in the same solution (Zhou and Roumeliotis [57]). An one-step ahead gradient descent algorithm is implemented and a novel adaptive relaxation algorithm is proposed. This algorithm is shown to have identical results as an exhaustive search-based algorithm. A planning path is assumed over a certain time-horizon, in spite of considering a one-step-ahead steepest descent cost-function. Regarding the sensors, range-only sensors are used. Single integrator dynamics in a 2D environment are considered, assuming perfect observation of the position of the agents. The targets are moving and a centralised fusion centre is needed for implementing the EKF to estimate the targets' position. The relaxation algorithm proposed smooths the cost function avoiding local minima while reducing the problem complexity, from exponential with the number of agents, to linear. This implementation still needs a centralised unit to be successfully implemented.

In Beard et al. [4], the convex squared distance towards the targets is minimised

$$J(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{q=1}^{m} ||\mathbf{t}^q_{\text{pos}} - \mathbf{x}^i_{\text{pos}}||^2,$$

where $\mathbf{x}^i_{\text{pos}}$ is the position of the agent i and $\mathbf{t}^q_{\text{pos}}$ is the position of the target q. The goal is to obtain the best image as possible, spaced with some desired fixed time-intervals. An one-step ahead gradient descent algorithm is implemented. Image sensing is modelled as a function of the relative target-agent distance. A fixed-wing path planning is considered on real UAVs. A minimal forward motion constraint was considered. The targets are assumed to be static, and no estimation of their position is considered. Flight-test results were obtained with fixed-wing UAVs flying at a predefined altitude. Also, consensus communication (Olfati-Saber et al. [42], Yang and Johansson [53]) algorithms are implemented to ensure the right communication needed for distributing the solution. Here, we can say that no continuous tracking is implemented, the agents still have to follow a predefined set of way-points.

In Mathews and Durrant-Whyte [34], an *entropy* fashion performance criterion is used. They minimise the entropy over a given finite planning horizon. Steepest descent rules are used to perform the optimisation. Regarding all the models, for the sensors bearing-only models are used, single integrator dynamics considered for the agents and static targets are assumed. The position of the agents is known at all times and the targets estimation is implemented with an EKF. However, the EKF implementation makes use of a centralised fusion centre. The *Partial Separability* concept is introduced which allows the local algorithms to avoid having full models of the other agents.

In Olfati-Saber [41], a potential function

$$U_\lambda = U(q) + \lambda I_\lambda$$

is used as the performance criterion. This cost-function has two main components: a structural potential $U(q)$ which seeks to achieve a given flocking formation and an information value component I_λ given by a A-optimal design fashion. An one-step ahead gradient descent algorithm is implemented with local constant-step descent update rules. The sensors are modelled as range-and-bearing sensors. 2^{nd} order 2D vehicle dynamics are mentioned, and the targets are moving. A Distributed Kalman Filter is implemented in each agent to update the targets' estimation. Perfect observation on the agents' states is assumed. The author presents a joint tracking- and flocking-based behaviour, joining together two different concepts mention in the literature.

3.4.4 Distributed multi-target tracking

In Parker [44] and Parker and Emmons [46], The authors assume that sensing a target with one agent only is enough to track it. The problem proposed is to maximise time during which each target is begin monitored by at least one sensor-agent. They implement a finite predictive horizon of N-steps with a steepest descent update rule and a performance criterion as follows.

$$J(\mathbf{x}) = \sum_{t=k+1}^{k+N} \sum_{q=1}^{m} \bigvee_{i=1}^{n} b_t^{i,q},$$

where $b_t^{i,q}$ is a boolean function defined as

$$b_t^{i,q} = \begin{cases} 1, & \text{if the agent } \mathbf{x}^i \text{ is observing the target } \mathbf{t}_{\text{pos}} \text{ at time } t, \\ 0, & \text{otherwise.} \end{cases}$$

Since this problem is proved to be NP-hard, a relaxation using force fields is considered. The concept of force fields is based on sources emanating forces that can attract or repel the agents. In this case, the targets origin a attractive force field around them and the agents origin an repelling one. On the one hand, if a given agent is placed near a certain target q, there is a force trying to bring this agent closer

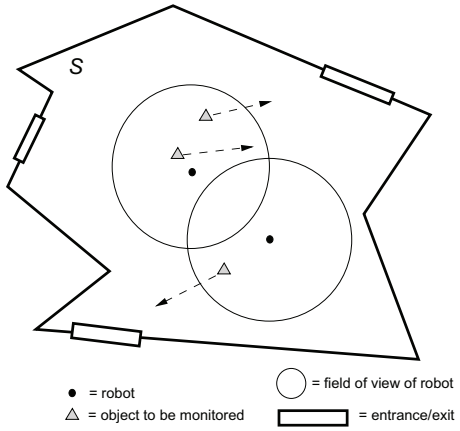

Figure 3.6: Simulation scenario with entries and exits (from Parker [44]).

to target q. On the other hand, if this same agent is close to another agent, there exists a force trying to separate them. This force fields concepts is significantly easier to implement but it does not ensure to find an optimal solution. Also, stability and convergence analysis is difficult to study using this setup. Regarding the sensors, omnidirectional cameras are considered, with a limited sensing range. The agents are modelled as a single integrator with bounded velocity. Their position is assumed to be measured with precision. The targets are moving and no estimation strategy is used. All the agents share the same global coordinates system and a limited communication range is considered. Also, the number of targets is assumed to change with time. A finite set of enter and exit areas is considered, from where targets can entry or exit the simulation, as in Figure 3.6.

In Parker [45], the maximisation of the following metric is considered

$$J(\mathbf{x}) = \sum_{t=k+1}^{k+N} \sum_{q=1}^{m} \frac{g(B_t, q)}{N},$$

where $g(B_t, q)$ is given by

$$g(B_t, q) = \begin{cases} 1, & \text{if there exists an } i \text{ such that } b_t^{i,q} = 1, \\ 0, & \text{otherwise.} \end{cases}$$

They use an one-step-ahead local descent update rules with constant step size. Important to notice that perfect communication is considered between all the agents. The sensors are, once more, omnidirectional cameras with limited sensing range. Single integrator agents with bounded velocity are considered. The position of the agents is assumed to be perfectly observed. The targets do move, however no estimation strategy is implemented. The goal is to determine where the sensors should be located to maintain all the targets in view. This algorithm is implemented in a distributed way, considering a limited communication range where each agent can communicate only with the nearby agents. This communication range is considered to be greater than the sensing range. The ratio between the targets and agents is considered to be around 1/2. Here, also the same coordinated system is assumed among the different agents which facilitates its distribution.

In Matsuyama and Ukita [35], the goal is to guarantee an image of all the targets regardless their movement. They implement a finite predictive horizon of N-steps with a steepest descent update rule. Their scenario considers tilting cameras with a limited bearing range. These cameras are in fixed positions, i.e., the agents do not move. Therefore, the goal is to head the cameras such that a single image towards

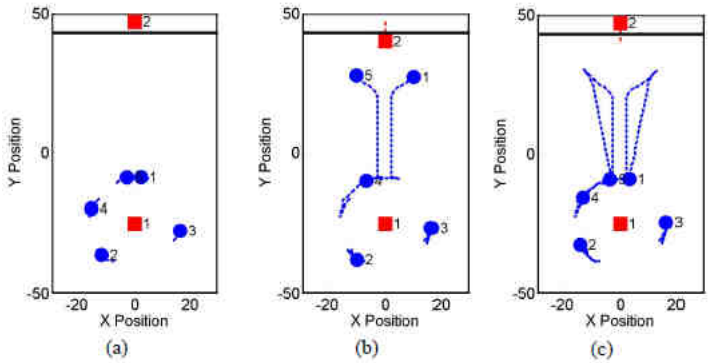

Figure 3.7: RoboFlag patrolling example (from Chung et al. [13]).

all the targets is obtained. The targets are moving in a given space. No estimation strategies are used as the targets' position is assumed to be known. Simulations of cooperative distributed vision in a room are considered.

In Grocholsky et al. [22], the performance criterion is based on the Shannon information. They also implement a Receding Horizon Controller of N-steps with a steepest descent update rule. The sensors considered are bearing-only. Non-linear 2D UAV dynamics with linear constraints are considered. A set of static target-features is considered and this work uses a Decentralised Data Fusion strategy for their estimation.

In Chung et al. [13], the problem is formulated using D-optimal design. The authors use one-step ahead local descent update rules, with constant step size and considering perfect communication. Regarding the models, they consider range-only sensors, single integrator agent dynamics with perfect observation and random-walk moving targets. The targets' position is estimated with a decentralised EKF based on perfect communication between agents. Agent-target collisions are avoided considering a sweet spot in the range-sensor. The performance of the decentralised gradient descent implementation is compared with a centralised exhaustive search. An interesting scenario is described in the context of the *RoboFlag* problem: five agents seek to track one single target when another target arrives, then the agents split to sense both targets and they gather together again when the second target disappears (Figure 3.7). Note that here the problem of agent-target assignment is addressed and solved implicitly.

Tang and Ozguner [51] is one of the few papers considering specifically a limited resources scenario, i.e., a number of agents lower than the number of targets $n \ll m$. Based on the fact that the uncertainty of a target status is proportional to how frequently the target track is updated, the performance criterion is to minimise the average time duration between two consecutive observations for all targets. The motion-planning is stated as a NP-hard optimisation-based problem solved with a gradient-based method. A suboptimal approach is proposed to reduce largely the computational load. The sensors considered are cameras which are able to sense any target moving on the ground plane inside their "footprint"-sensing area. The *Dubins* car model is agents model. Perfect observers are considered, i.e., the state of the agents is always known. The targets move randomly without any estimation strategy. The motion planning consists in two stages: task decomposition (tracking the whole set of targets is divided into disjoint task assignments based on K-mean clustering developed in Bishop [6]) and individual path generation using a heuristic method (Murty [38]). However, the authors consider the agents to move much faster than the targets, which avoids constant updates on the task decomposition.

In Simonetto and Keviczky [48], the performance criterion is given according to graph theory, defined as the joint maximisation of connectivity and visibility of the considered Graph. Even for a single target, the resulting problem is non-linear and NP-hard. Due to the non-convexity of the cost-function, it is

relaxed using 1^{st} order Taylor expansions to obtain a convex problem. They are interested only in the network topology, then no sensors are considered. Single integrator vehicle dynamics are considered, and the 2D vehicle positions is assumed to be known. The targets are moving with a random walk behaviour. Their velocity is bounded, therefore their future position is assumed to lay in a circle which the centre is the current position. The performance criterion considers all the possible future targets' position and computes the agents' position that maximises the connectivity for the worst case. Thus, this problem definition can be seen as a robust maximisation problem. Local SDPs are derived from the global problem and solved by each agent with the use of local communication. The distributed algorithm is simulated and compared with the centralised behaviour, fulfilling the global constraints of minimal connectivity and visibility. Both centralised and distributed behaviours are presented and the solutions of both formulations are compared.

3.4.5 Other related fields

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. A possible scenario is the estimation of a fire perimeter, updating frequently estimations on the fire progress (Casbeer et al. [12]). A group of UAVs is used to monitor and track the propagation of large forest fires in order to supply quick information to the fire fighters. The goal is to minimise the information latency, i.e., the performance criterion is the time-delay between sensing the environment and transmitting the information to a ground station. A decentralised approach is followed considering range communication limitations. In order to accomplish agents refuel, a cooperation algorithm is developed to deal with a dynamically changing number of agents. In order to deal with dynamic fire, the algorithm also deals with expansions and contractions of the fire perimeter. Monte-Carlo simulations are implemented using a 6DOF UAV model flying around the perimeter of a high fidelity fire model (the EMBYR model Hargrove et al. [23]). These simulations assume limited UAV fuel, highly dynamic fire and a even number of UAVs.

Another field detection case is the parameter estimation of a certain distributed system defined in a spatial domain (Uciński [52]). The problem is to obtain the optimal sensors position to accurately estimate a set of parameters of a continuous system. The dynamics of the vehicles and various geometric constraints on their movement are taking into account. The cost function is defined in a centralised way, using a Fisher Information Matrix. Four optimality criteria are defined, being one of them the D-optimal design. Three types of generic vehicles dynamics are treated. Sensor clusterisation (different sensors tending to the same point), collision avoidance and vehicles limitations are defined as problem constraints. The solution is obtained numerically through iteratively linearisation and optimisation.

Formation control is another research topic relating UAVs and distributed control. A decentralised receding horizon scheme to achieve coordination among agents is described in Keviczky et al. [27] and Borrelli et al. [7]. In these papers, the main focus is on the high-level control and coordination of autonomous vehicle teams to keep a certain formation. A graph structure is used to describe the communication topology between the agents. On each vehicle, a RHC performance criterion is implemented and information from the direct-neighbours is used to predict their position. The graph structure and the decentralised RHC scheme are described in detail. This procedure assumes the agents to be modelled by MIMO piecewise linear models and it implements low computational effort controllers. Collision avoidance is always ensured, even when the local RHC problems become infeasible. The final algorithm is proved to be easily implemented in hardware platforms and run in real-time. Stability analysis on the receding horizon controllers for decoupled/distributed systems can be found in Keviczky et al. [26].

The field of area coverage is motivated by environmental monitoring applications, where the goal is

to cover a given set of interest. An area coverage problem is formulated in Howard et al. [25]. Giving an area of interest, the goal is to maximise the area covered by the sensors network, while minimising the time spent to deploy this network. The solution is deployed using a potential field approach, constituting a distributed and highly scalable solution. Therefore, the agents can react fast to changes on the environment area or changes on the number of sensors. The use of potential fields is mainly motivated by highly dynamic environments and low computational power. This approach is easy and fast to implement, in order to compute the path planning in real-time.

Distributed control to solve coverage problems is mentioned in Bullo et al. [11]. A space-partitioning problem for groups of autonomous mobile agents is suggested: the authors seek to find the optimal partition with respect with a multi-centre performance criterion. A gossip coverage algorithm is proposed to solve this territory partitioning problem, based on local asynchronous unreliable communication. This coverage algorithm is based on the "centring and partitioning" algorithm, first introduced by the classic methods of Lloyd (Lloyd [30]). Convergence properties of this gossip algorithm to the optimal centroidal Voronoi partitions (Du et al. [19]) are discussed in detail.

Ad hoc communication networks is a researching field that creates and maintains a network of mobile agents connected by wireless. The problem proposed seek to relay data from the exploring robots towards ground-stations, and vice-versa. In Simonetto et al. [49] and references therein, heuristic approaches are formulated regarding the network connectivity of these networks. A fixed ground-station, so-called *hub*, a set of n agents and m targets, are defined in Simonetto et al. [49] where an ad hoc communication network problem is posed considering a limited resources scenario $n \ll m$. The authors maximise the connectivity of the communication network over time of the shortest-path case for each target. The connectivity between two sensors, or a sensor and a target, is constrained to line-of-sight between them. The problem is solved using dynamic potential fields, where the current overall needs of the team dynamically change the applicable potential fields.

3.5 Considerations

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 the metric to be minimised in Beard et al. [4]; and the assumption that sensing a target with one agent only is enough to track it is also considered by the authors Parker [44] and Parker and Emmons [46].

Tang and Ozguner [51] also considers a limited resources scenario, where the proposed NP-hard problem is to guide a set of agents to minimise the mean of the Average Time Duration between two consecutive observations of each target. However, the authors assume that the agents move much faster than the targets, thus 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. It is largely used in the literature as a simple kinematic model of fixed-wing UAVs moving in a given plane (Zhan et al. [54] and Tang and Ozguner [51], to name a few). Good analysis on the stability and robustness of the time-optimal trajectory generation for *Dubins* car model can be found in Balluchi et al. [3].

Regarding the observation and estimation models, most of the mentioned literature (Derenick et al. [17], Tang and Ozguner [51], Zhou and Roumeliotis [56], Mathews and Durrant-Whyte [34], Parker [45] and Olfati-Saber [41]) also assume perfect observation of the agents' states (Assumption 1). The position of the targets is also assumed to be known with precision (Derenick et al. [17], Parker [45], Parker [44] and Parker and Emmons [46]), motivating our initial noiseless estimation assumption within a given sensing range (2.16).

Chapter 4

Naive centralised solution

After formulating the problem (Chapter 2), we first consider a so-called naive centralised approach. This proof-of-concept approach help us concluding whether the distributed-algorithms we propose later meet our goal: to obtain the single best estimation of all targets.

In this approach we assume the following

- *a centralised implementation*: the problem is formulated in a centralised optimisation scheme where a central entity is required to do all the computations for all agents;
- *the original performance criterion*: we implement the original performance criterion (Definition 3) which represents a non-convex cost-function which is in general difficult to solve;
- *no maximum sensors' range*: unlimited sensors range $R^i = \infty, \;\; i = 1, ..., n$, therefore the full target coverage constraint is always guaranteed.

4.1 Problem Formulations

Considering separately the two agent models, two optimisation-based problems can be formulated: the quadrotors and fixed-wings centralised problem. Using the quadrotor model (2.9,2.10), the agents' state is given by the quadrotors' position and velocity, and the problem is constrained to linear equality and inequality constraints. Using the fixed-wing model (2.11,2.12), the position and attitude of the fixed-wings define the agents' state, and the problem has not only linear equality and inequality constraints, but also non-linear equality constraints due to the non-holonomic dynamics.

Problem 2 (Naive centralized tracking for Quadrotors)**.**

Given the optimisation variables $\mathbf{x}^i_{t+1} = [\mathbf{x}^i_{t+1,\text{pos}}, \mathbf{x}^i_{t+1,\text{vel}}]^\top, \mathbf{u}^i_t, \;\; i = \{1, ..., n\}, \;\; t = \{k, ..., k + N - 1\}$ *stacked as*

$$\mathbf{o} = \begin{bmatrix} \mathbf{u}^i_t \\ \mathbf{x}^i_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}^i_t \\ \mathbf{x}^i_{t+1,\text{pos}} \\ \mathbf{x}^i_{t+1,\text{vel}} \end{bmatrix}, \;\; i = \{1, ..., n\}, \;\; t = \{k, ..., k + N - 1\}, \tag{4.1}$$

where $\mathbf{x}^i_{\text{vel}} = [v^i_E, v^i_N, v^i_h]^\top$ *and* $\mathbf{u}^i = [f^i_E, f^i_N, f^i_h]^\top$*, and the cost function*

$$\begin{aligned}\bar{J}(\mathbf{o}) := \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left(\min_{i\in\{1,\dots,n\}} \{||\mathbf{x}^i_{t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right) + \rho \sum_{i=1}^{n} ||\mathbf{u}^i_t||^2 \right] + \\ + \varphi \sum_{q=1}^{m} \left(\min_{i\in\{1,\dots,n\}} \{||\mathbf{x}^i_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right),\end{aligned} \tag{4.2}$$

we define the problem

$$\min_{\mathbf{o}} \bar{J}(\mathbf{o}) \text{ s.t.}$$
$$\begin{cases} \begin{bmatrix} -\mathbf{B} & \mathbf{I}_6 \end{bmatrix} \begin{bmatrix} \mathbf{u}^i_t \\ \mathbf{x}^i_{t+1} \end{bmatrix} = \mathbf{A}\mathbf{x}^i_t, \;\; t = k \\ \begin{bmatrix} -\mathbf{A} & -\mathbf{B} & \mathbf{I}_6 \end{bmatrix} \begin{bmatrix} \mathbf{x}^i_t \\ \mathbf{u}^i_t \\ \mathbf{x}^i_{t+1} \end{bmatrix} = \mathbf{0}_{6\times 1}, \;\; t = \{k+1, ..., k+N-1\} \\ h^i_t \geq h_{\min}, \;\; t = \{k+1, ..., k+N\} \\ -v_{\max} \leq v^i_{t,E}, v^i_{t,N}, v^i_{t,h} \leq v_{\max}, \;\; t = \{k+1, ..., k+N\} \\ -f_{\max} \leq f^i_{t,E}, f^i_{t,N}, f^i_{t,h} \leq f_{\max}, \;\; t = \{k+1, ..., k+N\} \end{cases}, \;\; i = \{1, ..., n\}, \tag{4.3}$$

where, the matrices $\mathbf{A}, \mathbf{B}$ *are defined in* (2.9).

The parameters are

N	*predictive horizon*
n, m	*number of agents, targets*
ρ, φ	*performance weighting factors*
$h_{\min}, v_{\max}, f_{\max}$	*quadrotors' specifications*
$W_{\max}$	*targets' maximum velocity*
Δt	*sampling period.*

Problem 3 (Naive centralized tracking for Fixed-wings)**.**

Given the optimisation variables $\mathbf{x}^i_{t+1} = [E^i_{t+1}, N^i_{t+1}, \theta^i_{t+1}]^\top, \mathbf{u}^i_t = u^i_t, \;\; i = \{1, ..., n\}, \;\; t = \{k, ..., k+N-1\}$ *stacked as*

$$\mathbf{o} = \begin{bmatrix} \mathbf{u}^i_t \\ \mathbf{x}^i_{t+1} \end{bmatrix} = \begin{bmatrix} u^i_t \\ E^i_{t+1} \\ N^i_{t+1} \\ \theta^i_{t+1} \end{bmatrix}, \;\; i = \{1, ..., n\}, \;\; t = \{k, ..., k+N-1\}, \tag{4.4}$$

where $\mathbf{x}^i_{\text{pos}} = [E^i_{t+1}, N^i_{t+1}, \bar{h}]^\top$*, and the cost function*

$$\begin{aligned}\bar{J}(\mathbf{o}) := \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left(\min_{i\in\{1,\dots,n\}} \{||\mathbf{x}^i_{t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right) + \rho \sum_{i=1}^{n} |u^i_t|^2 \right] + \\ + \varphi \sum_{q=1}^{m} \left(\min_{i\in\{1,\dots,n\}} \{||\mathbf{x}^i_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2\} \right),\end{aligned} \tag{4.5}$$

we define the problem

$$\min_{\mathbf{o}} \bar{J}(\mathbf{o}) \text{ s.t.}$$

$$\begin{cases} \begin{bmatrix} -\Delta t & 1 \end{bmatrix} \begin{bmatrix} u_t^i \\ \theta_{t+1}^i \end{bmatrix} = \theta_t^i, \quad t = k \\ \begin{bmatrix} -1 & -\Delta t & 1 \end{bmatrix} \begin{bmatrix} \theta_t^i \\ u_t^i \\ \theta_{t+1}^i \end{bmatrix} = 0, \quad t = \{k+1, ..., k+N-1\} \\ \begin{bmatrix} E_{t+1}^i \\ N_{t+1}^i \end{bmatrix} - \begin{bmatrix} E_t^i \\ N_t^i \end{bmatrix} = \begin{bmatrix} V_M \Delta t \cos\theta_t^i \\ V_M \Delta t \sin\theta_t^i \end{bmatrix}, \quad t = \{k, ..., k+N-1\} \\ -u_{\max} \leq u_t^i \leq u_{\max}, \quad t = \{k+1, ..., k+N\} \end{cases}, \quad i = \{1, ..., n\}. \tag{4.6}$$

The parameters are

N	*predictive horizon*
n, m	*number of agents, targets*
ρ, φ	*performance weighting factors*
$\bar{h}, V_M, u_{\max}$	*fixed-wings' specifications*
$W_{\max}$	*targets' maximum velocity*
Δt	*sampling period.*

Note that, since this is a naive approach is only a proof-of-concept, the full target coverage constraint is not considered. I.e., an unlimited sensing range is assumed $R^i = \infty, \quad i = \{1, ..., n\}$.

Remark 2 (Control Variables)**.**

Considering the naive centralised problems 2 and 3, the actual control variables are set of inputs $\mathbf{u}_t^i, i = \{1, ..., n\}$, $t = \{k, ..., k+N-1\}$. *I.e., the future agents states* $\mathbf{x}_{t+1}$ *are obtained using the agents model, knowing the current state* $\mathbf{x}_k$ *and the future inputs* $\mathbf{u}_t$. *However, to simplify the problem and write the cost as a function of the future agents states, we leave the dependency on the vehicle dynamics implicitly in the constraints* (4.3) *and* (4.6) *Therefore, we add the future agents states considering the optimisation variables* $[\mathbf{u}_t^i, \mathbf{x}_{t+1}^i]^\top, i = \{1, ..., n\}, t = \{k, ..., k+N-1\}$.

After defining the problem mathematically, it is important to label the type of optimisation problem before proceeding. Knowing the problem we are dealing with helps us analysing which optimisation strategy to use. A key point is to define whether we are dealing with a convex or a non-convex problem. A convex optimisation problem is defined with a convex real-valued function, subjected to convex inequality constraints and affine equality constraints, restricting the problem to a convex set.

Analysing the performance criteria (4.2) and (4.5), we can directly see that apart from the point-wise minima function "min", the other functions are Euclidean norms. Euclidean norms are convex functions and the sum of convex functions is also a convex function. The point-wise minimum function is shown in Figure 4.1, and defined as follows.

$$f(x, y) = \min\{x, y\}.$$

From Figure 4.1, one can see that the current function is not a convex function, in fact it is a concave function. Therefore, we conclude that both naive centralised problems 2 and 3 are non-convex problems. Non-convex problems are difficult to solve, and in general, compromise between a very long computational

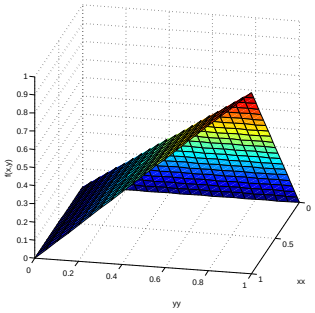

Figure 4.1: Point-wise minimum function.

time and finding the global optimum has to be made. We implemented the two problems with the `MatLab` tool `fmincon`, which applies an interior point algorithm to solve non-convex problems.

4.2 Simulation results

Static targets

First a simple static targets scenario is considered: eight targets $m = 8$, statically placed inside the given space $\mathbf{t}^q_{\text{pos}} \in S$, forming two groups of four targets in a square shape. The goal is to check with a basic static example, where the final optimal placement can be directly seen, whether the presented centralised approach can achieve the optimal position. Two quadrotors are considered, placed without velocity at random initial positions inside the convex space S and meeting the constraint $h^i \geq h_{\min}$. The optimisation parameters used are presented in Table 4.1.

Parameter	Value	Description
N	3	finite horizon
n, m	2, 8	number of agents, targets
ρ, ϕ	0.01, 0.01	performance weighting factors
$h_{\min}, v_{\max}, f_{\max}$	10, 2, 2	quadrotors' specifications
Δt	1	time-step (in seconds)
k	60	number of time-steps of the simulation

Table 4.1: Static targets using quadrotors

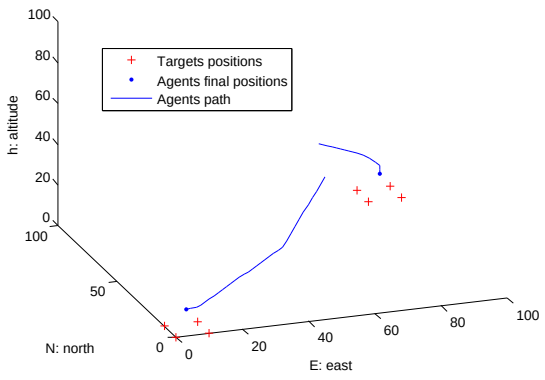

Figure 4.2: Static targets Agents' Movement.

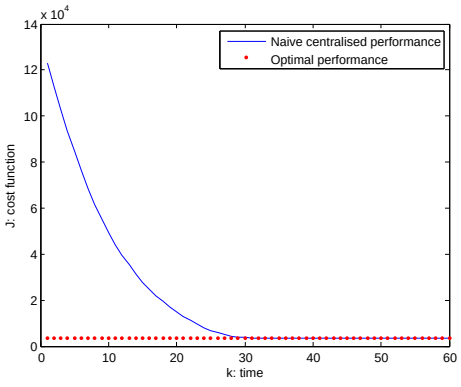

Figure 4.3: Static targets performance.

The agents' movement of a single simulation during 60 time-steps is presented in Figure 4.2, and the correspondent performance in Figure 4.3. The desired optimal placement, single best estimation for all

targets, can be directly seen due to the placement of the targets. The global optimum is obtained when each quadrotor is placed in the centre of each square group of targets, with the minimum altitude and null velocity. The agents converge to the desired optimum at steady-state.

Therefore, this naive centralized approach arrives to the expected behaviour in this static case, and the optimal performance is reached.

Moving targets

Then a moving targets scenario is considered using quadrotors and fixed-wings. Using 10 agents to track 100 random-walk targets, the parameters list is presented in Table 4.2.

Parameter	Value	Description
N	3	finite horizon
n, m	10, 100	number of agents, targets
ρ, ϕ	0.01, 0.01	performance weighting factors
$h_{\min}, v_{\max}, f_{\max}$	10, 2, 2	quadrotors' specifications
$\bar{h}, V_M, u_{\max}$	10, 2, 0.8	fixed-wings' specifications
$W_{\max}$	3	targets' maximum velocity
Δt	1	time-step (in seconds)
k	60	number of time-steps of the simulation

Table 4.2: Moving targets using quadrotors or fixed-wings

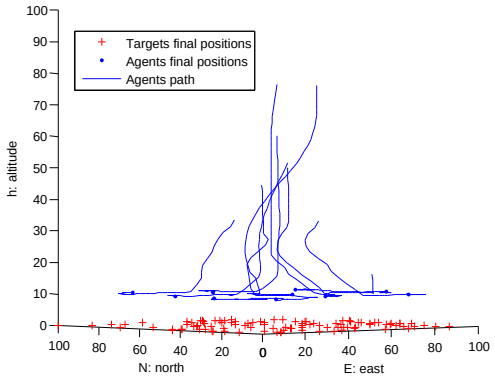

Figure 4.4: Quadrotors' descent.

For the quadrotors case, we can see the agents moving towards the minimum altitude $h^i = h_{\min}$ (Figure 4.4) while for the fixed-wings case, they fly at a fixed altitude of $h^i = \bar{h}$. The agents' movement during the simulation is presented for the quadrotors and fixed-wings in Figures 4.5, 4.6 respectively. For the fixed-wing case, we can see the expected flying-dynamics without steep turns.

The performance criteria for both cases is shown in Figure 4.7. First, in a transitory stage, the agents move towards matching the different groups of targets. For the quadrotors' case, the performance criterion starts with a significantly higher value due to initial random altitudes $h_0^i > 10, \;\; i = 1, ..., n$, instead of $h_0^i = 10, \;\; i = 1, ..., n$, for the fixed-wings' case. Then, the performance criteria stabilises and, due to the targets' movement, oscillates around a "stationary" value.

Comparing both agents' dynamics, after the transitory phase, the performance criterion stabilises around $\bar{J}_{\text{quad}} = 0.73 \times 10^5$ and $\bar{J}_{\text{wing}} = 0.81 \times 10^5$. The fixed-wing model has a decreasing in performance around 11% when compared with the quadrotors. This decrease is motivated by having the turns constrained to smooth movements.

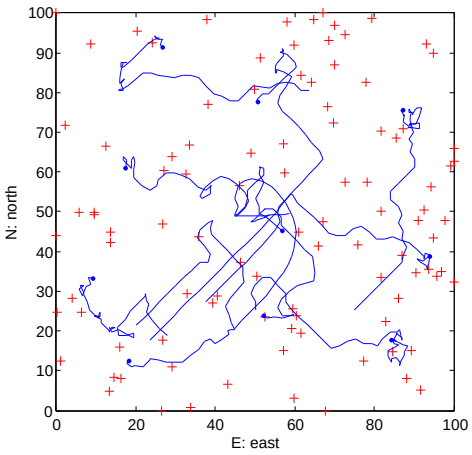

Figure 4.5: Moving targets using Quadrotors (see legend of Figure 4.4).

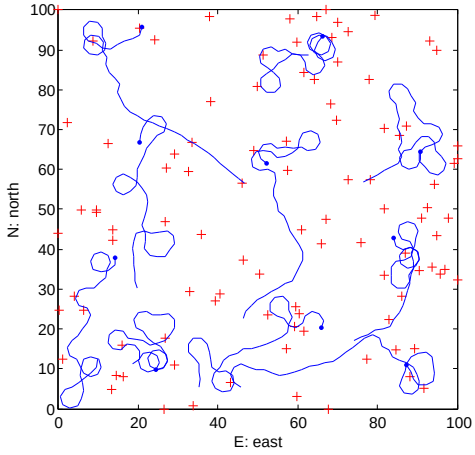

Figure 4.6: Moving targets using Fixed-wings (see legend of Figure 4.4).

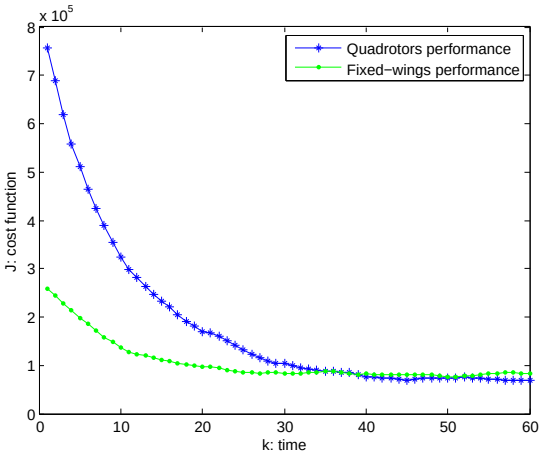

Figure 4.7: Moving targets performance.

Computational Time

The naive centralised moving targets simulations took 678 seconds per time-step for the quadrotors and 117 seconds per time-step for the fixed-wings. The time difference is motivated by the size of the optimisation variables, 9 per agent per horizon-step for the quadrotors and 4 per agent per horizon-step for the fixed-wings. Also, the quadrotors can move along a 3D space (E, N, h), in contrast with the fixed-wings which are constrained to a fixed altitude $\bar{h}$.

Regarding the time complexity with the number of agents, the `MatLab` tool `fmincon` uses a Sequential Quadratic Programming (SQP) method. The number of agents increase the number of optimisation variables by $9N$ for the quadrotors problem and by $4N$ for the fixed-wings problem. Simulation results comparing the computational time of a single time-step with different number of quadrotors and fixed-wings can be seen in Figure 4.8.

4.3 Final outlook

This naive centralised approach took minutes to compute the control inputs for a single time-step using ten agents. Two reasons motivate these time results: being a centralised algorithm and being a non-convex optimisation. In a centralised implementation, the number of optimisation variables increase

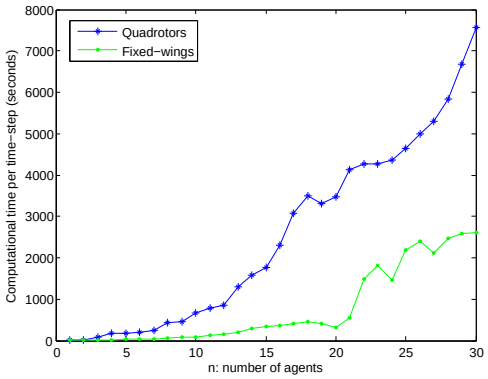

Figure 4.8: Computational time of a single time-step with the number of agents.

largely with the number of agents. In our case, the number of optimisation variables is $9N$ per quadrotor and $4N$ per fixed-wing. 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 far from feasible for real implementations.

Regarding optimality and feasibility, we conclude that achieving the global optimum is not our first priority. Considering moving targets, the global optimum is constantly changing due to the highly dynamic changing environment. Therefore, we are far more interested in obtaining fast and feasible algorithms. The UAVs should be guided to decrease our cost-function towards a local optimum along feasible solutions, rather than guaranteeing to find the global minimum in the long term.

In order to face the results obtained with this naive centralised implementation, we aim to

- relax the problem formulation to obtain a convex optimisation which it is easier and faster to solve: we aim to achieve an implementation that runs in less than one second per time-step;
- distribute the problem over the agents: to reduce the increase of time complexity with the increase of the number of agents;
- introduce a sensors' range limitation and ensure full target coverage.

Chapter 5

A novel two-layer approach

The proposed tracking problem (Chapter 2) was formulated using a centralised approach (Chapter 4). This centralised implementation meets the desired behaviour but does not fulfil our requirements in term of computational load and time complexity. In this chapter, we propose a relaxed approach to implement the problem in a distributed scheme via convex optimisation-based algorithms.

Single-agent case

First, considering the single-agent case, the performance criterion (2.8) becomes

$$\bar{J}_k = \sum_{t=0}^{N-1} \left[\sum_{q=1}^{m} \left(||\widehat{\mathbf{x}}_{k+t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \right) + \rho ||\mathbf{u}_{k+t}||^2 \right] + \varphi \sum_{q=1}^{m} \left(||\widehat{\mathbf{x}}_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \right). \tag{5.1}$$

Note that, the point-wise minimum function is not present. This performance criterion is a convex function and the problem can be formulated as a convex optimisation. Also, the problem is inherently local, i.e., there is only one agent involved.

Therefore, we can conclude that the current formulation for the single-agent case already fulfils our goals: a convex optimisation and a distributed scheme. This result motivates the use of this "single-agent metric" (5.1) also for the multi-agents case. Considering each agent as a separated tracking problem suggests that we might achieve convexity and distribution for our problem.

Global picture

Considering the multi-agents case, the performance criterion

$$\begin{aligned} \bar{J}_k(\mathbf{x}_k, \mathbf{u}_k, ..., \mathbf{u}_{k+N-1}) := \sum_{t=0}^{N-1} \left[\sum_{q=1}^{m} \left(\min_{i \in \{1,...,n\}} \{ ||\widehat{\mathbf{x}}^i_{k+t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \} \right) + \rho \sum_{i=1}^{n} ||\mathbf{u}^i_{k+t}||^2 \right] + \\ + \varphi \sum_{q=1}^{m} \left(\min_{i \in \{1,...,n\}} \{ ||\widehat{\mathbf{x}}^i_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \} \right) \end{aligned} \tag{5.2}$$

has the point-wise minimum which plays an important role in this problem: it computes for each target which is the closer agent. This role is called in the literature the *agent-target assignment* as it assigns for each target which is the agent to consider, it formulates the global picture comparing the positions of the all the agents with respect to each target. In fact, the point-wise minimum function does not implement the assignment separately, the assignment and the tracking problems are coupled in (5.2).

If we were considering the agents' and the targets' positions to be static, then the "min" function would simply compute the agent-target assignment. However, the vehicles' positions change along the predictive horizon, changing the agent-target assignment together with the vehicle dynamics.

If the agent-target assignment was solved *a priori*, the targets would be already divided among the agents and we could apply (5.1). Unfortunately, this is not the case, our proposed formulation has a centralised fashion (5.2) which couples the assignment and the tracking problems.

Relaxation

In the current formulation, the point-wise minimum function solves the agent-target assignment in a centralised way and 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. Thus, our assignment and tracking problems are fully coupled in (5.2). We relax the current formulation to split the Assignment Problem from the Tracking Problem (see Figure 5.1). In the relaxed form, first we solve the agent-target assignment beforehand at each time-step. Then, we steer the agents using the local tracking performance criterion (5.1). This relaxation allows us to meet one of our goals: to convexify the tracking problem.

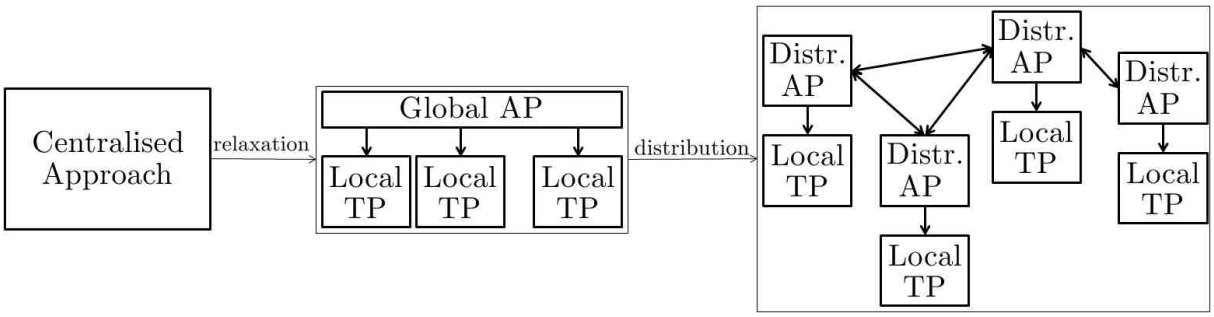

Figure 5.1: Problem relaxation and distribution.

The centralised approach assigns each target to the closer agent placed at $\mathbf{x}^i_{k+t,\text{pos}}$ along the predictive horizon $t = \{1, ..., N\}$. Thus, the assignment changes during the optimisation along the predictive horizon, depending on the future predicted positions of the agents $\mathbf{x}_{k+t,\text{pos}}$.

In our relaxation, in order to compute the Assignment Problem beforehand, we will not compute the assignment along the predictive horizon. Instead, we will compute the assignment only once for each time-step k, and use it along the predictive horizon of the optimisation $k + t$, $t = 1, ..., N$. Therefore, our relaxation will consider only a single future position for each agent denoted by $\mathbf{x}^i_{\text{F,pos}}$ (Figure 5.2).

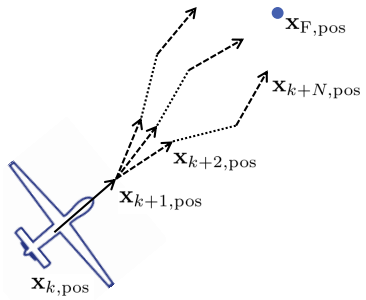

Figure 5.2: Single future position computed beforehand.

The single future agents position $\mathbf{x}_{\text{F,pos}}$ is the variable that we use to divide the space among the

agents, i.e., to solve the Assignment Problem. The global assignment problem is defined as a function of $\mathbf{x}_{\mathrm{F,pos}}$ as follows.

Problem 4 (Global assignment problem)**.**

Given the set of n agents, the set of m targets, and a set of future positions for each agent $\mathbf{x}^i_{\mathrm{F,pos}}$*, we seek to assign each target* $q = \{1, ..., m\}$ *to the closer agent* $\{a^1, ..., a^m\}$*, by computing*

$$a^q = \underset{i \in \{1,...,n\}}{\arg\min} \{||\mathbf{x}^i_{\mathrm{F,pos}} - \mathbf{t}^q_{\mathrm{pos}}||^2\}. \tag{5.3}$$

In Chapter 6, motivating by the fact that the local tracking problem steers the agents towards the targets centre-of-mass, we define this future agents position as the targets' centre-of-mass of a given partition.

With this definition for $\mathbf{x}_{\mathrm{F,pos}}$, we guarantee that the steady-state solution of the original and the relaxed problems are equivalent (Chapter 8). This result (Theorem 4), together with the comparison analysis done in Chapter 8, are very important as they prove the close relation between the two problems.

Distribution

The global AP (Problem 4) needs a centralised entity to compute for all targets which is the closer agent that will add the target to its local TP (see Figure 5.1).

This assignment problem relates to coverage problems (Bullo et al. [11], Du et al. [19],Lloyd [30]) which seek to split a given area according to a certain performance criterion. Using coverage algorithms, the space is partitioned in sectors, and each sector is assigned to one agent. Thus, the targets inside a given sector will appear in the respective agent's tracking problem.

Therefore, our idea is to distribute the assignment problem among the agents. Motivated by work developed in Bullo et al. [11], we implement distributed partition algorithms to solve our assignment problem based only on synchronous or asynchronous communication between neighbours.

Once the assignment problem is distributed (Figure 5.1), we meet our goals obtaining a convex tracking problem using a distributed optimisation scheme.

Two-layer approach

Splitting our naive centralised approach into a agent-target assignment (AP) and local tracking problems (TP) motivates our *two-layer approach*. A first layer solves the AP, i.e., it splits the set of targets among the agents. Then, n local TPs, one for each agent, are implemented in the second layer. This novel two-layer approach has a distributed scheme with local convex optimisation-based tracking problems.

1. *Global assignment problem* [Chapter 6]: the first layer represents the global problem of assigning the set of targets among the different agents. A dynamic graph with the targets and agents is defined in which, at each time-step, each target appears only on the tracking problem of one agent.

   The global assignment problem is defined in Problem 4 which we show to be equivalent to a formulation using Voronoi partitions (Problem 5). To solve this problem we first propose a centralised Lloyd algorithm, and then we introduce two distributed algorithms based on the work of Bullo

et al. [11]. A synchronous- and an asynchronous-distributed algorithm are implemented based on different communication protocols between neighbour agents only.

Global optimality is not guaranteed for any of the distributed algorithms. In fact, not even the centralised Lloyd algorithm ensures that. However, both algorithms ensure to converge (almost surely) to local optima. Comparing both algorithms, the synchronous-distributed algorithm converges faster but it needs every agent to communicate with all its neighbours at all time-steps in a synchronised fashion. In spite of converging slower, the asynchronous-distributed algorithm needs only a single pair of random neighbours to communicate at each time-step and it converges using $\sim 75\%$ less communications than the synchronous-based algorithm.

2. *Local tracking problem* [Chapter 7]: the second layer implements n local tracking problems. Each agent tracks the set of targets that are assigned to it. This layer is inherently divided into n independent problems.

   The local tracking problem is defined using the performance criterion (5.1) for quadrotors and fixed-wings agents. A full target coverage constraint is added using a worst-case scenario fashion, guaranteeing persistent feasibility of the local tracking problems as long as there is a feasible solution.

   The quadrotors' problem is a convex optimisation problem, formulated in a Semi-Definite Positive program (SDP) form. For the fixed-wings' case, a non-linear equality constraint is needed due to the non-holonomic dynamics. Thus, this constraint is linearised and a Sequential Convex Programming (SCP) algorithm is introduced to solve the linearised SDPs sequentially. Both problems have linear time complexity with the number of targets $\mathcal{O}(m)$. And, since this problem is defined locally for each agent, there is no time complexity dependence with the number of agents.

In Chapter 8, the complete two-layer algorithm is formulated. The pseudo-code to implement on each agent is derived for the synchronous- and asynchronous-distributed two-layer algorithms. The centralised and the novel two-layer approach solutions are analysed, and equivalence of the steady-state solutions of both approaches for static targets is proved (Theorem 4). Also, some considerations are presented regarding moving targets. Our algorithms are extended with heterogeneous teams adaptations, collision avoidance and refuelling manoeuvres.

Finally, in Chapter 9, the centralised, the synchronous-distributed and the asynchronous-distributed algorithms are simulated and compared for three scenarios: static targets, slow targets and fast targets. Considering $n = 10$ agents and $m = 100$ targets, assuming to have n machines, one for each of the n agents, and using SeDuMi 1.3 in `MATLAB R2012a`; each machine would take around 0.8-0.95 seconds to compute the path planning for the respective agent at each time step achieving real-time. Also, noise simulations are analysed and the refuelling manoeuvre performance presented.

Chapter 6

The global assignment problem

The global Assignment Problem (AP) assigns each target to one agent, such that each agent has a finite and defined number of target to track (TP). This problem is defined in a centralised way depending on the agents' and targets' positions (Problem 4). Once the problem only depends on the current positions, it is formulated in the same way for quadrotors and fixed-wings. Work on this target-agent assignment problem can be found under *coverage control* problems. Coverage problems seek to study the partitioning of tasks for groups of autonomous agents (Howard et al. [25] and Bullo et al. [11]).

In this chapter, first of all we will recall the Assignment Problem defined in Chapter 5. Then as a first step, we will transform it into a future position's Voronoi-partition problem which divides the ground-plane among the agents. One of the possible approximate solutions of this "dynamic" Voronoi-partition problem is the Lloyd algorithm, considering a set of fixed-points, which is however a centralised algorithm. Lloyd algorithm can be seen as a minimisation procedure of a certain *energy functional*. By the use of Bullo et al. [11] we will see how to devise distributed algorithms to minimise the same energy functional of the Lloyd algorithm and ultimately solve the given AP.

6.1 Assignment problem

The Assignment Problem (AP) is first defined in a centralised way in Problem 4 which computes for each target q the closer agent a^q, considering a set of future agents' positions $(\mathbf{x}^i_{\mathrm{F,pos}})_{i=1}^n$

$$a^q = \underset{i\in\{1,\dots,n\}}{\arg\min} \{||\mathbf{x}^i_{\mathrm{F,pos}} - \mathbf{t}^q_{\mathrm{pos}}||^2\}, \tag{6.1}$$

i.e., we seek to divide the space into n regions, one for each agent. Considering the ground-plane ($h = 0$) of our $3D$ convex space S, we define a $2D$ set of interest $Q := [E, N]^\top \subset \mathbb{R}^2$. Note that Q is a compact convex subset of $\mathbb{R}^2$ with non-empty interior. We define a n-partition of Q as follows.

Definition 10 (n-partition of Q)**.** *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 . $\forall_{i,j\in\{1,\dots,n\}, i\neq j}\ \mathrm{int}(v^i) \cap \mathrm{int}(v^j) = \emptyset$.

Also, $(v^i)_{i=1}^n \in \mathcal{V}_n$, where $\mathcal{V}_n$ denotes the set of n-partitions of Q.

Let $i = \{1, ..., n\}$ denote the unique number of the set of agents (Definition 4), 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 $v^i \subset Q$. We also define a set of points $(\mathbf{y}^i)_{i=1}^n$ in Q, one for each component of the partition $(v^i)_{i=1}^n$. We refer to $\mathbf{y}^i \in Q$ as the *centre-point* of the tracking region v^i of the agent i.

6.2 Step 1. The equivalent Voronoi partition problem

Among all the possible n-partitions $(v^i)_{i=1}^n \in \mathcal{V}_n$ of Q, the define the *Voronoi n-partition* of Q, generated by the set of points $(\mathbf{y}^i)_{i=1}^n$, as follows.

Definition 11 (Voronoi n-partition of Q by $\mathbf{y}$)**.** *The Voronoi n-partition $(v^i(\mathbf{y}))_{i=1}^n$ of Q by the set of points $(\mathbf{y}^i)_{i=1}^n$ is defined as*

$$v^i(\mathbf{y}) := \{\mathbf{g} \in Q \mid \forall_{j\in\{1,\dots,n\}, j\neq i} \ ||\mathbf{g} - \mathbf{y}^i|| \leq ||\mathbf{g} - \mathbf{y}^j||\}, \quad i = \{1, \dots, n\} \tag{6.2}$$

where $\mathbf{y}^i$ represents the centre-point of the tracking region v^i, and $\mathbf{g} = (E, N)$ the coordinate in the ground-plane Q.

This partition is particularly suitable for dividing the space evenly among the agents. Also, the Voronoi partition (Definition 11) allows us to redefine our Global Assignment Problem as follows.

Problem 5 (Equivalent global assignment problem)**.**

Given the set of n agents, the set of m targets, and the Voronoi n-partition using the position of the agents $(v^i(\mathbf{x}_{\mathrm{F,pos}}))_{i=1}^n$, we seek to assign each target to the closer agent, by assigning a tracking region v^i for each agent i. The tracking regions are defined as

$$v^i(\mathbf{x}_{\mathrm{F,pos}}) := \{\mathbf{g} \in Q \mid \forall_{j\in\{1,\dots,n\}, j\neq i} \ ||\mathbf{g} - \mathbf{x}^i_{\mathrm{F,pos}}|| \leq ||\mathbf{g} - \mathbf{x}^j_{\mathrm{F,pos}}||\}, \quad i = \{1, \dots, n\}. \tag{6.3}$$

In fact both AP formulations (Problem 4 and Problem 5), have exactly the same solution. However, the equivalent Voronoi partition Problem 5 makes easier the distribution of the assignment problem. In Figure 6.1, we can see the global assignment problem solved using the Voronoi partition.

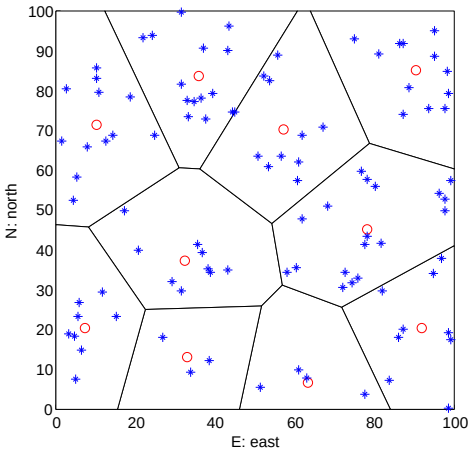

Figure 6.1: Partition example for solving agent-target assignment problem. The red circles represent the position of the agents and the blue stars the position of the targets.

Remark 3 (Heterogeneous teams)**.**

In the case where we have heterogeneous teams, i.e., agents with different sensing ranges or dynamics, we can introduce an extended version of the our Assignment Problem by introducing the power Voronoi partitions.

Definition 12 (Power Voronoi n-partition of Q by $\mathbf{y}$)**.** *The power Voronoi n-partition $(v^i(\mathbf{y}))_{i=1}^n$ of Q by the set of points $(\mathbf{y}^i)_{i=1}^n$ is defined as*

$$v^i(\mathbf{y}) := \{\mathbf{g} \in Q \mid \forall_{j\in\{1,...,n\},j\neq i} \ ||\mathbf{g}-\mathbf{y}^i|| - w^i \leq ||\mathbf{g}-\mathbf{y}^j|| - w^j\}, \ \ i = \{1,...,n\} \tag{6.4}$$

where $\mathbf{y}^i$ represents the centre-point of the tracking region v^i, w^i the power factor of the agent i, and $\mathbf{g} = (E, N)$ the coordinate in the ground-plane Q.

Note that the higher the power-factor w^i, the more space is assigned to the agent i. Therefore the weight can be seen as a measure of the sensing-capability of a certain agent comparing with the others.

6.3 Step 2. Lloyd algorithm

The global Assignment Problem 5 seeks to find the Voronoi partition $(v^i(\mathbf{x}_{\mathrm{F,pos}}))_{i=1}^n$ considering a future position for each agent $\mathbf{x}^i_{\mathrm{F,pos}}$. Knowing that the Tracking Problem seeks to steer each agent i towards the "centre" of the targets inside the tracking region v^i, motivates the use of this "centre" as the its future position $\mathbf{x}^i_{\mathrm{F,pos}}$. Also, this choice help us obtaining convergence guarantees in Chapter 8. Therefore, the proposed problem can be solved using the so-called "centring and partitioning" algorithm first introduced by Lloyd [30].

Defining the *generalised centroid* of each subset $(v^i)_{i=1}^n$ of Q as in Bullo et al. [11].

Definition 13 (Generalised centroid C of the subset $v^i \subset Q$)**.** *The generalised centroid of a compact set $v^i \subset Q$ is given by the unique minimum*

$$C(v^i) := \arg\min_{\mathbf{y}^i} \left\{ \int_{v^i} f(||\mathbf{y}^i - \mathbf{g}||)\phi(\mathbf{g})d\mathbf{g} \right\}, \tag{6.5}$$

where $\mathbf{y}^i$ is any point in v^i, v^i is a compact subset of Q, $\phi(\mathbf{g}) : Q \to \mathbb{R}^+$ a bounded integrable positive density function defined in v^i, $\mathbf{g}$ the coordinate in the subset $v^i \subset Q$, and f a performance Lipschitz function $f : \mathbb{R}^+ \to \mathbb{R}^+$.

And defining the performance function as the 2-norm Euclidean distance $f(\mathbf{g}) = ||\mathbf{g}||^2$, the global minimum of the *generalised centroid* of v^i is the *centre-of-mass*, defined as follows.

Definition 14 (Centre-of-mass C of the subset $v^i \subset Q$)**.** *The centre-of-mass of a compact set $v^i \subset Q$ is given by*

$$C(v^i) := \left(\int_{v^i} \phi(\mathbf{g})d\mathbf{g} \right)^{-1} \int_{v^i} \mathbf{g}\phi(\mathbf{g})d\mathbf{g}, \tag{6.6}$$

where v^i is a compact subset of Q, $\phi(\mathbf{g}) : Q \to \mathbb{R}^+$ a bounded integrable positive density function defined in v^i, and $\mathbf{g}$ the coordinate in the subset $v^i \subset Q$.

To study the global convergence of the Lloyd algorithm, we first define the *Lloyd map* (Du et al. [18]).

Definition 15 (Lloyd Map M)**.** *The Lloyd map is given by $M = C \circ V$, where C denotes the map that matches the partition $\{v^i(\mathbf{y})\}_{i=1}^n$ to the mass centroids $\{C(v^i)\}_{i=1}^n$, and V denotes the Voronoi partition $\{v^i(\mathbf{y})\}_{i=1}^n$ of the set of points $\{\mathbf{y}^i\}_{i=1}^n$.*

Then we recall some results from Du et al. [19].

Proposition 2 (Fixed point of the Lloyd Map)**.** *Any limit point $\mathbf{y} = (\mathbf{y}^i)_{i=1}^n$ of the Lloyd algorithm is a fixed point of the Lloyd map M (Definition 15) and a critical point of the energy functional $\mathcal{H}$.*

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

Theorem 2 (Finite set of fixed points)**.** *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 $\mathcal{H}$.*

Our Lloyd map has typically more than a single fixed point, as we will see in the simulation results. Therefore, global optimality of the Lloyd algorithm is not ensured. However, considering moving targets scenarios, converging to the global optimum is not our first priority.

6.4 Step 3. The energy functional

The Lloyd's "centring and partitioning" algorithm chooses the partition v and the centre-points $\mathbf{y}$ that minimise the following *energy functional*

$$\mathcal{H}(v, \mathbf{y}) := \sum_{i=1}^{n} \int_{v^i} f(||\mathbf{y}^i - \mathbf{g}||)\phi(\mathbf{g})d\mathbf{g}, \tag{6.7}$$

where $\mathbf{y}^i$ is any point in Q, $(v^i)_{i=1}^n$ is a n-partition of Q, $\phi(\mathbf{g}) : Q \to \mathbb{R}^+$ a bounded integrable positive density function defined in Q, $\mathbf{g}$ the coordinate in the ground-plane Q, and f a performance Lipschitz function $f : \mathbb{R}^+ \to \mathbb{R}^+$. We define our density function $\phi(\mathbf{g})$ as a *target density function*

$$\phi(\mathbf{g}) := \sum_{q=1}^{m} \delta(\mathbf{g} - \mathbf{t}_{\text{pos}}^q), \tag{6.8}$$

where $\mathbf{t}_{\text{pos}}^q$ represents the position of the target q, $\mathbf{g}$ the coordinate in the ground-plane Q, and δ the Dirac function. Note that this density function is bounded, integrable, non-negative and defined in along the ground-plane $\mathbf{g} \in Q$. Using the density function (6.8) and using the 2-norm performance function $f(\mathbf{g}) = ||\mathbf{g}||^2$, we rewrite the *energy functional* of the Assignment Problem as follows.

$$\mathcal{H}(v, \mathbf{y}) = \sum_{i=1}^{n} \int_{v^i} ||\mathbf{y}^i - \mathbf{g}||^2 \sum_{\mathbf{t}_{\text{pos}}^q \in v^i} \delta(\mathbf{g} - \mathbf{t}_{\text{pos}}^q)d\mathbf{g} = \sum_{i=1}^{n} \left(\sum_{\mathbf{t}_{\text{pos}}^q \in v^i} ||\mathbf{y}^i - \mathbf{t}_{\text{pos}}^q||^2 \right). \tag{6.9}$$

Regarding the energy functional metric (6.9) that the Lloyd algorithm seek to minimise, we recall the Proposition 2.4. from Bullo et al. [11] (Proposition 3).

Proposition 3 (Energy functional $\mathcal{H}(v, \mathbf{y})$)**.** *Considering any partition $(v^i)_{i=1}^n \in \mathcal{V}_n$, any set of points $(\mathbf{y})_{i=1}^n \in Q$, and the energy functional* (6.7)*, the following properties hold*

$$\begin{cases} \mathcal{H}(v_{\text{voronoi}}(\mathbf{y}), \mathbf{y}) \leq \mathcal{H}(v, \mathbf{y}), \\ \mathcal{H}(v, C(v)) \leq \mathcal{H}(v, \mathbf{y}), \end{cases} \tag{6.10}$$

where $v_{\text{voronoi}}(\mathbf{y})$ *represents the Voronoi partition of* Q *by* $\mathbf{y}$ *(Definition 11) and* $C(v)$ *corresponds to the centre-of-mass of the subsets* $(v^i)_{i=1}^n \in Q$ *(Definition 14).*

Therefore, 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 a monotonically non-increasing with the *energy functional* (6.9).

6.5 Step 4. Distributed algorithms

We distribute the proposed Lloyd algorithm to solve the global AP into distributed AP units adapting the work done of Bullo et al. [11]. The authors present two distributed extensions of the Lloyd "centring and partitioning" algorithm, relying on synchronous and asynchronous local communication respectively.

First we introduce some definitions needed to distribute our Assignment Problem.

Definition 16 (Neighbour agents)**.** *A pair of agents* (i, j) *with tracking regions* $v^i, v^j \in Q$ *are considered neighbours if and only if:*

$$\partial v^i \cap \partial v^j \neq \emptyset. \tag{6.11}$$

Definition 17 (Delaunay graph)**.** *The Delaunay graph (de Berg et al. [15]) is the undirected graph with the node set* $(\mathbf{y}^i)_{i=1}^n$*, and with edges in* $(\mathbf{y}^i, \mathbf{y}^j)$ *if and only if the agents* (i, j) *are neighbours.*

Definition 18 ($(\mathbf{y}^i, \mathbf{y}^j)$-bisector half-space)**.** *Considering the Voronoi partition (Definition 11) and two distinct points* $\mathbf{y}^i, \mathbf{y}^j \in Q$*, the* $(\mathbf{y}^i, \mathbf{y}^j)$*-bisector half-space is given by*

$$B(\mathbf{y}^i, \mathbf{y}^j) := \{\mathbf{g} \in Q \mid ||\mathbf{g} - \mathbf{y}^i|| \leq ||\mathbf{g} - \mathbf{y}^j||\}, \tag{6.12}$$

where $\mathbf{y}^i, \mathbf{y}^j$ *represents two points in* Q *and* $\mathbf{g}$ *the coordinate in the ground-plane* Q*.*

Then, using the density function (6.8) and the 2-norm performance function $f(\mathbf{g}) = ||\mathbf{g}||^2$, we compute the centre-of-mass (6.6) of $v^i \subset Q$ as follows.

$$C(v^i) = \left(\int_{v^i} \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \delta(\mathbf{g} - \mathbf{t}^q_{\text{pos}}) d\mathbf{g} \right)^{-1} \int_{v^i} \mathbf{g} \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \delta(\mathbf{g} - \mathbf{t}^q_{\text{pos}}) d\mathbf{g} = \frac{1}{m^i} \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \mathbf{t}^q_{\text{pos}}, \tag{6.13}$$

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

Remark 4 (Analytical solution for the centre-of-mass equation)**.**

Note that using the targets' density function (6.8) *and a 2-norm Euclidean distance performance function, the centre-of-mass Equation* (6.13) *avoids the numerical computation of integrations which speeds up the assignment algorithm.*

Finally, the two partition-based coverage algorithms are presented, based on the work of Bullo et al. [11]. These algorithms minimise the energy functional (6.9) with distributed communication schemes. The tracking regions update of the algorithms is defined as follows.

$$v^i_{k+1} = \left(v^i_k \cup v^j_k \right) \cap B(C(v^i_k), C(v^j_k)), \tag{6.14a}$$

$$v_{k+1}^j = \left(v_k^i \cup v_k^j \right) \cap B(C(v_k^j), C(v_k^i)), \tag{6.14b}$$

where the bisector half-space $B(.,.)$ is given by (6.12), and the centre-of-mass $C(v^i)$ by (6.13).

For both algorithms, a distributed scheme is implemented based on direct-neighbours communication. However, the communication protocol required varies between the two presented algorithms.

1. *synchronous-reliable-distributed communication:* needs a synchronous communication protocol between the agents without failures;

Algorithm 1 (Synchronous-distributed coverage)**.**

- for all $k \in \mathbb{N}$, each agent $i \in \{1, ..., n\}$ maintains in memory a tracking region $v_k^i \subset Q$

- given $(v_0^1, ..., v_0^n)$ an initial polygonal n-partition of Q

repeat

1. for every pair of communication neighbour agents (i, j) (6.11)

2. each agent computes its centre-of-mass $C(v_k^i), C(v_k^j)$ (6.13)

3. agent i communicates its centroid $C(v_k^i)$ to agent j and vice-versa

4. update tracking regions v_{k+1}^i (6.14a) *and v_{k+1}^j* (6.14b)

***until** the simulation is stopped.*

---

2. *asynchronous-unreliable-distributed communication:* the communication protocol can be asynchronous and failures may occur without compromising the performance.

Algorithm 2 (Asynchronous-distributed coverage (Gossip))**.**

- for all $k \in \mathbb{N}$, each agent $i \in \{1, ..., n\}$ maintains in memory a tracking region $v_k^i \subset Q$

- given $(v_0^1, ..., v_0^n)$ an initial polygonal n-partition of Q

repeat

1. select a random pair of communication neighbour agents (i, j) (6.11)

2. agents i, j compute its centre-of-mass $C(v_k^i), C(v_k^j)$ (6.13)

3. agent i communicates its centroid $C(v_k^i)$ to agent j and vice-versa

4. update tracking regions v_{k+1}^i (6.14a) *and v_{k+1}^j* (6.14b)

***until** the simulation is stopped.*

The amount of communication needed differs largely between the two algorithms. In the Algorithm 1 (Figure 6.2), at each time-step every agent communicates with its direct neighbours in the *Delaunay graph* (Definition 17). In the worst-case scenario, an agent might have $(n-1)$ neighbours to communicate with. Therefore, each agent must be able to communicate with all the other agents for this worst-case scenario. In the Algorithm 2, the communication needed is largely reduced: only a random pair of neighbour agents communicate at each time-step (Figure 6.3).

6.5.1 Distributed algorithms properties

Our synchronous-reliable distributed Algorithm 1, is in fact a adaptation of the classical Lloyd "centring and partition" algorithm (Du et al. [18]). The Lloyd algorithm motivated many research efforts in later

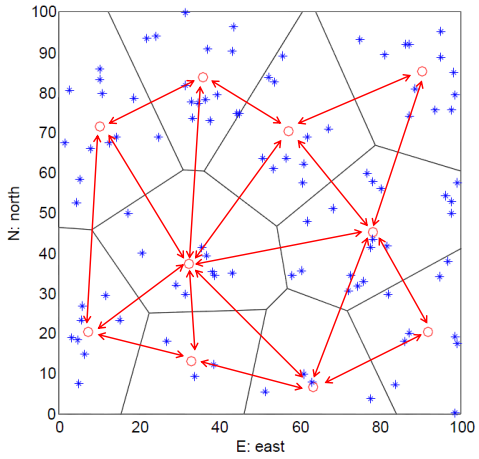

Figure 6.2: Synchronous-distributed Algorithm 1 where every agent communicates with its neighbours every time-step.

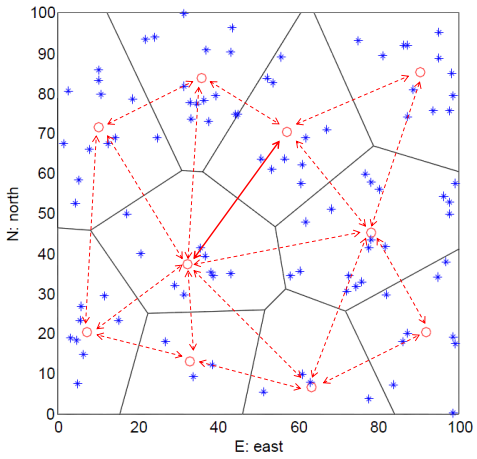

Figure 6.3: Asynchronous-distributed Algorithm 2 where only a random pair of neighbour agents communicate at each time-step.

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 (Du et al. [19]). The main converging result for the Lloyd algorithm is presented here in Proposition 3.

Generally, the global minimum of the *energy functional* (6.9) is not guaranteed with the proposed distributed coverage Algorithm 1. The "centring and partition" is an alternating variable algorithm, i.e., we do not minimise for the optimisation variables $(v, \mathbf{y})$ together, we first optimise for $\mathbf{y}$ and then for v.

Also, the synchronous-distributed coverage Algorithm 1 has some limitations, as it is applicable only in the presence of synchronized and reliable communication between all the agents. Also, in the worst-case scenario, an agent must be able to communicate with all the other agents.

Therefore, we formulate the asynchronous-distributed coverage Algorithm 2 which is a simple, robust and effective algorithm for noisy and uncertain communication protocols. This algorithm is implementable in an asynchronous, unreliable and delayed communication. However, due to having less communications per time-step, the asynchronous-distributed coverage algorithm converges slower than the distributed coverage one.

To study the convergence of the asynchronous-unreliable distributed Algorithm 2, with respect to the distributed solution of Algorithm 1, we recall a convergence result from Bullo et al. [11].

Theorem 3 (Convergence under persistent gossip communication)**.** *Considering the Lloyd map M (Definition 15) which is a stochastic process. If M is a randomly persistent stochastic process, then the gossip coverage Algorithm 2 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 coverage Algorithm 2 converges almost surely to a fixed point of the Lloyd map, i.e., it converges to a local minimum. However, the solution obtained with this algorithm can, in general, be different than the solution using the synchronous-distributed coverage Algorithm 1. Also, the global optimality of this asynchronous-distributed Algorithm 2 is not guaranteed, as in the synchronous-distributed Algorithm 1.

6.5.2 Distributed algorithms results

The convergence of the two proposed coverage algorithms is analysed with numerical results. We propose a synchronous-distributed coverage Algorithm 1 and an asynchronous-distributed coverage Algorithm 2, which seek to find the n-partition of the space Q that minimises the *energy functional* (6.9).

Note that, with moving targets scenarios, there is no stationary solution. Thus, the convergence of these algorithms is studied using static targets. The parameters used for the simulation are presented in Table 6.1.

Parameter	Value	Description
n, m	10, 100	number of agents, targets
k	60	number of time-steps of the simulation

Table 6.1: Assignment problem parameters using static targets.

Synchronous-distributed coverage algorithm global convergence

Regarding our synchronous-distributed coverage Algorithm 1 and using the same targets' positions, the energy-functions from ten simulations using different initial points are presented (Figure 6.4).

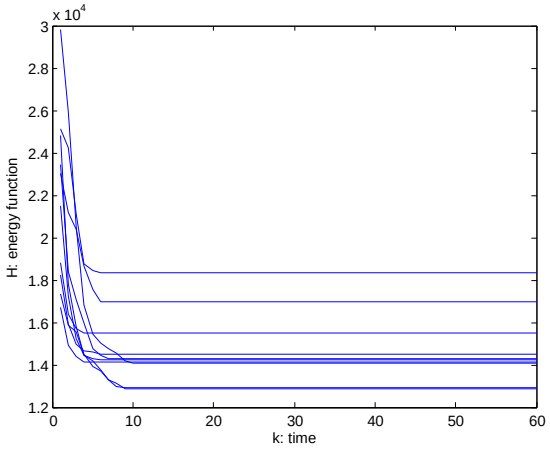

Figure 6.4: Synchronous-distributed coverage algorithm: many local minima.

As mentioned in the previous section, our synchronous-distributed algorithm, an adaptation of the classical Lloyd algorithm, does not guarantee to find the global optimum. We see clearly that, for the same problem (same targets' positions), we obtain different solutions depending on the initial condition (initial Voronoi partition). Therefore, we can conclude that, in general, our problem has many local minima. The presented algorithm does not avoid local optima which makes the steady-state solution very sensitive to the initial conditions.

Asynchronous-distributed coverage algorithm convergence

The asynchronous-distributed coverage Algorithm 2, requires less communication between the agents. However, its results are not deterministic, in the sense that, the algorithm does random movements in the choice of which agents should communicate at each time step. Using the same static scenario (targets' position) and starting with the same initial conditions (initial Voronoi partition), the results of the synchronous-distributed algorithm and of ten independent simulations using the asynchronous-distributed algorithm are shown (Figure 6.5).

The asynchronous-distributed coverage Algorithm 2 converges almost surely to a local minimum of the energy function (6.9) (Theorem 3). However, as realised in the simulations, the Algorithms 1 and 2 may not have the same partition solution. Therefore, depending on the random movement of the gossip coverage algorithm, it will end up in different local optima.

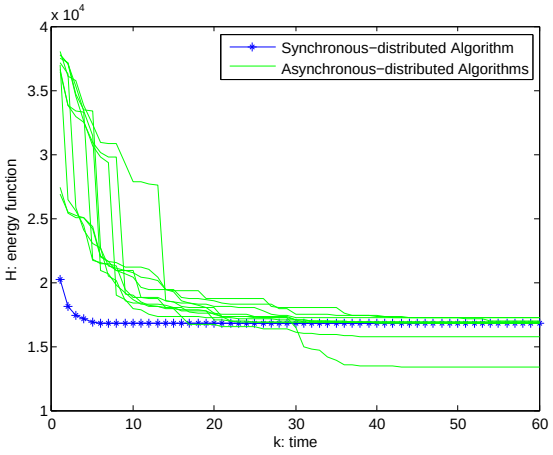

Figure 6.5: Asynchronous-distributed coverage algorithm: convergence to synchronous-distributed solution.

The obtained results also suggest that the slower convergence of the asynchronous-distributed algorithm might in fact help finding better partitions than the one obtained using the synchronous-distributed algorithm. As seen in Figure 6.5, the asynchronous-unreliable algorithm manage to escape the local optimal obtained with the synchronous-reliable algorithm and find a better partition in two out of ten simulations.

Synchronous- and asynchronous-distributed coverage algorithms comparison

The synchronous-distributed coverage Algorithm 1 takes an average of 6.6 time-steps to converge to the steady-state solution, while the asynchronous-distributed coverage Algorithm 2 needs 32.5 time-steps in average.

In terms of communication protocol, the asynchronous-distributed coverage does not need a synchronous nor reliable communication network. Regarding bandwidth, the synchronous-distributed coverage needs around 20 agent-to-agent communications per time-step in this scenario, while the asynchronous-distributed coverage algorithm uses a single agent-to-agent communication per time-step.

The asynchronous-distributed coverage needs significantly less communications to converge to the steady-state solution. The Algorithm 2 needs $32.5 \times 1 = 32.5$ communications to converge, while the Algorithm 1 needs roughly four times more communications $6.6 \times 20 = 132$.

To summarise, despite converging slower, the asynchronous-distributed coverage algorithm removes some limiting assumptions regarding the communication protocol made by the synchronous-distributed coverage algorithm. The asynchronous-distributed coverage converges using 75% less communications, without compromising the performance of the steady-state solution. It is able to optimise the Assignment Problem even with an asynchronous, unreliable, and delayed communication (Bullo et al. [11]). In fact, the simulation results suggest that the steady-state energy-function might even be lower.

Remark 5 (Vehicle dynamics)**.**

Note that these partition algorithms do not take into account the vehicle dynamics. Therefore the AP is defined in the same way for quadrotors and fixed-wings, regardless the local TPs.

Chapter 7

The local tracking problem

In this chapter we define the local tracking problems for both UAV models: quadrotors and fixed-wings. After assigning the set of targets to be tracked by each agent, we implement the corresponding tracking algorithm. These local problems are inherently distributed, i.e., we have one problem to be solved independently by each agent, corresponding to the second layer of our novel two-layer approach.

Starting with the quadrotor case, we define the local multi-target tracking problem for one agent. The problem is introduced as in the naive approach, then it is rearranged to a Semi-Definite Positive program (SDP) to steer the UAV. The SDP is a common sub-field of convex optimisation which deals with the optimisation of a linear cost function subjected to linear equality constraints and to Linear-Matrix Inequalities (LMIs). The full target coverage constraints are also introduced as LMIs. This type of problem can be solved efficiently by different solvers which typically implement interior point methods.

For the fixed-wing case, the problem is non-convex due to their non-holonomic dynamics. Therefore, we linearise the problem to obtain a convex problem. A Sequential Convex Programming (SCP) is proposed to solve SDPs iteratively and steer the fixed-wings.

7.1 Local tracking problem for a quadrotor

Using a single quadrotor (Figure 2.2), we first recall the discrete-time double integrator unit point-mass dynamics

$$\mathbf{x}_{k+1} \equiv \begin{bmatrix} \mathbf{x}_{k+1,\text{pos}} \\ \mathbf{x}_{k+1,\text{vel}} \end{bmatrix} = \mathbf{f}_{\text{quad}}(\mathbf{x}_k, \mathbf{u}_k) \equiv \mathbf{A} \begin{bmatrix} \mathbf{x}_{k,\text{pos}} \\ \mathbf{x}_{k,\text{vel}} \end{bmatrix} + \mathbf{B}\mathbf{u}_k, \quad \mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3\Delta t \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times 3} \\ \mathbf{I}_3\Delta t \end{bmatrix},$$
$$\text{s.t.} \begin{cases} h \geq h_{\min} \\ -v_{\max} \leq v_E, v_N, v_h \leq v_{\max} \\ -f_{\max} \leq f_E, f_N, f_h \leq f_{\max} \end{cases}, \tag{7.1}$$

where $\mathbf{x}$ is the state of the quadrotor given by its position $\mathbf{x}_{\text{pos}}$ and its velocity $\mathbf{x}_{\text{vel}} = [v_E, v_N, v_h]^\top$, the input $\mathbf{u} = [f_E, f_N, f_h]^\top$ corresponds to the force applied along the three axis, and Δt is the sampling period. We consider limitations on the quadrotors minimum flying altitude, bounded velocity and force, given by the parameters $h_{\min}, v_{\max}, f_{\max}$.

7.1.1 Performance criterion

Using the our performance criterion (4.2), used in centralised approach (Problem 2) for the single quadrotor case

$$\bar{J} = \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left(||\mathbf{x}_{t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \right) + \rho ||\mathbf{u}_t||^2 \right] + \varphi \sum_{q=1}^{m} \left(||\mathbf{x}_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \right), \tag{7.2}$$

which seeks to minimise the distance over time towards the current targets' position. We rewrite the cost function removing the constant terms (depending only on $\mathbf{t}$) as follows.

$$\begin{aligned} \bar{J} = & \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left((\mathbf{x}_{t+1,\text{pos}})^\top \mathbf{x}_{t+1,\text{pos}} - 2\mathbf{x}_{t+1,\text{pos}} \mathbf{t}^q_{k,\text{pos}} + \rho \mathbf{u}_t^\top \mathbf{u}_t \right] \right. \\ & + \varphi \sum_{q=1}^{m} \left((\mathbf{x}_{k+N,\text{pos}})^\top \mathbf{x}_{k+N,\text{pos}} - 2\mathbf{x}_{k+N,\text{pos}} \mathbf{t}^q_{k,\text{pos}} \right). \end{aligned}$$

Using the optimisation variables $\mathbf{U} = \mathbf{u}_t, \mathbf{X}_{\text{pos}} = \mathbf{x}_{t+1,\text{pos}}, \mathbf{X}_{\text{vel}} = \mathbf{x}_{t+1,\text{vel}}, t = \{k, ..., k+N-1\}$, stacked as

$$\mathbf{o} = \begin{bmatrix} \mathbf{u}_t \\ \mathbf{x}_{t+1,\text{pos}} \\ \mathbf{x}_{t+1,\text{vel}} \end{bmatrix}, t = \{k, ..., k+N-1\},$$

and defining, using the *Kronecker* product $\otimes$,

$$\mathbf{Q_1} = \mathbf{I}_{3N},\ \mathbf{Q_2} = \begin{bmatrix} \mathbf{0}_{3(N-1)\times 3(N-1)} & \\ & \mathbf{I}_3 \end{bmatrix},\ \mathbf{d} = \begin{bmatrix} -2 \sum_{q=1}^{m} \mathbf{t}^q_{k,\text{pos}} \end{bmatrix}_{3\times 1},\ \mathbf{c_1} = [\mathbf{1}_{N\times 1} \otimes \mathbf{d}],\ \mathbf{c_2} = \begin{bmatrix} \mathbf{0}_{3(N-1)\times 1} \\ \mathbf{d} \end{bmatrix}. \tag{7.3}$$

We can write the performance criterion in a quadratic form

$$\bar{J} = \mathbf{X}^\top_{\text{pos}} [m\mathbf{Q_1} + m\varphi\mathbf{Q_2}] \mathbf{X}_{\text{pos}} + \mathbf{U}^\top [\rho \mathbf{Q_1}] \mathbf{U} + [\mathbf{c_1} + \varphi \mathbf{c_2}]^\top \mathbf{X}_{\text{pos}}.$$

7.1.2 Equality constraints

Considering the equality constraints in (4.3) of the Problem 2 for a single quadrotor, we can formalise them in the form

$$\mathbf{A_{eq}} \mathbf{o} = \mathbf{b_{eq}},$$

with

$$\mathbf{A_{eq}} = \begin{bmatrix} \begin{bmatrix} -\mathbf{B} & \mathbf{I}_6 \end{bmatrix} & & & & & \\ \begin{bmatrix} \mathbf{0}_{6\times 3} & -\mathbf{A} \end{bmatrix} & \begin{bmatrix} -\mathbf{B} & \mathbf{I}_6 \end{bmatrix} & & & & \\ & \begin{bmatrix} \mathbf{0}_{6\times 3} & -\mathbf{A} \end{bmatrix} & \begin{bmatrix} -\mathbf{B} & \mathbf{I}_6 \end{bmatrix} & & & \\ & & \ddots & & \ddots & \\ & & & \begin{bmatrix} \mathbf{0}_{6\times 3} & -\mathbf{A} \end{bmatrix} & \begin{bmatrix} -\mathbf{B} & \mathbf{I}_6 \end{bmatrix} \end{bmatrix}, \tag{7.4}$$

and

$$\mathbf{b_{eq}} = \begin{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{x}_{k,\text{pos}} \\ \mathbf{x}_{k,\text{vel}} \end{bmatrix} \\ \mathbf{0}_{6(N-1)\times 1} \end{bmatrix}, \tag{7.5}$$

where, the matrices $\mathbf{A}, \mathbf{B}$ are defined in (7.1).

7.1.3 Inequality constraints

Considering the inequality constraints in (4.3) of the Problem 2 for a single-quadrotor, we can formalise them in the form

$$\mathbf{A_{ineq}o} \leq \mathbf{b_{ineq}},$$

with

$$\mathbf{A_{ineq}} = \mathbf{I}_N \otimes \begin{bmatrix} \mathbf{I}_3 & & & \\ -\mathbf{I}_3 & & & \\ & \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} & & \\ & & & \mathbf{I}_3 \\ & & & -\mathbf{I}_3 \end{bmatrix} \tag{7.6}$$

and

$$\mathbf{b_{ineq}} = \mathbf{1}_{N\times 1} \otimes \begin{bmatrix} \mathbf{1}_{3\times 1} \otimes f_{\max} \\ \mathbf{1}_{3\times 1} \otimes f_{\max} \\ -h_{\min} \\ \mathbf{1}_{3\times 1} \otimes v_{\max} \\ \mathbf{1}_{3\times 1} \otimes v_{\max} \end{bmatrix}. \tag{7.7}$$

7.1.4 Full target coverage constraint

At all times, we constrain the agent's position to be able to sense all the targets that are assigned to it. To achieve that, our full target coverage constraint considers all the possible targets' positions for the next time steps. However, constraining the problem considering all the possible targets' positions for the entire predictive horizon would be too restrictive due to two main reasons:

- the targets' movement is considered independent from the agents' positions, thus it is unlikely that the targets will travel often at full speed away from the agents;
- the targets that move away from the line-of-sight of one agent are likely to be assigned to another closer agent later on.

Therefore, we will constraint the problem considering all the possible targets' positions only for the next time step $\mathbf{t}_{k+1,\text{pos}}$.

Considering our UAV dynamics, both for quadrotors and fixed-wings, the relationship input-output is delayed by one time-step. Any change in the input takes one extra time-step to be noticed in the agent's position, i.e., the input $\mathbf{u}_k$ influences the state at $k+1$ but the position only at $\mathbf{x}_{k+2,\text{pos}}$, instead of $\mathbf{x}_{k+1,\text{pos}}$. Thus, the full target constraint will be applied considering the positions two time-steps ahead, bounding the distances $||\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^q_{k+1,\text{pos}}||, q = \{1, ..., m\}$.

The next targets' position is given by the random walk model (2.14):

$$\mathbf{t}^q_{k+1,\text{pos}} = \mathbf{t}^q_{k,\text{pos}} + \mathbf{t}^q_{k,\text{vel}}\Delta t, \quad q \in \{1, ..., m\},$$

where $\mathbf{t}^q_{\text{pos}}$ is the position of the target q with no altitude $h^q = 0$, and the velocity is given by $\mathbf{t}^q_{\text{vel}} = [w^q_E, w^q_N, 0]^\top$, where w^q_E, w^q_N are zero mean bounded Gaussian noises, bounded by $||\mathbf{t}_{\text{vel}}|| \leq W_{\max}$. Note that in general we can have targets with different maximum velocities in the same scenario. The agent does not know the future targets' position, only their maximum velocity. Thus, the coverage has to be guaranteed for all possible next cases.

To guarantee full target coverage, we define the worst case scenario such that if we guarantee to fulfil the constraints for these points, we are feasible for any future targets' positions. This worst case scenario

is illustrated in the Figure 7.1, and it happens exactly when all the targets move away from the agent at full speed

$$\mathbf{t}^{q,*}_{k+1,\text{pos}} = \mathbf{t}^{q}_{k,\text{pos}} + \frac{W_{\max}}{||\mathbf{t}^{q}_{k,\text{pos}} - \mathbf{x}_{k,\text{pos}}||}\left(\mathbf{t}^{q}_{k,\text{pos}} - \mathbf{x}_{k,\text{pos}}\right)\Delta t, \quad q = \{1, ..., m\}. \tag{7.8}$$

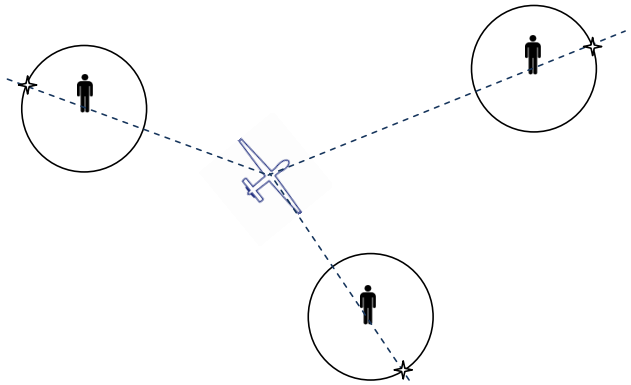

Figure 7.1: Full target coverage: the possible next targets' positions (bounded by the circumferences) and their worst-case scenario (marked with stars).

With this easy-to-implement formulation, we guarantee that the agent will be always capable to sense every target for the next time step regardless their movement. To motivate the need of this implementation, in Figure 7.2, one can clearly see that the cost function will be reduced if the agent moves closer to the group of targets placed together. However this constraint guarantees that the worst case scenario of the far away target is still within the agents' tracking range.

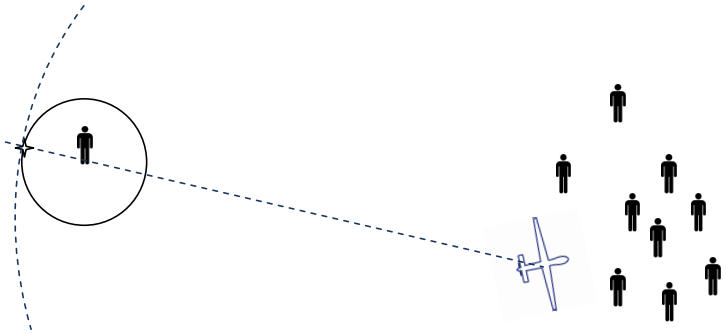

Figure 7.2: Full target coverage: worst case scenario motivation.

Once defining the worst case scenario (Equation (7.8)), we can introduce the remaining full target coverage constraints:

$$\mathbf{M}^q = \begin{bmatrix} \mathbf{I}_3 & (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^{q,*}_{k+1,\text{pos}}) \\ (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^{q,*}_{k+1,\text{pos}})^\top & R^2 \end{bmatrix} \succeq 0, \quad q = \{1, ..., m\}.$$

7.1.5 Semi-Definite Positive Program

Once we have only linear constraints: linear equalities, linear inequalities and linear matrix inequalities (full target coverage constraint); we seek to obtain a linear performance criterion to have a so-called Semi-Definite Positive program (SDP). Thus, the quadratic terms of the cost-function are transformed

into LMIs using the Schur Complement (Definition 6), obtaining a so-called *epigraph form* problem.

$$\min_{\mathbf{o}} \bar{J}_1(\mathbf{o}), \;\; \bar{J}_1(\mathbf{o}) := \mathbf{X}_{\text{pos}}^{\top}[m\mathbf{Q_1} + m\varphi\mathbf{Q_2}]\mathbf{X}_{\text{pos}} + \mathbf{U}^{\top}[\rho\mathbf{Q_1}]\mathbf{U} + [\mathbf{c_1} + \varphi\mathbf{c_2}]^{\top}\mathbf{X}_{\text{pos}} \Leftrightarrow$$

$$\Leftrightarrow \min_{\mathbf{o}} \bar{J}_2(\mathbf{o}) \;\text{ s.t. }\; \begin{bmatrix} [m\mathbf{Q_1} + m\varphi\mathbf{Q_2}]^{-1} & \mathbf{X}_{\text{pos}} \\ \mathbf{X}_{\text{pos}}^{\top} & \beta_{\mathbf{X}} \end{bmatrix} \succeq 0 \;\wedge\; \beta_{\mathbf{X}} \geq 0 \;\wedge\; \begin{bmatrix} [\rho\mathbf{Q_1}]^{-1} & \mathbf{U} \\ \mathbf{U}^{\top} & \beta_{\mathbf{U}} \end{bmatrix} \succeq 0 \;\wedge\; \beta_{\mathbf{U}} \geq 0,$$

$$\bar{J}_2(\mathbf{o}) := [\mathbf{c_1} + \varphi\mathbf{c_2}]^{\top}\mathbf{X}_{\text{pos}} + \beta_{\mathbf{X}} + \beta_{\mathbf{U}}.$$

Now, we can define the local TP for the quadrotor case as a SDP.

Problem 6 (Quadrotor local Tracking Problem in a SDP form)**.**

Given the optimisation variables $\mathbf{U} = \mathbf{u}_t, \mathbf{X}_{\text{pos}} = \mathbf{x}_{t+1,\text{pos}}, \mathbf{X}_{\text{vel}} = \mathbf{x}_{t+1,\text{vel}}, t = \{k, ..., k+N-1\}$, *stacked as*

$$\mathbf{o} = \begin{bmatrix} \mathbf{u}_t \\ \mathbf{x}_{t+1,\text{pos}} \\ \mathbf{x}_{t+1,\text{vel}} \end{bmatrix}, \;\; t = \{k, ..., k+N-1\},$$

and the cost function

$$\bar{J}(\mathbf{o}) := \mathbf{c}^{\top}\mathbf{X}_{\text{pos}} + \beta, \tag{7.9}$$

we define the problem

$$\min_{\mathbf{o}} \bar{J}(\mathbf{o}) \;\text{ s.t.}$$
$$\begin{cases} \mathbf{A_{eq}}\mathbf{o} = \mathbf{b_{eq}}, \\ \mathbf{A_{ineq}}\mathbf{o} \leq \mathbf{b_{ineq}}, \\ \mathbf{M}^q = \begin{bmatrix} \mathbf{I}_3 & (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}_{k+1,\text{pos}}^{q,*}) \\ (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}_{k+1,\text{pos}}^{q,*})^{\top} & R^2 \end{bmatrix} \succeq 0, \;\; q = \{1, ..., m\}, \\ \begin{bmatrix} \mathbf{Q}_{\mathbf{X}}^{-1} & \mathbf{X}_{\text{pos}} \\ \mathbf{X}_{\text{pos}}^{\top} & \beta_{\mathbf{X}} \end{bmatrix} \succeq 0, \;\; \beta_{\mathbf{X}} \geq 0, \;\; \begin{bmatrix} \mathbf{Q}_{\mathbf{U}}^{-1} & \mathbf{U} \\ \mathbf{U}^{\top} & \beta_{\mathbf{U}} \end{bmatrix} \succeq 0, \;\; \beta_{\mathbf{U}} \geq 0, \end{cases} \tag{7.10}$$

where $\mathbf{c} = [\mathbf{c_1} + \varphi\mathbf{c_2}]$, $\beta = \beta_{\mathbf{X}} + \beta_{\mathbf{U}}$, $\mathbf{Q_X} = [m\mathbf{Q_1} + m\varphi\mathbf{Q_2}]$ *and* $\mathbf{Q_U} = [\rho\mathbf{Q_1}]$. *The matrices* $\mathbf{Q_1}, \mathbf{Q_2}, \mathbf{c_1}, \mathbf{c_2}, \mathbf{A_{eq}}, \mathbf{b_{eq}}, \mathbf{A_{ineq}}, \mathbf{b_{ineq}}$ *are defined in* (7.3), (7.4), (7.5), (7.6), (7.7). *The worst case scenario* $\mathbf{t}_{k+1,\text{pos}}^{q,*}, q = \{1, ..., m\}$ *defined in* (7.8).

The parameters of the problem are

N	*predictive horizon*
m	*number of targets*
ρ, φ	*performance weighting factors*
$h_{\min}, v_{\max}, f_{\max}$	*agents' specifications*
$W_{\max}$	*targets' maximum velocity*
Δt	*sampling period*
R	*agents' maximum sensing range.*

Note that this problem is a convex problem (defined with a convex cost function subjected to convex inequality constraints and affine equality constraints). It was simulated using the solver SeDuMi from the MATLAB toolbox YALMIP (Lofberg [31]), commonly used to solve Semi-Definite Positive programs.

A simulation with multiple moving targets is presented to show the local tracking algorithm for quadrotors. The simulation parameters are presented in Table 7.1 and in the Figures 7.3,7.4 the quadrotor's

path and the cost-function during the simulation. As expected, the agent moves closer to the set of ten moving targets decreasing the cost function. Then, the performance stabilizes, oscillating due to the movement of the targets.

Parameter	Value	Description
N	3	finite horizon
m	10	number of targets
ρ, φ	0.01, 0.01	performance weighting factors
$h_{\min}, v_{\max}, f_{\max}$	10, 2, 2	quadrotors' specifications
$W_{\max}$	3	targets' maximum velocity
Δt	1	time-step (in seconds)
R	50	agents' maximum sensing range
k	80	number of time-steps of the simulation

Table 7.1: Quadrotor local tracking SDP parameters.

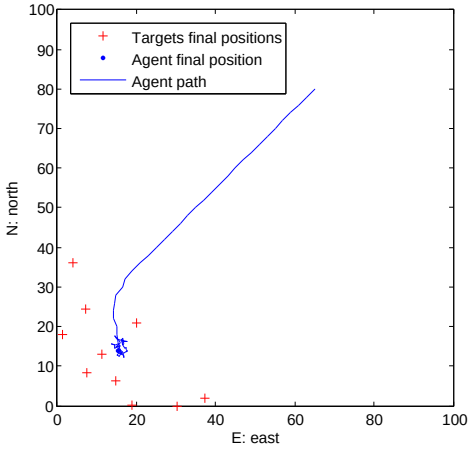

Figure 7.3: Quadrotor local SDP path.

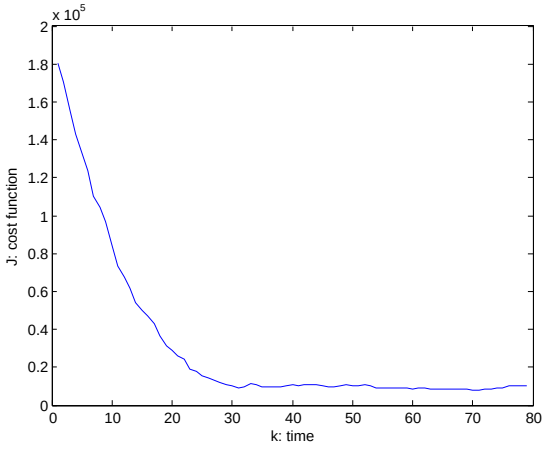

Figure 7.4: Quadrotor local SDP performance.

This simulation took 0.6 seconds per time-step to run using SeDuMi 1.3 in `MATLAB R2012a`, which represents a great improvement and matches our objectives. The time complexity of this problem can be derived analytically, it is a SDP which uses interior-point methods. The time complexity is given by $\mathcal{O}(\max\{o^3, o^2c\})$ (see Nemirovski [39]), where o represents the number of optimisation variables and c the number of constraints.

In this case, the number of optimisation variables is $o = 9N$ and the number of constraints $c = 24N + 8m + 2$, where N is the predictive horizon and m the number of targets. Therefore the time complexity of this problem depend mainly on the predictive horizon N but it also increases with the number of targets m. The dependence with the number of targets is linear, i.e., our algorithm is $\mathcal{O}(m)$. Since this problem is defined locally for each agent, there is no dependence with the number of agents: each agent solves its own tracking problem in a parallel fashion.

7.2 Local tracking problem for a fixed-wing

Using a single fixed-wing (Figure 2.3), we recall the discrete-time fixed-altitude nonholonomic *Dubins* car model

$$\begin{aligned} \mathbf{x}_{k+1} \equiv \begin{bmatrix} E_{k+1} \\ N_{k+1} \\ \theta_{k+1} \end{bmatrix} = \mathbf{f}_{\text{wing}}(\mathbf{x}_k, u_k) \equiv \begin{bmatrix} E_k \\ N_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} V_M \cos\theta_k \\ V_M \sin\theta_k \\ u_k \end{bmatrix} \Delta t, \\ \text{s.t.} \quad -u_{\max} \leq u \leq u_{\max}, \end{aligned} \tag{7.11}$$

where $\mathbf{x}$ is the state of the fixed-wing given by its 2D position $[E, N]^\top$ and orientation θ^i, the scalar input u corresponds to the change of heading command, and Δt is the sampling period. These agents fly with constant velocity V_M and with a fixed-altitude $\mathbf{x}_{\text{pos}} = [E, N, \bar{h}]^\top$. We also consider that the fixed-wings have a maximum turning rate.

7.2.1 Performance criterion

Using the our performance criterion (4.5), used in centralised approach (Problem 3) for the single fixed-wing case, rewritten in the same manner as the quadrotors' case, using the optimisation variables $\mathbf{U} = u_t, \mathbf{X} = \mathbf{x}_{t+1}, \boldsymbol{\theta} = \theta_{t+1}, t = \{k, ..., k+N-1\}$, stacked as

$$\mathbf{o} = \begin{bmatrix} u_t \\ \mathbf{x}_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} u_t \\ E_{t+1} \\ N_{t+1} \\ \theta_{t+1} \end{bmatrix}, t = \{k, ..., k+N-1\},$$

and defining, using the *Kronecker* product $\otimes$,

$$\mathbf{Q_1} = \mathbf{I}_{2N},\ \mathbf{Q_2} = \begin{bmatrix} \mathbf{0}_{2(N-1)\times 2(N-1)} & \\ & \mathbf{I}_2 \end{bmatrix},\ \mathbf{d} = \begin{bmatrix} -2 \sum_{q=1}^{m} \mathbf{t}^q_{k,\text{pos}} \end{bmatrix}_{2\times 1},\ \mathbf{c_1} = [\mathbf{1}_{N\times 1} \otimes \mathbf{d}],\ \mathbf{c_2} = \begin{bmatrix} \mathbf{0}_{2(N-1)\times 1} \\ \mathbf{d} \end{bmatrix}, \tag{7.12}$$

we can also write the performance criterion in a quadratic form for the fixed-wing case

$$\bar{J} = \mathbf{X}^\top [m\mathbf{Q_1} + m\varphi\mathbf{Q_2}]\mathbf{X} + \mathbf{U}^\top [\rho \mathbf{I}_N]\mathbf{U} + [\mathbf{c_1} + \varphi \mathbf{c_2}]^\top \mathbf{X}.$$

7.2.2 Linear equality constraints

Considering the linear equality constraints in (4.6) of the Problem 3 for a single fixed-wing, we can formalise them in the form

$$\mathbf{A_{eq}}\mathbf{o} = \mathbf{b_{eq}},$$

with

$$\mathbf{A_{eq}} = \begin{bmatrix} \begin{bmatrix} -\Delta t & 0 & 0 & 1 \end{bmatrix} & & & \\ \begin{bmatrix} 0 & 0 & 0 & -1 \end{bmatrix} & \begin{bmatrix} -\Delta t & 0 & 0 & 1 \end{bmatrix} & & \\ & \ddots & \ddots & \\ & & \begin{bmatrix} 0 & 0 & 0 & -1 \end{bmatrix} & \begin{bmatrix} -\Delta t & 0 & 0 & 1 \end{bmatrix} \end{bmatrix}, \tag{7.13}$$

and

$$\mathbf{b_{eq}} = \begin{bmatrix} \theta_k \\ \mathbf{0}_{(N-1)\times 1} \end{bmatrix}. \tag{7.14}$$

7.2.3 Non-linear equality constraints

Considering the non-linear equality constraints in (4.6) of the Problem 3 for a single fixed-wing, we can formalize it in the form

$$\mathbf{A_{nonlin}o} + \mathbf{d_{nonlin}(o)} = \mathbf{b_{nonlin}},$$

with

$$\mathbf{A_{nonlin}} = \begin{bmatrix} \begin{bmatrix} \mathbf{0}_{2\times1} & \mathbf{I}_2 & \mathbf{0}_{2\times1} \end{bmatrix} & & & \\ \begin{bmatrix} \mathbf{0}_{2\times1} & -\mathbf{I}_2 & \mathbf{0}_{2\times1} \end{bmatrix} & \begin{bmatrix} \mathbf{0}_{2\times1} & \mathbf{I}_2 & \mathbf{0}_{2\times1} \end{bmatrix} & & \\ & \ddots & \ddots & \\ & & \begin{bmatrix} \mathbf{0}_{2\times1} & -\mathbf{I}_2 & \mathbf{0}_{2\times1} \end{bmatrix} & \begin{bmatrix} \mathbf{0}_{2\times1} & \mathbf{I}_2 & \mathbf{0}_{2\times1} \end{bmatrix} \end{bmatrix}, \tag{7.15}$$

$$\mathbf{d_{nonlin}} = -V_M \Delta t \begin{bmatrix} \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \cos\theta_{k+N-1} \\ \sin\theta_{k+N-1} \end{bmatrix} \end{bmatrix}, \tag{7.16}$$

and

$$\mathbf{b_{nonlin}} = \begin{bmatrix} \begin{bmatrix} E_k \\ N_k \end{bmatrix} \\ \mathbf{0}_{2(N-1)\times1} \end{bmatrix}. \tag{7.17}$$

7.2.4 Inequality constraints

Considering the inequality constraints in (4.6) of the Problem 3 for a single fixed-wing, we can formalise them in the form

$$\mathbf{A_{ineq}o} \leq \mathbf{b_{ineq}},$$

with

$$\mathbf{A_{ineq}} = \mathbf{I}_N \otimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}, \tag{7.18}$$

and

$$\mathbf{b_{ineq}} = \mathbf{1}_{N\times1} \otimes \begin{bmatrix} u_{\max} \\ u_{\max} \end{bmatrix}. \tag{7.19}$$

7.2.5 Full target coverage constraint

Using the same worst case scenario, modelled by

$$\mathbf{t}^{q,*}_{k+1,\text{pos}} = \mathbf{t}^{q}_{k,\text{pos}} + \frac{W_{\max}}{||\mathbf{t}^{q}_{k,\text{pos}} - \mathbf{x}_{k,\text{pos}}||}\left(\mathbf{t}^{q}_{k,\text{pos}} - \mathbf{x}_{k,\text{pos}}\right)\Delta t, \quad q = \{1, ..., m\}. \tag{7.20}$$

we can reintroduce the full target constraint as in (7.10):

$$\mathbf{M}^q = \begin{bmatrix} \mathbf{I}_3 & (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^{q,*}_{k+1,\text{pos}}) \\ (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^{q,*}_{k+1,\text{pos}})^\top & R^2 \end{bmatrix} \succeq 0, \quad q = \{1, ..., m\}.$$

Note that again here a change in the input u_k has effects only in the position two time-steps ahead $\mathbf{x}_{k+2,\text{pos}}$. Therefore, we will implement the full target constraint again with a two steps horizon.

7.2.6 Non-convex Problem

Having the constraints defined, and simplifying again the performance criterion in the *epigraph form*

$$\min_{\mathbf{o}} \bar{J}(\mathbf{o}) \text{ s.t. } \begin{bmatrix} [m\mathbf{Q_1} + m\varphi\mathbf{Q_2}]^{-1} & \mathbf{X} \\ \mathbf{X}^\top & \beta_{\mathbf{X}} \end{bmatrix} \succeq 0 \ \wedge \ \beta_{\mathbf{X}} \geq 0 \ \wedge \ \begin{bmatrix} [\rho\mathbf{I}_N]^{-1} & \mathbf{U} \\ \mathbf{U}^\top & \beta_{\mathbf{U}} \end{bmatrix} \succeq 0 \ \wedge \ \beta_{\mathbf{U}} \geq 0,$$

$$\bar{J}(\mathbf{o}) := [\mathbf{c_1} + \varphi\mathbf{c_2}]^\top \mathbf{X} + \beta_{\mathbf{X}} + \beta_{\mathbf{U}},$$

we can formulate our local tracking problem for the fixed-wing case. However, we end up with a non-convex problem due to the presence of non-linear equality constraints $\mathbf{A_{nonlin}o} + \mathbf{d_{nonlin}}(\mathbf{o}) = \mathbf{b_{nonlin}}$.

7.2.7 Linearisation of the non-linear dynamics

In order to convexify our problem, there are different ways of linearising non-linear constraints

1. *Feedback dynamic inversion*: this technique is widely used in the literature (see Keviczky et al. [27]) as it obtains a reasonable good performance. However it is difficult to apply for this case, we want to control at least $2N$ outputs (positions $(E_{k+t}, N_{k+t})_{t=1}^N$) using only N control variables (changes of heading $(u_{k+t-1})_{t=1}^N$);

2. *Non-holonomic transformation*: our vehicle dynamics, the *Dubins* car, is a nonholonomic dynamics. In Fazenda and Lima [21], the Authors cover the case when the dynamics can be made holonomic by using a suitable transformation. They define a mapping from the initial variables to the holonomic dynamics variables. However in our case, we still have constraints in the actual position of the agents. Thus, even if we find a transformation which makes the dynamics holonomic, we will need to have the inverse of the transformation for the positions' constraints which will lead to non-linear constraints once more;

3. *Sequential Convex Programming*: perhaps the most suitable for this rather "simple" discrete-time case. Using the first Taylor expansion of nonlinear functions, the Sequential Convex Programming (Definition 19) linearises the equality constraints and the problem becomes a Semi-Definite Positive program. We compute a SDP optimisation around the linearised vector $\boldsymbol{\theta_0}$, then we update the initial vector $\boldsymbol{\theta_0}$ and we run the optimisation again until the stopping criterion is met.

7.2.8 Sequential Convex Programming

To solve our fixed-wing local tracking, we propose to use a Sequential Convex Programming.

Definition 19 (Sequential Convex Programming (SCP))**.** *A Sequential Convex Programming (SCP) is a local optimisation method for non-convex problems, where convex portions of the problem are solved efficiently (in our case using a Semi-Definite Positive program). SCP starts with an initial point (initial-headings $\boldsymbol{\theta_0}$) and it updates this point by*

$$\boldsymbol{\theta_{0,i+1}} = (1-\alpha)\boldsymbol{\theta_{0,i}} + \alpha\boldsymbol{\theta_i^*}(\boldsymbol{\theta_{0,i}}), \tag{7.21}$$

where α is the step-size, $\boldsymbol{\theta_{0,i}}$ the linearised point and $\boldsymbol{\theta_i^*}(\boldsymbol{\theta_{0,i}}) = [\theta_{k+1}, ..., \theta_{k+N-1}]^\top$ the solution of the linearised convex SDP. The Sequential Convex Programming stops when the distance between the new linearised point and the solution obtained is small enough

$$||\boldsymbol{\theta_{0,i+1}} - \boldsymbol{\theta_i^*}(\boldsymbol{\theta_{0,i}})|| < \epsilon. \tag{7.22}$$

The choice of the step-size plays an important role in the SCP performance. Two main techniques can be used for determining the step-size α in Sequential Convex Programming problems:

- *Line-search methods:* the step-size is the variable of a line-search optimisation (see Bertsekas [5]). Assuming the initial point $\boldsymbol{\theta_0}$ and its SDP solution $\boldsymbol{\theta}^*(\boldsymbol{\theta_0})$, the initial point is updated following (7.21), where the step-size α is computed using the line-search

$$\alpha = \min_{\alpha \in [0,s]} \bar{J}((1-\alpha)\boldsymbol{\theta_0} + \alpha\boldsymbol{\theta}^*(\boldsymbol{\theta_0})) + \boldsymbol{\lambda}\mathbf{h}((1-\alpha)\boldsymbol{\theta_0} + \alpha\boldsymbol{\theta}^*(\boldsymbol{\theta_0})), \tag{7.23}$$

  where $\bar{J}$ is the cost function, $\boldsymbol{\lambda}$ is the vector with the Lagrangian multipliers of the equality constraints, $\mathbf{h}$ is the absolute value of the original nonlinear equality constraints $\mathbf{h} = |\mathbf{A_{nonlin}o} + \mathbf{d_{nonlin}(o)} - \mathbf{b_{nonlin}}|$, and s bounds the step-size $0 < s < 1$ to avoid large steps at a time.

  This method requires the computing another optimisation problem in addition to the linearised convex Semi-Definite Positive program. Also, the line-search problem is a non-convex problem which is generically difficult to solve. However this problem is a single-variable unconstrained one which can be solved in a relatively efficient way. Nevertheless, it will still require an extra computational effort which we wish to avoid.

- *Trust-region methods:* another option is the use of the so-called trust-regions (see Conn et al. [14]). In this method the step-size α is fixed and there is no need of solving another optimisation problem, instead a constraint is added to the linearised convex SDP problem to bound the solution $\boldsymbol{\theta}^*(\boldsymbol{\theta_0})$. The idea is guarantee a more precise linearisation by bounding the solution of the convex problem to a trust-region around the initial point

$$\left(\boldsymbol{\theta}^*(\boldsymbol{\theta_0}) - \boldsymbol{\theta_0}\right) \in \mathcal{T}.$$

  The most common trust region used, which will be also considered here, is to constraint the solution to a box around the linearised point. Therefore we constrain the solution vector element-wise

$$|\theta^*_{k+t}(\boldsymbol{\theta_0}) - \theta_{0,k+t}| \leq \tau \quad \Leftrightarrow \quad \begin{bmatrix} \theta^*_{k+t}(\boldsymbol{\theta_0}) \\ -\theta^*_{k+t}(\boldsymbol{\theta_0}) \end{bmatrix} \leq \begin{bmatrix} \tau + \theta_{0,k+t} \\ \tau - \theta_{0,k+t} \end{bmatrix}, \quad t = \{1, N-1\}. \tag{7.24}$$

  Then, adding the constraint (7.24), we can run the SDP algorithm and update the linearised point using a fixed step-size in (7.21) until the stopping criterion is met (7.22). This method adds no extra computational power and it guarantees that each optimisation solution is close enough to the linearised point, i.e., the solution is inside the trust-region where the linearisation is considered to be a good approximation of the non-linear function. The trust-region should be tuned (see Boyd [9]) since

  - τ too large leads to poor approximations and possibly a bad choice for $\boldsymbol{\theta}^*(\boldsymbol{\theta_0})$,
  - τ too small ensures good approximations but the progress gets slower.

Fixing an initial headings-vector $\boldsymbol{\theta_0} = [\theta_{0,k+1}, ..., \theta_{0,k+N-1}]^\top$, we linearise the trigonometric functions *cosine* and *sine* using the first Taylor expansion

$$\begin{cases} \cos\theta_{k+t} \approx \cos\theta_{0,k+t} - \sin\theta_{0,k+t}\left(\theta_{k+t} - \theta_{0,k+t}\right) \\ \sin\theta_{k+t} \approx \sin\theta_{0,k+t} + \cos\theta_{0,k+t}\left(\theta_{k+t} - \theta_{0,k+t}\right) \end{cases}, \quad t \in \{1, ..., N-1\},$$

where the linearisation precision is inversely proportional to the distance towards the linearised point $|\theta_t - \theta_{0,t}|$. In our case, it means that the turns of our fixed-wing should be smooth and the heading rate u should be bounded to a short interval.

We can rewrite the non-linear equality constraint as another linear equality constraint

$$\mathbf{A_{nonlin}o} + \mathbf{d_{nonlin}(o)} = \mathbf{b_{nonlin}} \quad \Rightarrow \quad \mathbf{A_{eq2}}(\boldsymbol{\theta_0})\mathbf{o} = \mathbf{b_{eq2}}(\boldsymbol{\theta_0}), \tag{7.25}$$

with

$$\begin{aligned} \mathbf{A_{eq2}}(\boldsymbol{\theta_0}) &= \begin{bmatrix} \mathbf{Q_3} & & & \\ \mathbf{Q_4}(\theta_{0,k+1}) & \mathbf{Q_3} & & \\ & \ddots & \ddots & \\ & & \mathbf{Q_4}(\theta_{0,k+N-1}) & \mathbf{Q_3} \end{bmatrix}, \\ \mathbf{Q_3} &= \begin{bmatrix} \mathbf{0}_{2\times 1} & \mathbf{I}_2 & \mathbf{0}_{2\times 1} \end{bmatrix}, \mathbf{Q_4}(\theta_{0,k+t}) = \begin{bmatrix} \mathbf{0}_{2\times 1} & -\mathbf{I}_2 & -V_M \Delta t \begin{bmatrix} -\sin(\theta_{0,k+t}) \\ \cos(\theta_{0,k+t}) \end{bmatrix} \end{bmatrix}, \end{aligned} \tag{7.26}$$

and

$$\mathbf{b_{eq2}}(\boldsymbol{\theta_0}) = \begin{bmatrix} \begin{bmatrix} E_k \\ N_k \end{bmatrix} + V_M \Delta t \begin{bmatrix} \cos\theta_k \\ \sin\theta_k \end{bmatrix} \\ V_M \Delta t \begin{bmatrix} \cos\theta_{0,k+1} + \theta_{0,k+1}\sin\theta_{0,k+1} \\ \sin\theta_{0,k+1} - \theta_{0,k+1}\cos\theta_{0,k+1} \end{bmatrix} \\ \vdots \\ V_M \Delta t \begin{bmatrix} \cos\theta_{0,k+N-1} + \theta_{0,k+N-1}\sin\theta_{0,k+N-1} \\ \sin\theta_{0,k+N-1} - \theta_{0,k+N-1}\cos\theta_{0,k+N-1} \end{bmatrix} \end{bmatrix}. \tag{7.27}$$

Thus, our linearised local tracking problem becomes a convex Semi-Definite Positive program.

Problem 7 (Fixed-wing local linearised Tracking Problem in a SDP form)**.**

Given the optimisation variables $\mathbf{U} = u_t, \mathbf{X} = \mathbf{x}_{t+1} = \begin{bmatrix} E_{t+1} \\ N_{t+1} \end{bmatrix}, \boldsymbol{\theta} = \theta_{t+1}, t = \{k, ..., k+N-1\}$, *stacked as*

$$\mathbf{o} = \begin{bmatrix} u_t \\ \mathbf{x}_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} u_t \\ E_{t+1} \\ N_{t+1} \\ \theta_{t+1} \end{bmatrix}, \quad t = \{k, ..., k+N-1\},$$

a fixed headings vector

$$\boldsymbol{\theta_0} = \begin{bmatrix} \theta_{0,k+1} \\ \vdots \\ \theta_{0,k+N-1} \end{bmatrix},$$

and the cost function

$$\bar{J}(\mathbf{o}) := \mathbf{c}^\top \mathbf{X} + \beta, \tag{7.28}$$

we define the problem

$$\min_{\mathbf{o}} \bar{J}(\mathbf{o}) \text{ s.t.}$$
$$\begin{cases} \mathbf{A_{eq}o} = \mathbf{b_{eq}}, \\ \mathbf{A_{ineq}o} \leq \mathbf{b_{ineq}}, \\ \mathbf{A_{eq2}}(\boldsymbol{\theta_0})\mathbf{o} = \mathbf{b_{eq2}}(\boldsymbol{\theta_0}), \\ \mathbf{M}^q = \begin{bmatrix} \mathbf{I}_3 & (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^{q,*}_{k+1,\text{pos}}) \\ (\mathbf{x}_{k+2,\text{pos}} - \mathbf{t}^{q,*}_{k+1,\text{pos}})^\top & R^2 \end{bmatrix} \succeq 0, \quad q = \{1, ..., m\}, \\ \begin{bmatrix} \mathbf{Q_X}^{-1} & \mathbf{X} \\ \mathbf{X}^\top & \beta_{\mathbf{X}} \end{bmatrix} \succeq 0, \quad \beta_{\mathbf{X}} \geq 0, \quad \begin{bmatrix} \mathbf{Q_U}^{-1} & \mathbf{U} \\ \mathbf{U}^\top & \beta_{\mathbf{U}} \end{bmatrix} \succeq 0, \quad \beta_{\mathbf{U}} \geq 0, \end{cases} \tag{7.29}$$

where $\mathbf{c} = [\mathbf{c_1} + \varphi\mathbf{c_2}]$, $\beta = \beta_{\mathbf{X}} + \beta_{\mathbf{U}}$, $\mathbf{Q_X} = [m\mathbf{Q_1} + m\varphi\mathbf{Q_2}]$ *and* $\mathbf{Q_U} = [\rho\mathbf{I}_N]$. *The matrices* $\mathbf{Q_1}, \mathbf{Q_2}, \mathbf{c_1}, \mathbf{c_2}, \mathbf{A_{eq}}, \mathbf{b_{eq}}, \mathbf{A_{ineq}}, \mathbf{b_{ineq}}, \mathbf{A_{eq2}}(\boldsymbol{\theta_0}), \mathbf{b_{eq2}}(\boldsymbol{\theta_0})$ *are defined in* (7.12), (7.13), (7.14), (7.18), (7.19), (7.26), (7.27). *The worst case scenario* $\mathbf{t}^{q,*}_{k+1,\text{pos}}, q = \{1, ..., m\}$ *defined in* (7.20).

The parameters of the problem are

N	*predictive horizon*
m	*number of targets*
ρ, φ	*performance weighting factors*
$V_M, u_{\max}, \bar{h}$	*agents' specifications*
$W_{\max}$	*targets' maximum velocity*
Δt	*sampling period*
R	*agents' maximum sensing range.*

We implemented a Trust-region Sequential Convex Programming to solve the linearised local SDP (Problem 7). Defining the parameters

α	step-size
τ	trust-region
ϵ	stopping criterion
$i_{\max}$	maximum number of iterations,

we implement our SCP algorithm.

Algorithm 3 (Sequential Convex Programming (SCP))**.**

- for every time-step k of the Tracking Problem 7, initiate the SCP with $i := 0$

- given a step-size α *and a trust-region* τ

- given a stopping criterion ϵ *and a maximum number of iterations* $i_{\max}$

- defining the starting point with the current heading $\boldsymbol{\theta}_{\mathbf{0,0}} := \mathbf{1}_{(N-1)\times 1} \otimes \theta_k$

repeat

1. Determine the optimal $\boldsymbol{\theta}^*(\boldsymbol{\theta}_{\mathbf{0,i}})$ *solving the linearised local tracking Problem 7, constrained to the trust-region inequality constraint* (7.24)

2. Update the linearisation point $\boldsymbol{\theta}_{\mathbf{0,i+1}} = (1-\alpha)\boldsymbol{\theta}_{\mathbf{0,i}} + \alpha\boldsymbol{\theta}^*_{\mathbf{i}}(\boldsymbol{\theta}_{\mathbf{0,i}})$, $i = i + 1$

until *the stopping criterion is satisfied* $||\boldsymbol{\theta}_{\mathbf{0,i}} - \boldsymbol{\theta}^*_{\mathbf{i-1}}(\boldsymbol{\theta}_{\mathbf{0,i-1}})|| < \epsilon \ \vee \ i = i_{\max}$.

Note that this Sequential Convex Programming problem becomes fully convex, it solves iteratively the linearised fixed-wing SDP (defined with a convex cost function subjected to convex inequality constraints

and linear equality constraints). Once more, the solver SeDuMi from the MATLAB toolbox YALMIP (Lofberg [31]) is used to compute each SDP optimisation.

Parameter	Value	Description
N	3	finite horizon
m	10	number of targets
ρ, φ	0.01, 0.01	performance weighting factors
$V_M, u_{\max}, h$	2, 0.8, 10	agents' specifications
$W_{\max}$	3	targets' maximum velocity
Δt	1	time-step (in seconds)
R	50	agents' maximum sensing range
k	80	number of time-steps of the simulation

Table 7.2: Fixed-wing local tracking SDP parameters.

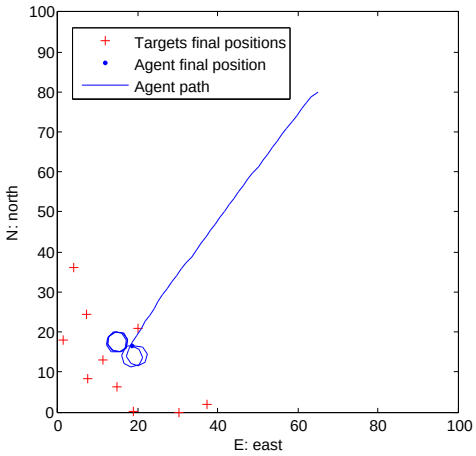

Figure 7.5: Fixed-wing non-linear local tracking SDP path.

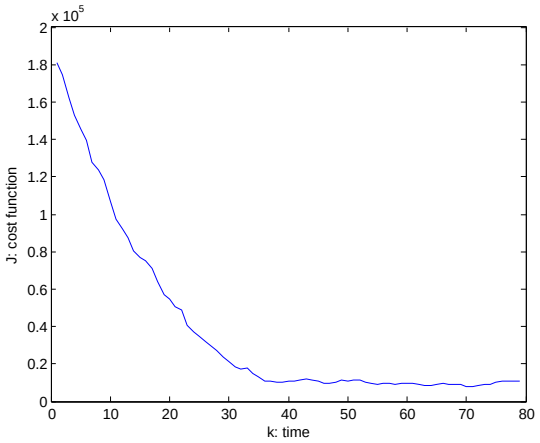

Figure 7.6: Fixed-wing non-linear local tracking SDP performance.

The same simulation presented in the local TP for a quadrotor with multiple moving targets is simulated here for a fixed-wing. The simulation parameters are presented in Table 7.2, and the results using the original non-linearised non-convex problem using the MatLab tool fmincon are presented in the Figures 7.5,7.6.

max # iterations	trust-region	time per time-step	mean of stationary performance
$i_{\max} = 1$	$\tau = 0.8$	0.5 seconds	$\bar{J} = 9811$
$i_{\max} = 2$	$\tau = 0.7$	0.8 seconds	$\bar{J} = 9765$
$i_{\max} = 3$	$\tau = 0.6$	1.3 seconds	$\bar{J} = 9640$
$i_{\max} = 4$	$\tau = 0.4$	1.8 seconds	$\bar{J} = 9632$
$i_{\max} = 5$	$\tau = 0.4$	2.3 seconds	$J = 9611$

Table 7.3: Fixed-wing SCP parameters and results.

The results using our Sequential Convex Programming to solve the linearised SDP problem are presented in the Figures 7.7,7.8. Fixing the stopping criterion $\epsilon = 0.01$ rad, different values for the step-size α are tested, ending up with $\alpha = 0.3$ as the more suitable one. Then, different values for the trust-region $\tau(i_{\max})$ as a function of the maximum number of iterations $i_{\max}$ are simulated. I.e., for a given maximum number of iterations, we pick the most suitable value for the trust-region. The list of the parameters used in the SCP, and their respective results, are summarized in Table 7.3. These simulations are obtained

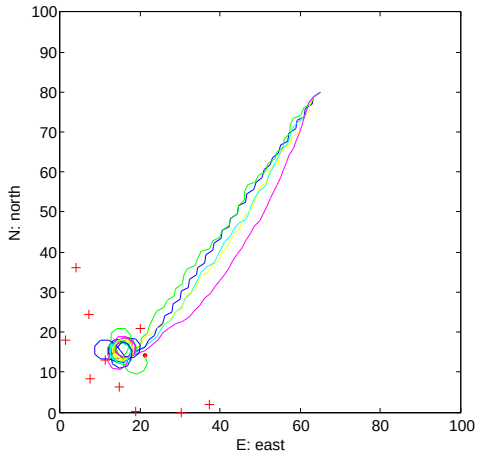

Figure 7.7: Fixed-wing linearised local tracking SCP paths: the red crosses represent the targets final positions and the coloured lines the agent path obtained with different SCP parameters.

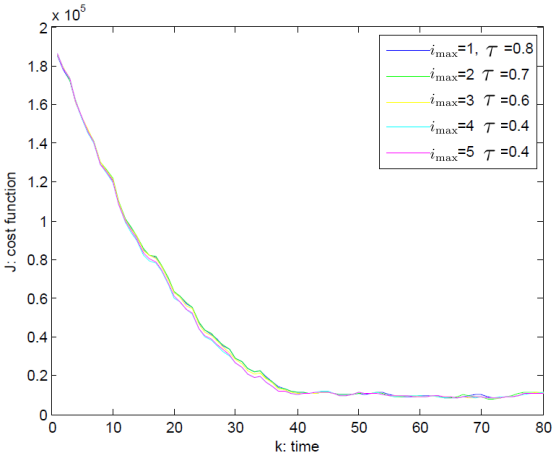

Figure 7.8: Fixed-wing linearised local tracking SCP performances.

using our Sequential Convex Programming Algorithm 3 and SeDuMi 1.3 in `MATLAB R2012a` to solve the local linearised fixed-wing Tracking Problem.

Comparing the non-linear and the SCP results, the performance and the path are quite similar. The Sequential Convex Programming has more oscillations due to the approximations done in the linearisation, but it offers an easier implementation due to its convex SDP formulation.

The Sequential Convex Programming has a similar time complexity to the SDP, i.e., given by $\mathcal{O}(\max\{io^3, io^2c\})$, where o is the number of optimisation variables $o = 4N$, c the number of constraints $c = 15N + 8m + 2$, and i the number of SDP iterations. Therefore, the time complexity depends mainly on the finite horizon N and, obviously, on the number of SDP iterations i. The dependence with the number of targets is linear, i.e., our algorithm is $\mathcal{O}(m)$. Since this problem is defined locally for each agent, there is no dependence with the number of agents: each agent solves its own tracking problem in a parallel fashion.

Regarding the different SCP simulations, the cost-function decreases with the number of iterations. However, the time spent per time-step increases almost proportionally with $i_{\max}$. The most interesting trade-off are the ones obtained with a maximum number of iterations of two or three. We will consider the case with $i_{\max} = 2, \tau = 0.7, \alpha = 0.3, \epsilon = 0.01$ which runs with a rate of 0.8 seconds per time-step.

Chapter 8

Distributed two-layer algorithms

In this chapter we define the final agent algorithms for our two-layer approach: using synchronous-reliable communication (Synchronous-distributed Algorithm 4) and using a asynchronous-unreliable communication (Asynchronous-distributed Algorithm 5). Equivalence between our novel two-layer approach and the centralised approach is proved under some conditions. The implementation with heterogeneous teams and emergency manoeuvres are also discussed.

8.1 Agent algorithms

In this section the algorithms to be implemented on each agent are presented. The pseudo-code of our two-layer approach, which has an Assignment Problem and a Tracking Problem component, is implemented locally on each agent.

8.1.1 Synchronous-distributed two-layer algorithm

Considering a synchronous-reliable communication protocol, the synchronous-distributed two-layer algorithm can be implemented. The pseudo-code scheme of each agent i is presented in Figure 8.1, and described in Algorithm 4.

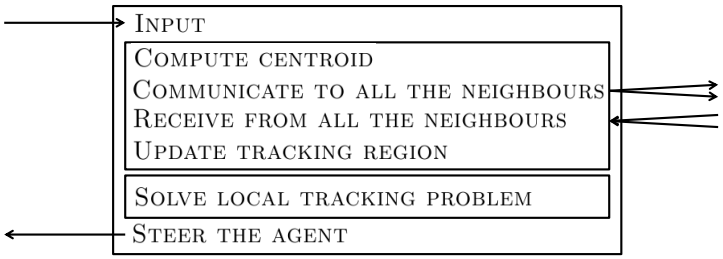

Figure 8.1: Synchronous-distributed two-layer algorithm of Agent i.

The synchronous-distributed two-layer algorithm receives three inputs:

1. the current state of the agent $\mathbf{x}_k^i$ obtained using a GPS/INS sensor (Assumption 1),
2. the current sensing measurements of all the targets inside the tracking region $\{\mathbf{t}_k^q\}_{q=1}^m : \mathbf{t}_k^q \in v_k^i$ which are within the agent's sensing range R^i (2.16),
3. the set of neighbours $j = \{1, ..., n^i\}$ (Definition 16).

Regarding the Assignment Problem, the agent computes the targets-centroid of its tracking region $C(v_k^i)$. Then, the agent sends this value to all its neighbours. Finally, for every neighbour j, it receives back its targets-centroid $C(v_k^j)$ and uses it to update its own tracking region.

Using its current tracking region v_k^i, the Tracking Problem is solved differently for a quadrotor and a fixed-wing. For a quadrotor the local SDP is solved, while for a fixed-wing a Sequential Convex Programming solves the local linearisation SDP iteratively. Finally, we steer the agent using the first control variable solution $\mathbf{u}_k$ of the Tracking Problem.

Algorithm 4 (Synchronous-distributed two-layer algorithm of Agent i)**.**

repeat

1. INPUT: $\mathbf{x}_k^i, \quad \{\mathbf{t}_k^q\}_{q=1}^m : \mathbf{t}_k^q \in v_k^i, \quad$ *set of neighbours* $j = \{1, ..., n^i\}$
2. COMPUTE CENTROID: $C(v_k^i)$ *(6.13 of Chapter 6)*
3. COMMUNICATE TO ALL THE NEIGHBOURS: $C(v_k^i)$
4. RECEIVE FROM ALL THE NEIGHBOURS: $C(v_k^j), j = \{1, ..., n^i\}$
5. UPDATE TRACKING REGION: v_{k+1}^i *using* $C(v_k^j), j = \{1, ..., n^i\}$ *(6.14a of Chapter 6)*
6. SOLVE LOCAL TRACKING PROBLEM: *obtain the control variables* $\mathbf{u}$

   Quadrotor: Solve the SDP Problem 6 (Chapter 7)

   Fixed-wing: Run the SCP Algorithm 3, solving the linearised SDP Problem 7 (Chapter 7)
7. STEER THE AGENT: *apply first control variable* $\mathbf{u}_k$

until *the simulation is stopped.*

8.1.2 Asynchronous-distributed two-layer algorithm

Considering an asynchronous-unreliable communication protocol, the asynchronous-distributed two-layer algorithm can be implemented. The pseudo-code scheme of each agent i is presented in Figure 8.2, and described in Algorithm 5.

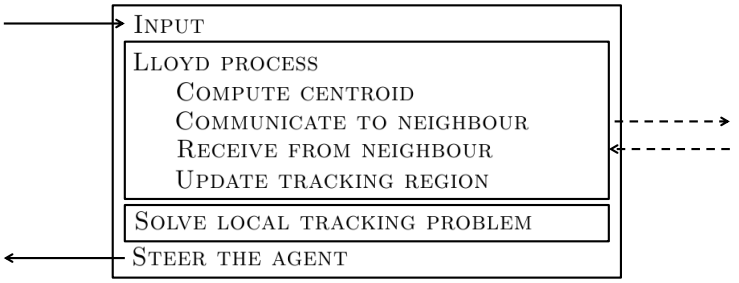

Figure 8.2: Asynchronous-distributed two-layer algorithm of Agent i.

The asynchronous-distributed two-layer algorithm receives the same three inputs as Algorithm 4: $\mathbf{x}_k^i$, $\{\mathbf{t}_k^q\}_{q=1}^m : \mathbf{t}_k^q \in v_k^i$ and the set of neighbours $j = \{1, ..., n^i\}$.

Regarding the Assignment Problem, the agent calls its Lloyd process, a randomly persistent stochastic process. It decides to "gossip" with one neighbour with a probability of $p_{\text{gossip}} = 1/n$. If the agent does not communicate, it updates its new tracking region $v_{k+1}^i = v_k^i$ and the algorithm continues with the Tracking Problem (step 7.). Otherwise, the agent selects a random neighbour j and computes the targets-centroid of its tracking region $C(v_k^i)$. Then, the agent sends this value to the neighbour j and receives back its targets-centroid $C(v_k^j)$. Finally, it updates its own tracking region using the information received.

Using the tracking region v_k^i, the Tracking Problem is solved as in Algorithm 4, for a quadrotor and a fixed-wing. Then, we steer the agent using the first control variable solution $\mathbf{u}_k$ of the Tracking Problem.

Algorithm 5 (Asynchronous-distributed two-layer algorithm of Agent i)**.**

repeat

1. INPUT: $\mathbf{x}_k^i, \quad \{\mathbf{t}_k^q\}_{q=1}^m : \mathbf{t}_k^q \in v_k^i, \quad$ *set of neighbours* $j = \{1, ..., n^i\}$
2. LLOYD PROCESS: *jump with probability* $(1 - p_{\text{gossip}})$ *to 7., pick a random neighbour* j
3. COMPUTE CENTROID: $C(v_k^i)$ *(6.13 of Chapter 6)*
4. COMMUNICATE TO NEIGHBOUR: $C(v_k^i)$
5. RECEIVE FROM NEIGHBOUR: $C(v_k^j)$
6. UPDATE TRACKING REGION: v_{k+1}^i *(6.14a of Chapter 6)*
7. SOLVE LOCAL TRACKING PROBLEM: *obtain the control variables* $\mathbf{u}$

   Quadrotor: Solve the SDP Problem 6 (Chapter 7)

   Fixed-wing: Run the SCP Algorithm 3, solving the linearised SDP Problem 7 (Chapter 7)
8. STEER THE AGENT: *apply first control variable* $\mathbf{u}_k$

until *the simulation is stopped.*

8.1.3 Algorithms discussion

The presented synchronous- and asynchronous-distributed local algorithms implement in each agent our distributed two-layer approach of multi-UAV multi-target tracking problem.

Regarding the communication protocol between neighbour agents, the synchronous-distributed algorithm needs a synchronous and reliable protocol where every pair of neighbour agents communicate with each other at every time-step. The asynchronous-distributed algorithms is implementable with an asynchronous and unreliable protocol where only one pair of neighbour agents communicate in average at each time-step. The required bandwidth is two coordinates (E, N) per message. The Assignment Problem is fully described and analysed in Chapter 6.

Regarding the computational required on-board, it is required for each UAV to have a Semi-Definite Positive program (SDP) solver. Using the solver SeDuMi 1.3 in `MATLAB R2012a`, the Tracking Problem 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. Regardless the vehicle dynamics, the time complexity is linear with the number of targets $\mathcal{O}(m)$, and it does not depend on the number of agents n. The Tracking Problem is derived and analysed in Chapter 7.

Remark 6 (Distributed two-layer algorithms initialisation)**.**

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* n*-partition of* Q *by* $\mathbf{x}_{0,\text{pos}}$ (6.2)*, where each agent* i *is initialised with the tracking region* $v_0^i(\mathbf{x}_{0,\text{pos}}) \subset Q$*.*

8.2 Centralised and distributed two-layer solutions comparison

We have presented two approaches to solve for our multi-target multi-UAV tracking problem. Analysing both problem formulations, we seek to examine under which conditions the two approaches can be consider equivalent or lead to similar results.

1. *a centralised naive approach:* where the problem is formulated in a fully centralised way, requiring a central entity with constant communication with all the agents. The centralised naive approach seeks to minimise the following performance criterion

$$\bar{J} = \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left(\min_{i=1,\dots,n} \{ ||\mathbf{x}_{t+1,\text{pos}}^i - \mathbf{t}_{k,\text{pos}}^q||^2 \} \right) + \rho \sum_{i=1}^{n} (||\mathbf{u}_t^i||^2) \right] + \varphi \sum_{q=1}^{m} \left(\min_{i=1,\dots,n} \{ ||\mathbf{x}_{k+N,\text{pos}}^i - \mathbf{t}_{k,\text{pos}}^q||^2 \} \right).$$

2. *a distributed two-layer approach:* where we divide the problem into a global assignment problem and many local tracking problems, it can be implemented either using a synchronous-reliable communication (distributed algorithm) or with a more generic asynchronous-unreliable protocol (gossip algorithm). The two-layer approach is given by the solution of two problems: the global assignment problem and the local tracking problems. The assignment problem seeks to find the partition $(v^i)_{i=1}^n$ which minimises the *energy functional* $\mathcal{H}(v, \mathbf{y})$. The local tracking problems, one for each subset v^i, steer the agent i to minimise the cost-function $\bar{J}^i$.

$$\begin{cases} \mathcal{H}(v, \mathbf{y}) = \sum_{i=1}^{n} \left(\sum_{\mathbf{t}_{\text{pos}}^q \in v^i} ||\mathbf{y}^i - \mathbf{t}_{\text{pos}}^q||^2 \right), \\ \bar{J}^i = \sum_{t=k}^{k+N-1} \left[\sum_{\mathbf{t}_{k,\text{pos}}^q \in v^i} \left(||\mathbf{x}_{t+1,\text{pos}}^i - \mathbf{t}_{k,\text{pos}}^q||^2 \right) + \rho ||\mathbf{u}_t^i||^2 \right] + \varphi \sum_{\mathbf{t}_{k,\text{pos}}^q \in v^i} \left(||\mathbf{x}_{k+N,\text{pos}}^i - \mathbf{t}_{k,\text{pos}}^q||^2 \right). \end{cases}$$

Single agent case

For the one-agent scenario, it is obvious to see that both formulations are exactly the same. With a single agent there is no partition of the space $v^1 \equiv Q$ and both implementations seek to minimise

$$\bar{J} \equiv \bar{J}^1 = \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left(||\mathbf{x}_{t+1,\text{pos}}^1 - \mathbf{t}_{k,\text{pos}}^q||^2 \right) + \rho ||\mathbf{u}_t^1||^2 \right] + \varphi \sum_{q=1}^{m} \left(||\mathbf{x}_{k+N,\text{pos}}^1 - \mathbf{t}_{k,\text{pos}}^q||^2 \right).$$

Steady-state solution

Considering static targets, we can compare the steady-state results defined as follows.

Definition 20 (Steady-state solution of a static Tracking Problem)**.** *Considering a static targets Tracking Problem with quadrotor agents, the steady-state solution is the static 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. The fixed-wings will instead tend to move in circles around the *steady-state solution* that could be obtained using quadrotors. Therefore, analysing the centralised and the two-layer steady-state solutions with respect to the final quadrotors' position, we prove the following.

Theorem 4 (Steady-state solution for static targets scenario)**.** *Considering the centralised naive problem and the distributed two-layer relaxation, under a static targets scenario, both approaches optimise the same functional at steady-state, thus they have the same steady-state global optimum.*

Proof. For the centralised implementation, the optimal *steady-state solution* is given by the minimum of the steady-state performance criterion

$$\min_{\mathbf{x}} \bar{J}_{ss} = \sum_{t=k}^{k+N-1} \left[\sum_{q=1}^{m} \left(\min_{i=1,...,n} \{||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} \right) + \rho \sum_{i=1}^{n} (||\mathbf{u}^i_{ss}||^2) \right] + \varphi \sum_{q=1}^{m} \left(\min_{i=1,...,n} \{||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} \right) \Leftrightarrow$$

$$\Leftrightarrow \min_{\mathbf{x}} \bar{J}_{ss} = (N+\varphi) \sum_{q=1}^{m} \left(\min_{i=1,...,n} \{||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} \right) \Leftrightarrow \min_{\mathbf{x}} \bar{J}'_{ss} = \sum_{q=1}^{m} \left(\min_{i=1,...,n} \{||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} \right),$$

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. Note that since the positions of the agents do not change, the input is null $\mathbf{u}^i_{ss} = \mathbf{0}, \;\; i = 1, ..., n$.

Now, we split the space Q into n tracking regions $(v^i)_{i=1}^n$ using a Voronoi partition (Definition 11). Picking the steady-state position of the agents $(\mathbf{x}^i_{ss,\text{pos}})_{i=1}^n$ as the Voronoi centroids, we divide the space into n regions that, by definition, satisfy

$$\mathbf{t}^q_{\text{pos}} \in v^i \Leftrightarrow \arg\min_{j=1,...,n} \{||\mathbf{x}^j_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} = i,$$

i.e., if a target q is in the tracking region of an agent i, it means that the agent i is the closest agent to the target q. Then, we reformulate our cost-function using the Voronoi partition $v^i(\mathbf{x}_{ss,\text{pos}})$ (Definition 11)

$$\min_{\mathbf{x}} \bar{J}'_{ss} = \sum_{q=1}^{m} \left(\min_{i=1,...,n} \{||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2\} \right) \Leftrightarrow \min_{\mathbf{x}} \bar{J}'_{ss} = \sum_{i=1}^{n} \left(\sum_{\mathbf{t}^q_{\text{pos}} \in v^i(\mathbf{x}_{ss,\text{pos}})} ||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right). \quad (8.1)$$

For the two-layer implementation, the agents' steady-state position is given by the optimal value of local performance criterion

$$\min_{\mathbf{x}^i_{ss,\text{pos}}} \bar{J}^i_{ss} = \sum_{t=k}^{k+N-1} \left[\sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \left(||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right) + \rho ||\mathbf{u}^i_{ss}||^2 \right] + \varphi \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \left(||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right) \Leftrightarrow$$

$$\Leftrightarrow \min_{\mathbf{x}^i_{ss,\text{pos}}} J_{iss} = (N+\varphi) \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \left(||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right) \Leftrightarrow \min_{\mathbf{x}^i_{ss,\text{pos}}} J'_{iss} = \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \left(||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right). \quad (8.2)$$

where the steady-state partition v^i is the solution of the global assignment problem, given by the minimum of the *energy functional*

$$\min_{v,\mathbf{y}} \mathcal{H}(v, \mathbf{y}) = \sum_{i=1}^{n} \left(\sum_{\mathbf{t}^q_{\text{pos}} \in v^i} ||\mathbf{y}^i - \mathbf{t}^q_{\text{pos}}||^2 \right). \quad (8.3)$$

Comparing the local tracking steady-state solutions (8.2) and the minimum of the solutions of the global assignment problem (8.3), we can realise that the steady-state solutions $\mathbf{x}^i_{ss,\text{pos}}$ indeed tend to the centre-of-mass $\mathbf{y}^i$ of the energy functional. Therefore, the steady-state solution satisfies $\mathbf{x}^i_{ss,\text{pos}} = \mathbf{y}^i, \;\; i = 1, ..., n$, and the two-layer performance criterion is given by

$$\min_{v,\mathbf{x}} \mathcal{H}(v, \mathbf{x}_{ss,\text{pos}}) = \sum_{i=1}^{n} \left(\sum_{\mathbf{t}^q_{\text{pos}} \in v^i} ||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \right),$$

which is achieved considering the Voronoi partition of the space (Proposition 3)

$$\min_{\mathbf{x}} \mathcal{H}(\mathbf{x}_{ss,\text{pos}}) = \sum_{i=1}^{n} \Bigg(\sum_{\mathbf{t}^q_{\text{pos}} \in v^i(\mathbf{x}_{ss,\text{pos}})} ||\mathbf{x}^i_{ss,\text{pos}} - \mathbf{t}^q_{\text{pos}}||^2 \Bigg). \tag{8.4}$$

Comparing the centralised performance criterion (8.1) and the assignment performance criterion (8.4), we conclude that the optimal *steady-state solution* is identical. □

The Theorem 4 says that the centralised naive and the distributed two-layer problems have the same optimal steady-state solutions (Definition 20). However, we can not ensure to have the same results using the two different implementations, both distributed coverage algorithms (Algorithm 1 and Algorithm 2) often end up in local minima, missing the global optimum.

Considerations for moving targets

Considering moving targets, there is no *steady-state solution*. The agents are constantly adapting to the new targets' positions. The centralised performance can be rearranged considering an optimal partition for every time-step along the predictive horizon $t = \{k, ..., k+N-1\}$. This optimal partition is given by the Voronoi partition using the position of the agents at each time step as the Voronoi centroids $(v^i(\mathbf{x}_{t+1,\text{pos}}))_{i=1}^n$, $t = k, ..., k+N-1$. The centralised performance criterion becomes

$$\min_{\mathbf{x}} \bar{J} = \sum_{i=1}^{n} \bar{J}^i(\mathbf{x}),$$

$$\bar{J}^i = \sum_{t=k}^{k+N-1} \Bigg[\sum_{\mathbf{t}^q_{k,\text{pos}} \in v^i(\mathbf{x}_{t+1\text{pos}})} \Big(||\mathbf{x}^i_{t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \Big) + \rho ||\mathbf{u}^i_t|||^2 \Bigg] + \varphi \sum_{\mathbf{t}^q_{k,\text{pos}} \in v^i(\mathbf{x}_{k+N,\text{pos}})} \Big(||\mathbf{x}^i_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \Big). \tag{8.5}$$

Comparing the centralised performance criterion per agent (8.5), with the local performance criterion of the two-layer approach

$$\bar{J}^i = \sum_{t=k}^{k+N-1} \Bigg[\sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \Big(||\mathbf{x}^i_{t+1,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \Big) + \rho ||\mathbf{u}^i_t||^2 \Bigg] + \varphi \sum_{\mathbf{t}^q_{\text{pos}} \in v^i} \Big(||\mathbf{x}^i_{k+N,\text{pos}} - \mathbf{t}^q_{k,\text{pos}}||^2 \Big), \tag{8.6}$$

we can realise that the only difference is in the partition $(v^i)_{i=1}^n$ chosen for each case

1. *centralised naive approach partition:* every time step, n partitions along the predictive horizon are computed centrally. These partitions divide the set of targets by their closest agent, for every future agents' positions
$$(v^i_t)_{i=1}^n = (v^i(\mathbf{x}_{t,\text{pos}}))_{i=1}^n, \quad t = k+1, ..., k+N. \tag{8.7}$$

2. *distributed two-layer approach partition:* every agent has its own *tracking region*, updated every time-step (distributed algorithm) by communicating with its direct neighbours, or updated sporadically (gossip algorithm) by communicating with only one of its direct neighbour agents. These partitions tend to the minimum of the *energy functional*, given the Voronoi partition over the points $\mathbf{y}$

$$\mathcal{H}(v, \mathbf{y}) = \sum_{i=1}^{n} \Bigg(\sum_{\mathbf{t}^q_{\text{pos}} \in v^i} ||\mathbf{y}^i - \mathbf{t}^q_{\text{pos}}||^2 \Bigg) \Rightarrow \mathcal{H}(\mathbf{y}) = \sum_{i=1}^{n} \Bigg(\sum_{\mathbf{t}^q_{\text{pos}} \in v^i(\mathbf{y})} ||\mathbf{y}^i - \mathbf{t}^q_{\text{pos}}||^2 \Bigg),$$

which we have seen to be equal to the performance criterion of the centralised implementation at steady-state (see Equation (8.1)).

Therefore we can see clearly here the differences between the two implementations. The tracking criterion is the same, however the space partition is done differently. In the centralised approach, the partition is given by the predicted agents' positions over the predictive horizon (8.7). However, in the distributed approach, the partition is done by a different variable $\mathbf{y}$ which tends to the steady-state position of the agents if we consider static targets. Considering moving targets, both the partition variable $\mathbf{y}$ (which follows the "centring and partition" algorithm) and the position of the agents (which tends to the targets' centre-of-mass) have the same optimal position.

Note that, it is very unlikely that both implementations will end up with the same results for the moving targets scenario. However, it is shown that both solutions are related, and the slower the targets move, the closer the agents get to the steady-state solution and the more similar are both implementations.

8.3 Heterogeneous teams

Regarding the use of heterogeneous teams, the proposed two-layer algorithms allow the use of

- *Quadrotors and Fixed-wings combinations:* with our novel two-layer approach the Assignment Problem and the Tracking Problem are solved independently. Once having the Assignment Problem independent of the vehicle dynamics, the agents' model only influences the local Tracking Problems. Thus, our two-layer approach inherently allows the use of different vehicle dynamics.

- *Different sensing capabilities:* the use of agents with different sensing ranges is also possible with the current implementation. In Remark 3 (Chapter 6), the Voronoi n-partitions is relaxed to Power Voronoi n-partitions (Definition 12). These partitions introduce power-factors $(w^i)_{i=1}^n$ which allow to assign more tracking-space to some agents and less to others. These weights can be seen as a measure of the relative sensing-capability of each agent comparing with the others.

8.4 Emergency manoeuvres

To complement our algorithm, we add a some manoeuvres to face two "emergency" situations. Both manoeuvres represent punctual situations that we seek to solve in order to continue with the optimal tracking task. Simulations regarding these manoeuvres can be seen in Chapter 9.

8.4.1 Collision avoidance

The first manoeuvre to add to our algorithm relates to the collision avoidance between our agents. Two agents are in danger of colliding when their position is closer than a given range R_{CA} (8.8).

$$\forall_{i=1,\dots,n} \exists_{j=1,\dots,n, j\neq i} : ||\mathbf{x}^i - \mathbf{x}^j|| < R_{CA}, \tag{8.8}$$

Therefore, when this situation occurs, 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.

Algorithm 6 (Collision Avoidance)**.**

- choose from the two agents in conflict $(i,j) : ||\mathbf{x}^i - \mathbf{x}^j|| < R_{CA}$*, 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 $||\mathbf{x}^i - \mathbf{x}^j|| \geq R_{CA}$

- for the fixed-wings' case, bring the agent i *back to the fixed altitude* $h_i = \bar{h}$

For both agents' dynamics, and after an initial transitory phase in the quadrotors' case, the multi-target multi-UAV tracking is done entirely in a plane with fixed altitude: $\bar{h}$ for fixed-wings and $h_{\min}$ for quadrotors. Therefore, the third space dimension, the altitude, can be successfully used for this collision avoidance manoeuvre. Note that for the fixed-wings' case, we have to bring back the agent to the fixed altitude $\bar{h}$; but for the quadrotors' case, this is done automatically by the algorithm.

8.4.2 Agents refuelling

UAVs are typically flying agents with a short endurance. The use of Low Altitude Short Endurance (LASE) UAVs is considered inexpensive and highly suitable for real applications, however due to their short endurance, intelligence to return for refuelling has been considered (Casbeer et al. [12]).

The problem of allowing agents to refuel is basically the same problem as dealing with a slow dynamic changing number of agents. When one agent needs to refuel, it leaves its mission and we have to compensate in order to run the target-tracking algorithm with one agent less. In a similar way, when one agent returns from refuelling, the number of agents at our disposal increases by one.

Thus, two algorithms will be proposed, an agent departure and an agent arrival algorithm, which simulation results can be seen in Chapter 9.

Algorithm 7 (Agent Departure)**.**

- given a departure agent j

- pick randomly one of the neighbours (agent i*) of the departing agent* j

- rearrange the tracking region $v^i_{k+1} = v^i_k \cup v^j_k$

- remove the agent j *from the set of agents* $n_{k+1} = n_k - 1$

Algorithm 8 (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

$$v^i_{k+1} = v^i_k \cap B(C(\mathbf{x}^i_k), C(\mathbf{x}^j_k))$$
$$v^j_{k+1} = v^i_k \cap B(C(\mathbf{x}^j_k), C(\mathbf{x}^i_k))$$

- add agent j *to the set of agents* $n_{k+1} = n_k + 1$

Chapter 9

Algorithm results

In this chapter our two-layer approach is simulated and compared with the original centralised formulation. First, using static targets, the results using the centralised, synchronous-distributed and asynchronous-distributed are compared for both quadrotor and fixed-wing agents. These simulations, specially using quadrotors, help us analysing the *steady-state solution* (Definition 20) for the three algorithms.

Then, the set of simulations is repeated using slow moving targets (targets moving slower than the agents) and fast moving targets (targets moving faster than the agents). Instead of analysing the steady-state results, using moving targets we are more interested in comparing how the different algorithms adapt to more dynamic environments.

To summarise, the three algorithms implemented are: (1) the centralised naive approach, (2) the two-layer synchronous-distributed approach, (3) the two-layer asynchronous-distributed approach. Using two agent models: (a) quadrotors, (b) fixed-wings. With three different targets' scenarios: (i) nearly static targets $W_{\max} = 0.01$, (ii) slow moving targets $W_{\max} = 1$, (iii) fast moving targets $W_{\max} = 3$. And using the simulation parameters presented in Table 9.1.

Parameter	Value	Description
N	3	finite horizon
n, m	10, 100	number of agents, targets
ρ, ϕ	0.01, 0.01	performance weighting factors
$h_{\min}, v_{\max}, f_{\max}$	10, 2, 2	quadrotors' specifications
$h, V_M, u_{\max}$	10, 2, 0.8	fixed-wings' specifications
$i_{\max}, \tau, \alpha, \epsilon$	2, 0.7, 0.2, 0.01	SCP parameters
$W_{\max}$	0.01 & 1 & 3	different targets' maximum velocity
R	40	agents' sensing range
Δt	1	time-step (in seconds)
k	60	number of time-steps of the simulation

Table 9.1: Final simulations parameters with different targets' velocities.

Note that the maximum quadrotors velocity is $v_{\max} = 2$ and the constant fixed-wings velocity is $V_M = 2$ (Table 9.1). Therefore, the slow (fast) moving targets scenario correspond to assuming that the targets move slower (faster) than the agents.

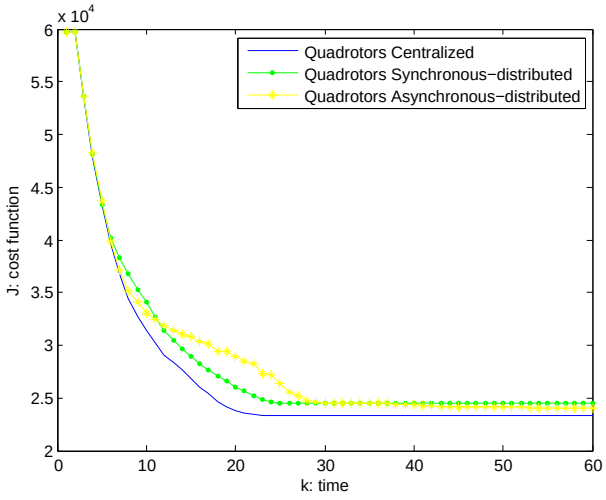

Figure 9.1: Static targets quadrotors performance.

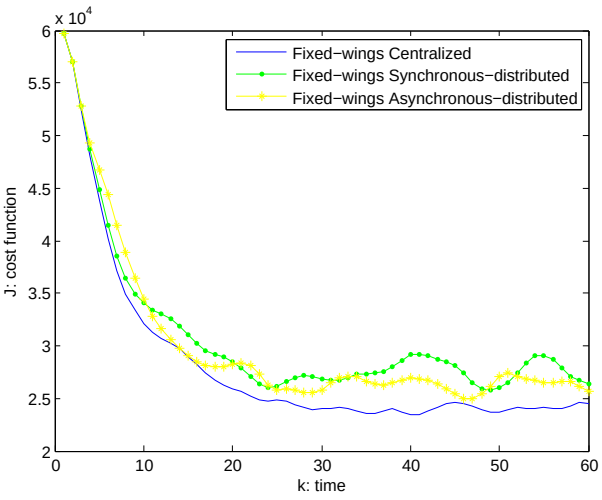

Figure 9.2: Static targets fixed-wings performance.

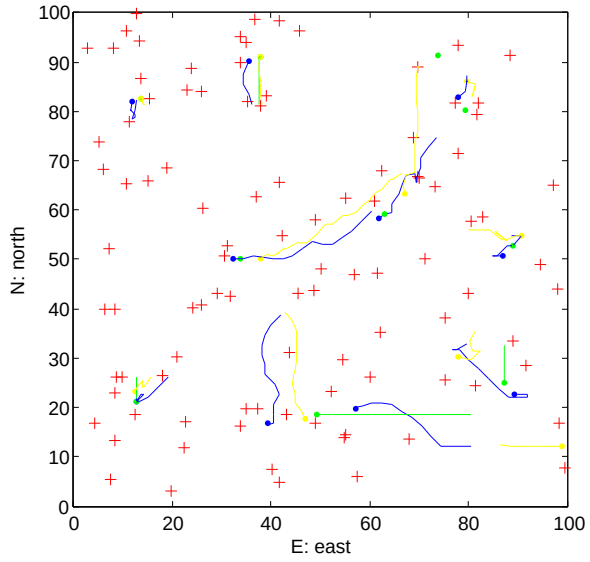

Figure 9.3: Static targets quadrotors path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.1.

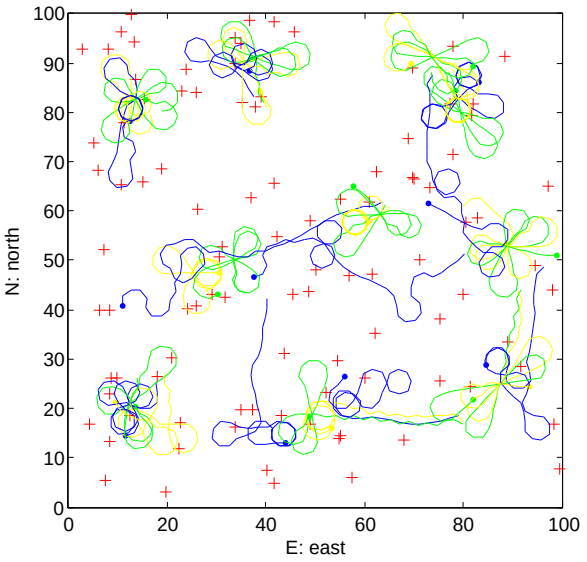

Figure 9.4: Static targets fixed-wings path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.2.

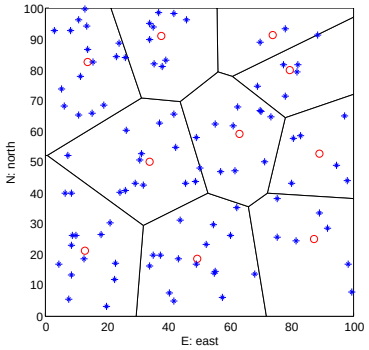

Figure 9.5: Static targets final synchronous-distributed partition (quadrotors, fixed-wings).

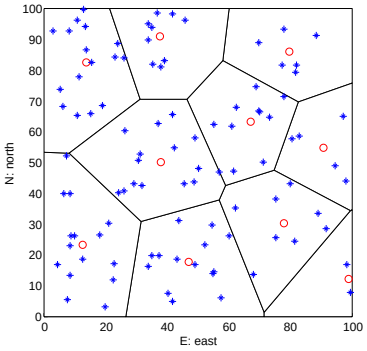

Figure 9.6: Static targets final asynchronous-distributed partition (quadrotors).

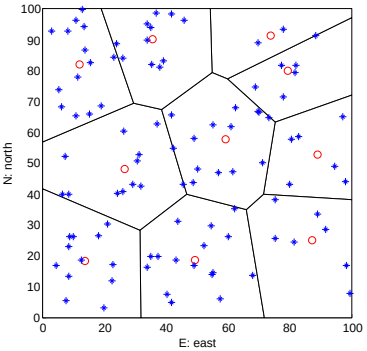

Figure 9.7: Static targets final asynchronous-distributed partition (fixed-wings).

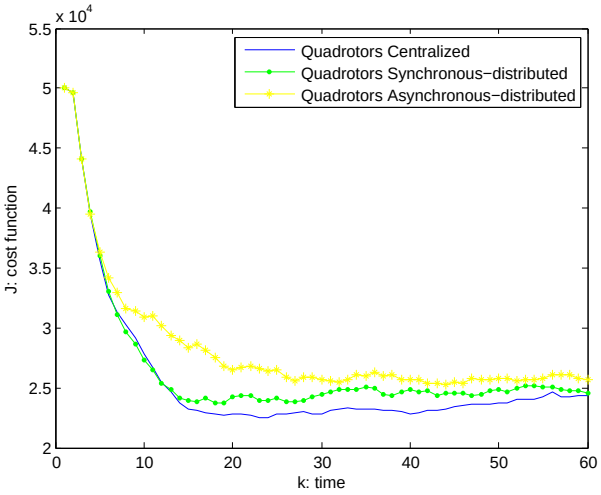

Figure 9.8: Slow targets quadrotors performance.

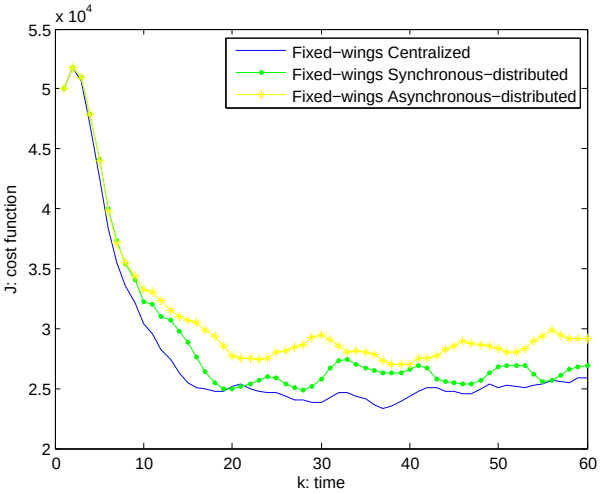

Figure 9.9: Slow targets fixed-wings performance.

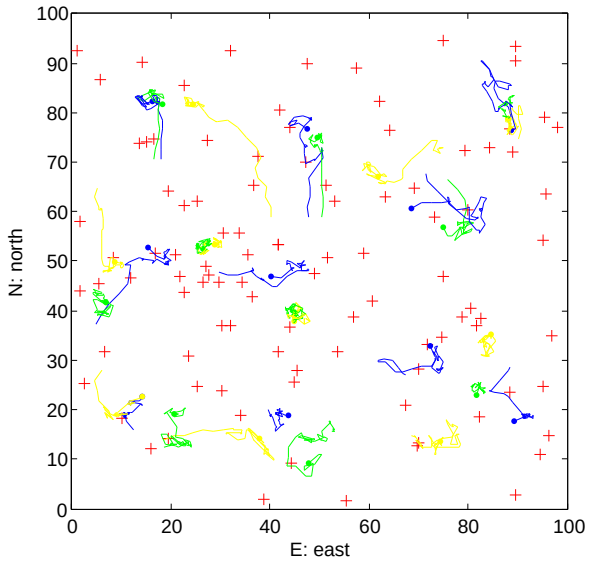

Figure 9.10: Slow targets quadrotors path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.8.

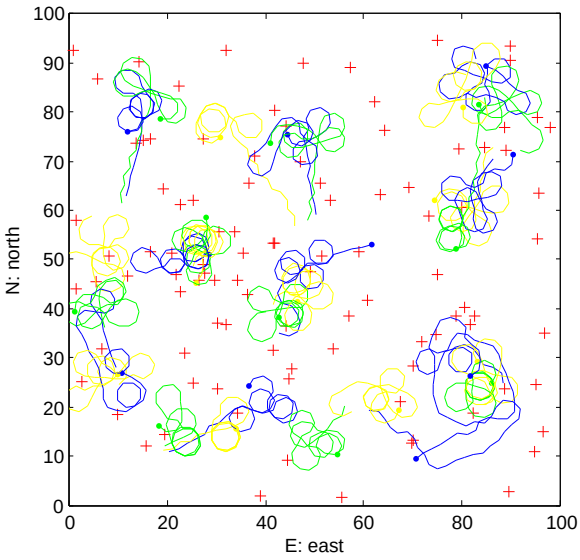

Figure 9.11: Slow targets fixed-wings path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.9.

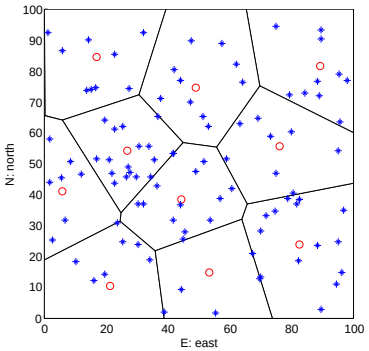

Figure 9.12: Slow targets final synchronous-distributed partition (quadrotors, fixed-wings).

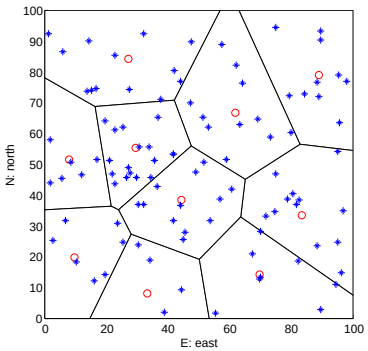

Figure 9.13: Slow targets final asynchronous-distributed partition (quadrotors).

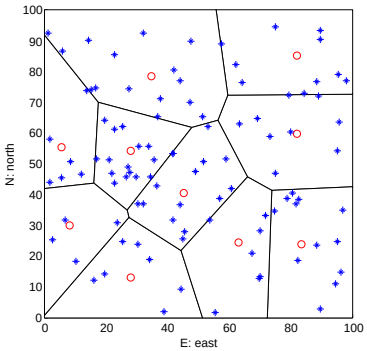

Figure 9.14: Slow targets final asynchronous-distributed partition (fixed-wings).

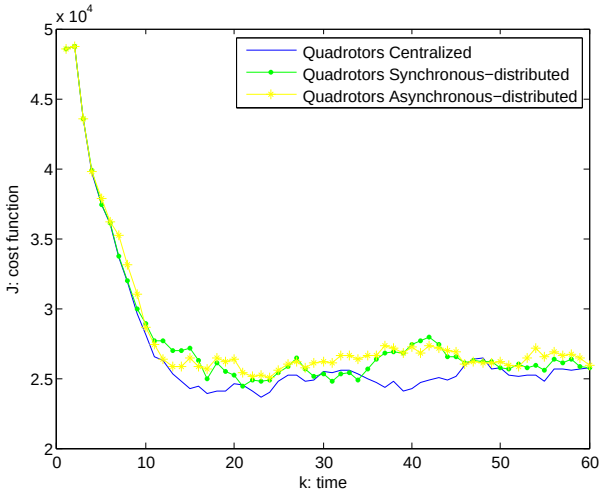

Figure 9.15: Fast targets quadrotors performance.

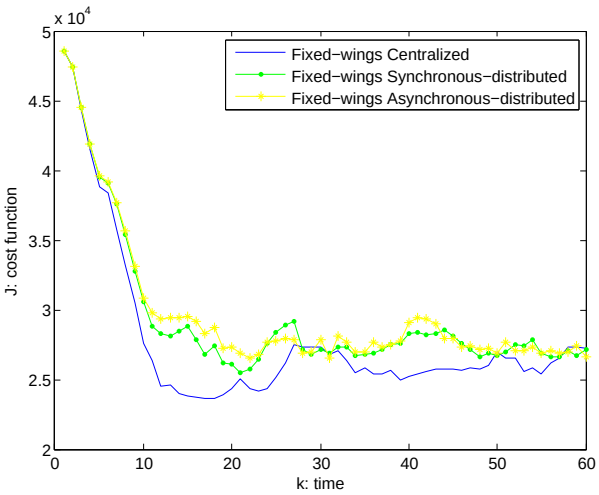

Figure 9.16: Fast targets fixed-wings performance.

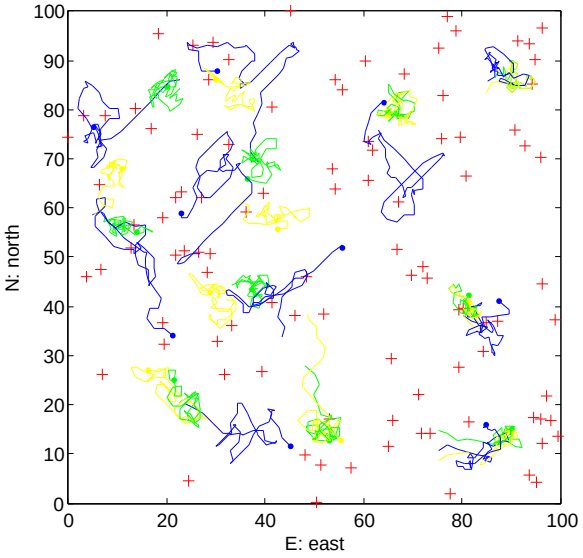

Figure 9.17: Fast targets quadrotors path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.15.

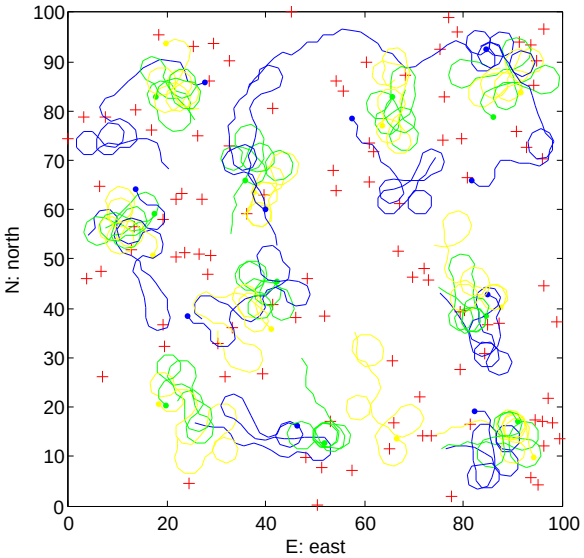

Figure 9.18: Fast targets fixed-wings path: the red crosses represent the targets final positions and the coloured lines the agents path obtained with the algorithms of Figure 9.16.

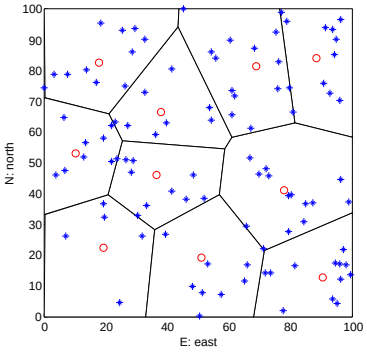

Figure 9.19: Fast targets final synchronous-distributed partition (quadrotors, fixed-wings).

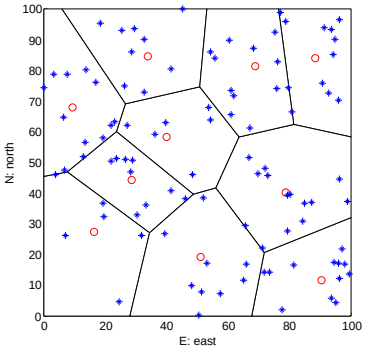

Figure 9.20: Fast targets final asynchronous-distributed partition (quadrotors).

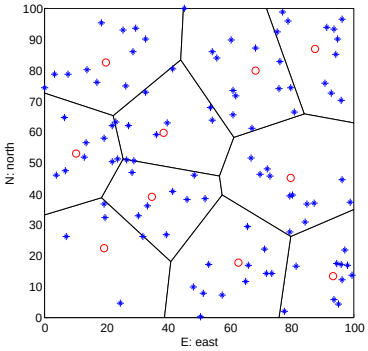

Figure 9.21: Fast targets final asynchronous-distributed partition (fixed-wings).

In Figures 9.1-9.7, we can see the results for the three algorithms using quadrotors and fixed-wings for a nearly static scenario $W_{\max} = 0.01$. In Figures 9.1 and 9.2, we see the cost-function of the three implementations for quadrotors and fixed-wings respectively. Considering the *steady-state solutions* of the quadrotors' case, we realise that the algorithms do not converge to the same optimal. We have a centralised algorithm which finds the best solution, and nearby optima for the two-layer approach (synchronous-distributed and asynchronous-distributed). Note that the asynchronous solution is better than the synchronous one, as expected in Chapter 6.

Despite converging slower, the asynchronous-distributed algorithm is able to avoid local minima better than the synchronous-distributed algorithm which demands more communication. In Figures 9.6 and 9.7, we see the two final partitions obtained with the asynchronous-distributed algorithm (for quadrotors and fixed-wings). They are different from each other due to the randomness of our asynchronous algorithm, and they are both better than the final partition obtained with the synchronous-distributed Algorithm 9.5. Therefore, we motivate that for static and nearly-static cases, the asynchronous-distributed algorithm not only needs less communication but also performs better than the synchronous-distributed algorithm.

The fixed-wings' case has worse results than the quadrotors'. They tend to oscillate around the steady-state solutions while achieving a considerably lower performance. This is clearly caused by the constrained non-holonomic dynamics of these agents. Nevertheless they exhibit the same qualitative results: the centralised algorithm performs better, the asynchronous-distributed algorithm converges slower but it achieves a better local minimum than the synchronous-distributed algorithm which converges faster but stops easier in local minima. In Figures 9.3 and 9.4, the path of the agents is showed.

In Figures 9.8-9.14, we show similar results obtained using the slow moving targets scenario $W_{\max} = 1$, while in Figures 9.15-9.21, we have simulated a fast moving targets scenario $W_{\max} = 3$. For the "slow" case, the targets move slower than the agents, and faster than the agents in the "fast" moving case. The first remark about the moving case is that there is no *steady-state solution* to compare the algorithms or the partitions. However, we can still compare the performance obtained by the different approaches.

For the moving targets' case, the performance obtained using our novel two-layer approach is quite similar to the centralised solution (for both communication schemes). 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. Adding the fact that it can run with an asynchronous and unreliable communication protocol, we opt for the asynchronous-distributed algorithm has our preferred algorithm to implement the two-layer multi-UAV multi-target tracking problem.

Agents	Centralized	Synchronous-distributed	Asynchronous-distributed
Quadrotors	146 seconds	8.2 (0.82) seconds	7.9 (0.79) seconds
Fixed-wings	94 seconds	9.4 (0.94) seconds	9.3 (0.93) seconds

Table 9.2: Simulation time in seconds per time-step (locally).

An average of the time spend in the previous simulations is presented in Table 9.2. Note that our two-layer approach distributes the problem among the agents. Therefore, for the synchronous-distributed algorithm and the asynchronous-distributed algorithm, the computational time spent for computing the tracking solution is done locally by each agent. Thus, for real implementations, we need n machines, one for each of the n agents. Each machine would take around 0.8-0.95 seconds to compute the path planning for the respective agent at each time step, i.e., lower than one second, achieving "real-time".

Regarding the communication needed for this implementations

- *Centralized naive approach:* needs a centralised entity to compute the solution of the centralised

problem formulation, all the agents need to communicate with this central unit at all time-steps;

- *Two-layer synchronous-distributed approach:* each agent computes its own solution, it needs a reliable and synchronous communication protocol where each agent needs to communicate at all time-steps with all its direct neighbours; they only communicate two coordinates with each other $C_E(v^i), C_N(v^i)$;

- *Two-layer asynchronous-distributed approach:* each agent computes its own solution, it can be implemented with a unreliable and asynchronous communication protocol; at each time-step only a random pair of neighbour agents communicate with each other; they also only communicate two coordinates $C_E(v^i), C_N(v^i)$ with each other.

Regarding feasibility, the algorithm constraints are fulfilled during all the simulations using the distributed algorithms. The worst-case scenario analysis guarantees feasibility for the local tracking problems as long as there is a feasible solution. The existence of a feasible solution depends mainly on the space size, number of agents and agents sensing range which analysis is not studied here.

Noise simulations

Considering our two-layer asynchronous-distributed algorithm for a slow targets scenario ($W_{\max} = 1$) using quadrotors, and the parameters presented in Table 9.3, we now considering the noisy sensing model (2.15)

$$\mathbf{z}_k^{i,q} = \mathbf{t}_{k,\text{pos}}^q + \boldsymbol{\zeta}_k^{i,q} ||\mathbf{x}_{k,\text{pos}}^i - \mathbf{t}_{k,\text{pos}}^q||,$$

where $\boldsymbol{\zeta}_k^{i,q} = [\boldsymbol{\zeta}_{k,E}^{i,q}, \boldsymbol{\zeta}_{k,N}^{i,q}, 0]^\top$, and $\boldsymbol{\zeta}_{k,E}^{i,q}, \boldsymbol{\zeta}_{k,N}^{i,q}$ are zero mean Gaussian noises with standard deviation ξ. In Figure 9.22, we can see the effect of the observing/sensing noise in our algorithm.

Parameter	Value	Description
N	3	finite horizon
n, m	10, 100	number of agents, targets
ρ, ϕ	0.01, 0.01	performance weighting factors
$h_{\min}, v_{\max}, f_{\max}$	10, 2, 2	quadrotors' specifications
$W_{\max}$	1	targets' maximum velocity
R	40	agents' sensing range
Δt	1	time-step (in seconds)
k	60	number of time-steps of the simulation

Table 9.3: Noise simulations parameters.

Feasibility is accomplished during the entire simulation for the different observing noises. Relaxing the targets perfect observation assumption, one can conclude that the performance is not compromised until an unitary standard deviation $\xi = 1$. For greater deviation, i.e. for poorer target-tracking sensors, the performance starts being compromised. However, the qualitative results are maintained and the performance decreases when the noise increase as expected.

Considering the exactly the same scenario as before, we also attempt to relax the Assumption 1 by consider imperfect actuators. The steering actuator is now modelled by

$$\mathbf{u}_k^i = \mathbf{u}_k^i + \boldsymbol{\sigma}_k^i,$$

where $\boldsymbol{\sigma}_k^i = [\sigma_{k,E}^i, \sigma_{k,N}^i, \sigma_{k,h}^i]^\top$ and $\sigma_{k,E}^i, \sigma_{k,N}^i, \sigma_{k,h}^i$ are zero mean Gaussian noises with standard deviation μ.

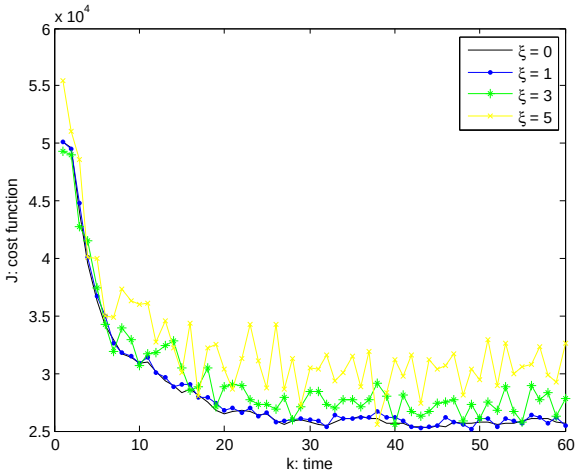

Figure 9.22: Slow targets with observing noise performance.

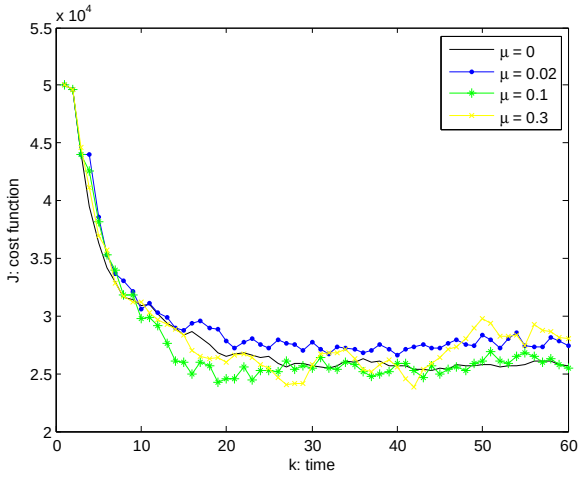

Figure 9.23: Slow targets with actuating noise performance.

One can conclude that our algorithm is more sensitive to actuating noise than to sensing noise (Figure 9.23). Feasibility is not always accomplished the simulations due to the vehicle dynamics constraints. Also, even with a small standard deviation error of $\mu = 0.02$, the decrease of performance is not neglectable. For larger noises $\mu = 0.1, 0.3$, the performance deviates from the noiseless solution, and the dependence with the level of noise is not as clear as for the observations noise. Therefore, we conclude that the proposed algorithm needs precise actuators to steer the UAVs accurately.

Emergency manoeuvres

Regarding the emergency manoeuvres, we consider the two-layer asynchronous-distributed algorithm for slow targets scenario ($W_{\max} = 1$). We simulate the agents' collision avoidance algorithm using fixed-wings (Figure 9.24) and the agents' refuelling algorithm using quadrotors (Figure 9.25), the list of parameters can be seen in Table 9.4.

Parameter	Value	Description
N	3	finite horizon
n, m	2 & 3, 100	number of fixed-wings & quadrotors, targets
ρ, ϕ	0.01, 0.01	performance weighting factors
$\bar{h}, V_M, u_{\max}$	10, 2, 0.8	fixed-wings' specifications
$h_{\min}, v_{\max}, f_{\max}$	10, 2, 2	quadrotors' specifications
$W_{\max}$	1	targets' maximum velocity
R_{CA}	10	collision avoidance range
R	120	agents' sensing range
Δt	1	time-step (in seconds)
k	100	number of time-steps of the simulation

Table 9.4: Emergency manoeuvres simulations parameters.

To test the agents' collision avoidance algorithm, we force a danger of collision situation between two fixed-wings. 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. After the desired separation is replaced, the agent 1 returns to its fixed-altitude $\bar{h}$.

The agents' refuelling is simulated and its performance shown in Figure 9.25. The performance

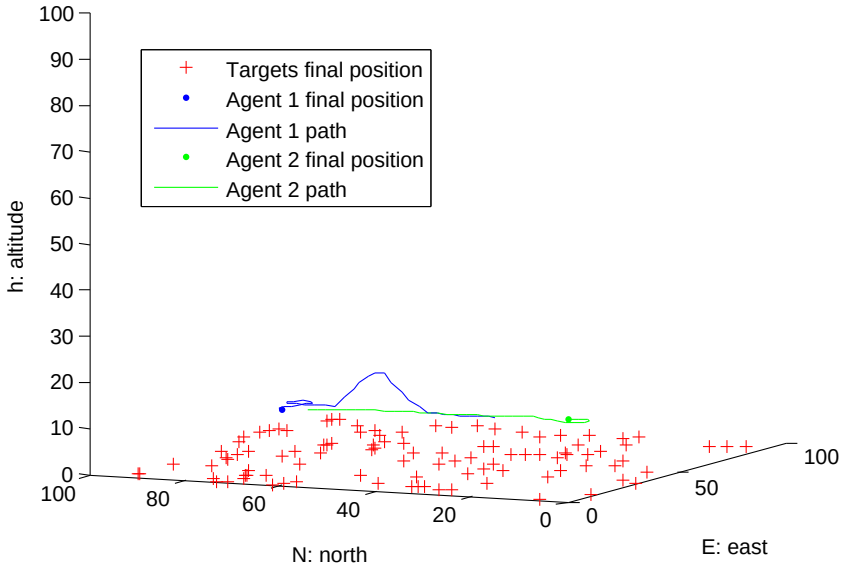

Figure 9.24: Slow targets collision avoidance path: the two agents change their tracking region and Agent 1 changes its flying altitude to avoid colliding with Agent 2.

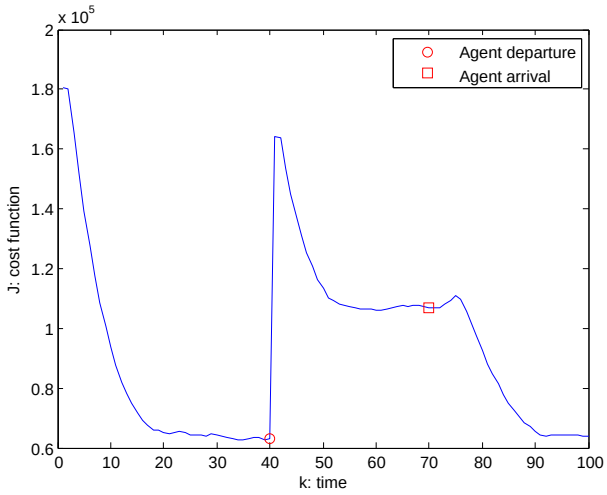

Figure 9.25: Slow targets refuelling performance: one agent departs to refuel at $k = 40$ (red circle) and it arrives back at $k = 70$ (red square).

decreases largely when one of the three agents departs to refuel ($k = 40$). Then, the remaining two agents reorganise themselves, lowering the cost-function J. When the agent returns from refuelling ($k = 70$), another transitory phase brings the performance criterion back at $k \approx 90 - 100$ to the initial performance achieved before the refuelling at $k \approx 20 - 40$.

Chapter 10

Conclusions and Future Work

Summary of the results

In this work we address the topic of distributed optimisation for collaborative multi-UAV multi-target tracking. A brief survey on the topic is presented, followed by an analysis of the most relevant state-of-the-art literature.

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. This formulation is first solved using a naive centralised approach that clearly does not meet our objectives, however it proves our desired tracking behaviour using the problem formulation.

The goal of convexifying the problem and distributing it among the agents, motivated our novel two-layer approach. We relax and distribute our original centralised problem into two layers helping us accomplish three important achievements.

First, our approach is based only in a local asynchronous and unreliable communication protocol between the neighbour agents. Being fully distributed and based in a realistic communication scenario, our proposed algorithm does not need any central node and it is highly scalable for large teams.

Second, the tracking problem is convexifyed which naturally brings nice converging guarantees and allows a faster implementation. Persistent feasibility is ensured using a worst-case scenario prediction. In the SDP tracking problem for quadrotors and in the Sequential Convex Programming algorithm which solves SDPs iteratively for fixed-wings, real-time of our target-tracking problem is achieved with a number of targets over number of agents ratio up to $m/n = 10$.

Splitting the global tracking problem into a global assignment problem and local tracking problems is the third important achievement worth to mention. This approach allows us to completely separate the partition algorithm from the tracking task. Therefore, the problems can be improved/tested separately, allowing the assignment problem to be run at a slower rate for instance.

Finally, a theoretical comparison between our two-layer proposed algorithm and our initial problem formulation is made. The solutions of both approaches are studied and proved to be highly related. For the static targets scenario, the optimal *steady-state solution* is proved to be identical, motivating the use of our algorithms. A comprehensive varied set of simulations is performed and their results discussed. Also, two emergency manoeuvres are added to our algorithm, in order to deal with collision avoidance between the agents and with agents' refuelling.

Recommendations

This work opens and suggests some challenges for future research. Here we pinpoint the research fields that we believe to be more interesting.

First of all, an implementation of our pseudo-code on real UAVs, with all the practical issues that will arise, is a must.

Another research field arises considering scenarios with obstacles. For instance, a set of known obstacles such as houses can be inserted in our scenario space. These obstacles constrain the line-of-sight from the agents towards the targets, and this should be taken into account in the optimisation problem. Also, these obstacles will change the so-called *worst-case scenario* in the targets' movement, changing the full target coverage constraint formulation. Instead of moving away from the agents, the worst case for a target could be hiding itself by moving closer to a certain obstacle.

Considering the non-holonomic fixed-wings dynamics, the convexification 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, higher weights can be given to the targets "in-front" of the agent so the Voronoi centroids would tend to be placed considering the vehicle's heading. We believe that this idea might be able to adjust the partition, improving the overall performance by taking into account the agents' dynamics.

Heterogeneous teams of agents are considered but not simulated. Our algorithms are able to deal with combining quadrotor and fixed-wing agents, with different ranges and velocities. Simulations using different agents combinations with different parameters might bring interesting results. The performance obtained through simulations using different combinations of agents can lead to an interesting analysis to determine which UAVs are able to track better in a given scenario.

Finally, in most of the real applications, the targets movement is not random. The targets are themselves vehicles with a certain dynamics. In a urban environment, the targets' movement can also be restricted to streets, or more likely to on certain routes. This change will require the presence of a target model to help the agents predicting where the target is going. Also, when one target travels from one tracking region to another, information about its model might be needed to be communicated between agents, in the case of online model estimation.

Bibliography

[1] T.M. Adami and J.J. Zhu. 6DOF flight control of fixed-wing aircraft by trajectory linearization. In *American Control Conference (ACC)*, pages 1610–1617, 2011.

[2] A. Bakhtari and B. Benhabib. An active vision system for multitarget surveillance in dynamic environments. *Trans. Sys. Man Cyber. Part B*, 37:190–198, 2007.

[3] A. Balluchi, A. Bicchi, B. Piccoli, and P. Soueres. Stability and robustness of optimal synthesis for route tracking by Dubins' vehicles. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1, pages 581–586, 2000.

[4] R.W. Beard, T.W. McLain, D.B. Nelson, D. Kingston, and D. Johanson. Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE*, 94(7):1306–1324, 2006.

[5] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multipliers Methods*. Athena Scientific, 1996.

[6] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1st edition, 1996.

[7] F. Borrelli, T. Keviczky, K. Fregene, and G. J. Balas. Decentralized receding horizon control of cooperative vehicle formations. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, pages 3955–3960, 2005.

[8] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for linear and hybrid systems*. 2013.

[9] Stephen Boyd. Sequential convex programming. Notes for EE364B, 2008.

[10] Fredrik Bruzelius. *Linear Parameter-Varying Systems: an approach to gain scheduling*. PhD thesis, Department of Signals and Systems, Chalmers University of Technology, 2004.

[11] F. Bullo, R. Carli, and P. Frasca. Gossip coverage control for robotic networks: Dynamical systems on the space of partitions. 50:419–447, 2012.

[12] D. W. Casbeer, S. Li, R. W. Beard, R. K. Mehra, and T. W. McLain. Forest fire monitoring with multiple small UAVs. *Proceedings of the American Control Conference*, pages 3530–3535, 2005.

[13] T.H. Chung, V. Gupta, J.W. Burdick, and R.M. Murray. On a decentralized active sensing strategy using mobile sensor platforms in a network. In *43rd IEEE Conference on Decision and Control.*, volume 2, pages 1914–1919, 2004.

[14] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust-region methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[15] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications*. Springer, 2000.

[16] J. De Geeter, J. De Schutter, H. Bruyninckx, H. Van Brussel, and M. Decrton. Tolerance-weighted L-optimal experiment design: a new approach to task-directed sensing. *Advanced Robotics*, 13:401–416, 1998.

[17] J. Derenick, J. Spletzer, and A. Hsieh. An optimal approach to collaborative target tracking with performance guarantees. *Journal of Intelligence Robotic Systems*, 56:47–67, 2009.

[18] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Rev.*, 41:637–676, 1999.

[19] Qiang Du, Maria Emelianenko, and Lili Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM J. Numer. Anal.*, 44:102–119, 2006.

[20] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79, 1957.

[21] Pedro Fazenda and Pedro Lima. Non-holonomic robot formations with obstacle compliant geometry. *6th IFAC Symposium on Intelligent Autonomous Vehicles*, 6:439–444, 2007.

[22] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-theoretic coordinated control of multiple sensor platforms. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1521–1526, 2003.

[23] W.W Hargrove, R.H Gardner, M.G Turner, W.H Romme, and D.G Despain. Simulating fire patterns in heterogeneous landscapes. *Ecological Modelling*, 135:243–263, 2000.

[24] O. Hlinka, F. Hlawatsch, and P.M. Djuric. Distributed particle filtering in agent networks: A survey, classification, and comparison. *IEEE Signal Processing Magazine*, 30:61–81, 2013.

[25] A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems*, pages 299–308, 2002.

[26] T. Keviczky, F. Borrelli, and G.J. Balas. Stability analysis of decentralized rhc for decoupled systems. In *44th IEEE Conference on Decision and Control and European Control Conference*, pages 1689–1694, 2005.

[27] T. Keviczky, F. Borrelli, K. Fregene, D. Godbole, and G. J. Balas. Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Transactions on Control Systems Technology*, 16:19–33, 2008.

[28] S.M. LaValle, D. Lin, L.J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 737–742, 1997.

[29] Steven M. LaValle, Hctor H. Gonzlez-Baos, Craig Becker, and Jean claude Latombe. Motion strategies for maintaining visibility of a moving target. In *Proceedings of the IEEE International Conference on Robotics & Automation (ICRA*, pages 731–736, 1997.

[30] S. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theor.*, 28:129–137, 2006.

[31] J. Lofberg. YALMIP: a toolbox for modeling and optimization in MATLAB. In *IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, 2004.

[32] James Manyika and Hugh Durrant-Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Prentice Hall PTR, 1995.

[33] S. Martínez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42: 661–668, 2006.

[34] G. Mathews and H. Durrant-Whyte. Decentralized optimal control for reconnaissance. *Proceedings of the Conference on Information, Decision and Control*, pages 314–319, 2007.

[35] T. Matsuyama and N. Ukita. Real-time multitarget tracking by a cooperative distributed vision system. *Proceedings of the IEEE*, 90:1136–1150, 2002.

[36] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. De Schutter. Active sensing for robotics - a survey. In *Proceedings of the 5th International Conference On Numerical Methods and Applications*, pages 316–324, 2002.

[37] Richard M. Murray. Optimization-based control, 2010.

[38] Katta G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16: 682–687, 1968.

[39] Arkadi Nemirovski. Notes on interior point polynomial time methods in convex programming. Notes for ISYE 8813, 2004.

[40] R. Olfati-Saber. Distributed kalman filter with embedded consensus filters. In *44th IEEE Conference on Decision and Control and European Control Conference*, pages 8179–8184, 2005.

[41] R. Olfati-Saber. Distributed tracking for mobile sensor networks with information-driven mobility. *Proceedings of the American Control Conference*, pages 4606–4612, 2007.

[42] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95:215–233, 2007.

[43] L. Ong, T. Bailey, H. Durrant-Whyte, and B. Upcroft. Decentralised particle filfilter for multiple target tracking in wireless sensor networks. *11th International Conference on Information Fusion*, pages 1–8, 2008.

[44] Lynne E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5: 5–19, 1999.

[45] Lynne E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12:231–255, 2002.

[46] Lynne E. Parker and Brad A. Emmons. Cooperative multi-robot observation of multiple moving targets. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 2082–2089, 1997.

[47] Xiaohong Sheng, Yu-Hen Hu, and P. Ramanathan. Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In *4th International Symposium on Information Processing in Sensor Networks*, pages 181–188, 2005.

[48] A. Simonetto and T. Keviczky. Distributed multi-target tracking via mobile robotic networks: a localized non-iterative SDP approach. *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 4226–4231, 2011.

[49] Andrea Simonetto, P. Scerri, and K. Sycara. A mobile network for mobile sensors. In *11th International Conference on Information Fusion*, pages 1–8, 2008.

[50] J. R. Spletzer and C. J. Taylor. Dynamic sensor planning and control for optimally tracking targets. *International Journal of Robotic Research*, 22:7–20, 2003.

[51] Zhijun Tang and U. Ozguner. Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21:898–908, 2005.

[52] D. Uciński. Optimal sensor location for parameter estimation of distributed processes. *International Journal of Control*, 73:1235–1248, 2000.

[53] Bo Yang and Mikael Johansson. Distributed optimization and games: A tutorial overview. In Alberto Bemporad, Maurice Heemels, and Mikael Johansson, editors, *Networked Control Systems*, Lecture Notes in Control and Information Sciences, pages 109–148. Springer London, 2010.

[54] Pengcheng Zhan, D.W. Casbeer, and A.L. Swindlehurst. Adaptive mobile sensor positioning for multi-static target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 46:120–132, 2010.

[55] S. Zhao, B. M. Chen, and T. H. Lee. Optimal sensor placement for target localization and tracking in 2D and 3D. *arXiv*, 2012.

[56] K. Zhou and S. I. Roumeliotis. Optimal motion strategies for range-only distributed target tracking. *Proceedings of the American Control Conference*, pages 5195–5200, 2006.

[57] K. Zhou and S. I. Roumeliotis. Optimal motion strategies for range-only distributed target tracking. Technical report, University of Minnesota, 2006.

[58] K. Zhou and S. I. Roumeliotis. Optimal motion strategies for range-only constrained multisensor target tracking. *IEEE Transactions on Robotics*, 24:1168–1185, 2008.

[59] K. Zhou and S.I. Roumeliotis. Multirobot active target tracking with combinations of relative observations. *IEEE Transactions on Robotics*, 27:678–695, 2011.