TÉCNICO LISBOA

Decentralized Rendezvous and Formation Algorithms for Multi-agent Systems in Disconnected Network Topologies

Rafael Meneses Lucas Ribeiro

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Daniel de Matos Silvestre
Prof. Carlos Jorge Ferreira Silvestre

Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia
Supervisor: Prof. Daniel de Matos Silvestre
Member of the Committee: Prof. João Manuel de Freitas Xavier

January 2021

Acknowledgments

Firstly, I express my gratitude towards my advisors, Professor Daniel Silvestre and Professor Carlos Silvestre, for accepting me as their apprentice and for their support and guidance during my research. I am profoundly grateful for their contributions in pushing me to improve my work methodology, analytical and creative thinking, and, most of all, for propelling me to write four research papers, at least two of which have already been accepted/published.

Secondly, I express my appreciation to the remaining member of the committee that evaluated my midterm progress, Professor João Xavier, for his comments and constructive feedback that helped me better understand the topic I was approaching.

To all my friends that I met in college: thank you for accepting the person I was five years ago, and thank you for making me the person I am today. I am truly thankful for what has become the best chapter of my life, with all the 48-hour coding ”nights”, all the vacations and game nights, all the pizza and all the sushi, and all the drinking nights simultaneously discussing exactly nothing and unquestionably everything that fueled my creative thinking and made me realize that if I can evolve so much in just five years with you, then you will make me reach impossible heights in the years to come.

Finally, my greatest appreciation goes to my parents for teaching me to think creatively and independently from a very young age, for providing me the environment, the opportunity, and the motivation for doing what I love, for always showing support when I failed, and for showing me how I could be independent by allowing me the freedom to cause, fix, and improve from my mistakes, from a young age.

Financial Support: This work was partially supported by the projects MYRG2016-00097-FST and MYRG2018-00198-FST of the University of Macau; by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) thorugh Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020.

Abstract

This dissertation addresses the problem of having a multi-agent system converge to multiple dynamic rendezvous areas for generally disconnected network topologies. There is no assumption on the connectedness of the network topology, which is unknown to all agents. Two algorithms are designed: a partially-decentralized algorithm for agents with no localization and limited communication capabilities, and a fully-decentralized algorithm for non-communicating agents with localization and measuring sensors whose additional objective is to create and maintain formations. For the first algorithm, the localization is performed by measurement/communication towers that determine the noisy positions and velocities of the relevant agents and transmit them in directional broadcasts. The proposed solutions consist of improved flocking-based movement algorithms tailored to the proposed scenario coupled with a utility-defined mission plane and mechanisms to prevent agent-agent and agent-obstacle collisions. The performance of the algorithms is presented through simulations for a multitude of environments. These empirical results show the agents rendezvous to the multiple dynamic rendezvous areas in the presence of undesirable areas, static environmental obstacles, and arbitrary elimination of desirable areas, with varying degrees of efficiency.

Keywords

Decentralized Control; Distributed Control; Cooperative Control; Multi-Agent Systems; Flocking Algorithms.

Resumo

A presente dissertação aborda o problema de convergir um sistema multiagente para múltiplas áreas de encontro dinâmicas para redes com topologia geralmente desconexa. O sistema não tem pressupostos sobre a conectividade da topologia de rede, desconhecida para todos os agentes. Dois algoritmos são propostos: um algoritmo parcialmente-descentralizado para agentes sem capacidades de localização e recursos de comunicação limitados e um algoritmo descentralizado para agentes não-comunicantes com sensores de localização e medição com o objetivo adicional de criar e manter formações. No primeiro algoritmo, a localização é realizada por torres de medição/comunicação que determinam as posições e velocidades, corrompidas por ruído, dos agentes relevantes, e transmitem-nas através de broadcasts direcionais. As implementações propostas consistem em algoritmos de movimento baseados em flocking adaptado aos cenários propostos, associados a um plano de missão definido por uma função de utilidade e mecanismos para evitar colisões agente-agente e agente-obstáculo. O desempenho dos algoritmos é apresentado através de simulações para diversos ambientes. Estes resultados empíricos demonstram que os agentes convergem para as múltiplas áreas dinâmicas desejadas na presença de áreas indesejáveis, obstáculos ambientais estaticamente posicionados e eliminação arbitrária de áreas desejáveis, com diferentes níveis de eficiência de convergência.

Palavras Chave

Controlo Descentralizado; Controlo Distribuído; Controlo Cooperativo; Sistemas Multiagente; Algoritmos de Flocking.

Contents

List of Figures

1

Introduction

Contents

1.1 Motivation

The rendezvous problem, defined as a group of agents attempting to converge to the same area in the mission plane, is a central issue when establishing a swarm-like network of agents to perform a given assignment. This thesis addresses the rendezvous problem for multiple dynamic rendezvous areas by proposing two algorithms that account for sensor noise, asynchronous directional communication, and an unknown disconnected network topology. Thus, it entails that the initial graph is composed of multiple connected components, precluding the traditional approach of using consensus algorithms. The rendezvous areas are classified as dynamic because both their values for the mission objective and positions are time-varying and mapped through a utility function. Moreover, such a flexible formulation is directly applicable to exploration missions where the value associated with the interest of a given location depends on past visits by the agents. Two algorithms are proposed to solve the rendezvous problem with specific characteristics in terms of the capabilities of the nodes.

The main design objectives for this problem are:

- no single point-of-failure;
- no constraints posed by the algorithms on the number of agents in the network;
- possibility to have nodes join and leave the network;
- computational and power efficiency in order to increase the battery autonomy of the mobile agents.

The first algorithm (*Algorithm 1*), presented in Chapter 2, is designed for agents with no localization sensors and limited communication to receive directional broadcasts of their position and velocity. The localization task is left for the fixed agents (towers), which is ideal for operations in GPS-denied environments, such as open-space indoor navigation or missions including a large number of vehicles since it allows to have inexpensive nodes and only a few resourceful agents with low mobility. Regardless of whether the implementation of the towers is a stationary structure located in the perimeter of the mission plane or more complex mobile agents, we will refer to them as *communication towers*. The strategy of offloading the measurements and estimation of position and velocity to the towers enables agents with smaller costs and higher power efficiencies due to the reduced payload, which results in greater battery autonomy. Thus, this algorithm is labeled as partially-decentralized in the sense that the protocol running in the agents is decentralized and makes decisions based on the received positions, but the communication towers act as a centralized unit.

The second algorithm (*Algorithm 2*), presented in Chapter 3, is designed to work in the scenario where agents have sensors to measure their location and velocity and the position of their nearby neighbors. Moreover, agents can sample the utility function at their position and that of their neighbors, removing the requirement to have a defined function for the entire mission plane. In practical examples,

this utility function can measure the concentration of some chemical or radiation, proximity to a resource, among others. In this scenario, we avoid the need to have communication towers, therefore improving the resilience of the system. *Algorithm 2* is fully-decentralized since it is composed of agents that compute their movement based on measured data without requiring inter-agent communication. In case the utility function can only be measured at the current position, the algorithm is distributed as the only required communication to fully explore the proposed control law is the utility values of the neighbors.

The design objectives are met differently by the two algorithms. In the partially-decentralized algorithm, achieving no single point-of-failure translates into the need for multiple communication towers. On the other hand, the fully decentralized algorithm achieves this goal unless all nodes fail. The second objective is pursued through the use of directional antennae in the communication towers of *Algorithm 1* and local area broadcasts from the agents of *Algorithm 2*. In doing so, the shared medium is used with efficiency with fewer limitations in terms of latency to communicate all positions or to transmit utility values. The choice of control laws based on flocking algorithms goes towards fulfilling the remaining objectives since nodes require low computational power and act independently based on the gathered local information.

In real missions, an important aspect is the need to avoid collisions, especially taking into consideration that measurements are performed with noise. Often, procedures that promote collision-free formations are employed based on the idea of potential virtual fields around the agents that repel nearby nodes. However, if one can consider worst-case scenarios for the noisy measurements, the algorithms can possess the theoretically prove property of avoiding collisions. This topic will be the subject of consideration in the discussion of both scenarios in Chapter 2 and Chapter 3.

The aforementioned discussion could hint at the reader that the described scenario is a trivial application of distributed consensus algorithms. However, these typically require a connected initial network topology as nodes move within the convex hull of the previous positions. As a consequence, a consensus-based approach would have all nodes within the same cluster to converge to a weighted average of their initial states. In this thesis, the envisioned solutions relax this assumption and are capable of leading the whole formation to the desirable rendezvous area. In the following subsections, we will dwell on the state-of-the-art methods related to the rendezvous problem and highlight the contributions of this thesis.

1.2 Previous Work Review

In this section, we aim at providing a general overview of how this thesis fits within the body of literature related to rendezvous problems and formations in multi-agent systems. A more in-depth survey is presented in each specific chapter whenever describing the proposed solution for each of the considered

scenarios.

The problem of rendezvousing to multiple areas is addressed in [3], which proposes a leader/follower approach. Agents are divided into groups (one group per rendezvous point), which subsequently create a hierarchical tree based on the agents positions. Each group leader has the knowledge of a rendezvous point and drives to that location while the followers implement a consensus algorithm in order to converge to the leader position. The implementation of [3] results in each group achieving rendezvous in different areas.

In order to have all nodes converge to a single unknown position, [4] assumes a scalar field defined by a utility function with a single maximum to correspond to the desired point. Each agent is equipped with sensors to sample the field and transmit its value to the nearby nodes. Upon receiving the neighbors values, each agent calculates a movement direction that depends on the relative difference to the evaluation of the utility function at its location. As a result, nodes are attracted to other agents in positions with higher values. This procedure can be thought of as a consensus method with a leader (the node with the highest utility value).

To mitigate the existence of multiple isolated clusters in an initially disconnected network topology, the proposed solution uses flocking-based movement, proposed in [5]. This movement is based on three rules, which create an emergent behavior similar to the movement observed in a flock of birds: Collision Avoidance, Velocity Matching, and Flock Centering. These rules are focused on maintaining the agents at a minimum distance, maximum distance, and moving in the same direction, respectively. When an agent encounters a new cluster, it dynamically merges with it, which can result in a decreasing number of isolated agents. Additionally, [5] has no assumptions on the communication capabilities of the agents.

In order to provide a collision-free environment, the agents must have a method to detect collisions. We present three options, dispersed in the spectrums of implementation simplicity and exactness: Bounding-Spheres, Ray-Tracing [6], and Bounding-Cylinders [7]. The first is the simplest to implement, which relies on bounding the agents' positions with virtual spheres, and detecting collisions based on the spheres: if the spheres are closer than the sum of their radius, there is a collision. Similarly, the Bounding-Cylinders method detects a collision if the bounding cylinders are intersecting. Ray-tracing is the most computational demanding of the three methods, which is a required compromise for its exactness. In two-dimensions, this method is exact at detecting collisions, based on tracing the future positions of a convex bounding polygon's vertices, verifying if these positions intersect other agents' bounding shapes.

In the literature, the solutions for the problem of formation construction and maintenance can be divided into three general categories: physical-leader formations, virtual-leader formations, and distance-based formations.

In the first category - Physical-Leader Formations - [8] addresses the problem of tracking a target's

trajectory while maintaining the desired formation. The target is considered the physical leader, with all agents tracking its position. The formation points' positions are defined relative to the position of the target, and are predetermined and pre-assigned: the structure does no adapt to new agents entering the formation or existing agents leaving. [1] uses a hierarchical leadership structure to define the formation structure. Each agent follows its immediate leader (according to the defined hierarchy) by maintaining a predetermined position relative to its leader agent. The slots are predetermined (resulting in not allowing new formation members), but the assignment is executed in real-time: the agents reach a consensus on the attribution of each slot, based on the agents' physical states. Due to the consensus protocols required by this implementation, it has limited scalability: it would be computationally infeasible reaching consensus whenever an agent joined/left a large network.

In the second category - Virtual-Leader Formations - [2] uses a single virtual leader for a group of mobile agents to move in a desired formation. This method requires the virtual leader's path to be known apriori, which is inexecutable when the agents' objective is to rendezvous in desired areas with dynamic, arbitrary positions. The formation structure is predetermined: no agents can join, and if an agent leaves, the structure does not adapt.

In the third category - Distance-based Formations - the formation structure is defined by a distance matrix: the entries correspond to the desired distance between the agents. This matrix is predetermined, which results in the structure being closed to new agents, and not adapting if an agent leaves. In [9], the agents move in the direction that minimizes the error between the entries in the distance matrix and the actual distances to the agents. In [10], it is assumed the centroid of the formation (defined by a distance matrix) has an arbitrary desired velocity, only available to the agents' leader, which results in a leader/follower protocol when the formation is stabilized.

Generally, the literature examples consider predetermined formation structures with pre-assigned slots, which results in no new agents being allowed to participate in the formation, and the structure not adapting to agents leaving. In the case of formations with an agent leader (virtual or physical), if the leader fails, the system cannot proceed.

1.3 Contributions

The solutions proposed in this thesis present a novel method to achieve rendezvous to multiple dynamic desirable areas without having assumptions on the connectedness of the network topology while providing guarantees for an environment without agent-agent collisions in the presence of noisy measurements. Furthermore, the implementation of Chapter 3 provides a fully-decentralized solution with the added objective of creating and maintaining non-rigid dynamic formations. The main contributions of both proposed solutions are:

- There is no apriori information regarding the number, positions, and movement dynamics of the rendezvous areas. Additionally, this number is not fixed: new rendezvous areas are created arbitrarily and areas are removed from the mission plane by being explored by the agents.

- The network topology does not need to be initially connected. Sections 2.7.3 and 3.7.3 provide simulation examples where the initial topology is disconnected and the agents reach a single cluster.

- The position and velocity measurements are assumed to be corrupted by noise.

- The collision avoidance method implemented provides guarantees of an environment without agent-agent collisions with each agent predicting the worst-case set of all possible positions for its neighboring agents considering their past known positions.

- High scalability due to the agents only having partial and localized knowledge of the network and not requiring consensus or leader selection protocols.

- No single point-of-failure. The system will continue to function, with generally decreased efficiency, with at least one agent and at least one communication towers (in *Algorithm 1*), and with only at least one agent (in *Algorithm 2*).

Algorithm 1 provides the following additional specific contributions:

- We provide theoretical results that the overall dynamical system composed of the various agents behaves as a consensus algorithm in the velocity and as a noisy gradient ascent in the position, thus converging to a local neighborhood of the global maximum when the utility function is concave;

- Agent power efficiency. By offloading the localization sensors to the communication towers and only using the communication equipment to receive broadcasts, the agents conserve more power, which can result in increased battery autonomy, or an inferior cost if the agents are equipped with smaller batteries - which additionally reduces weight.

- Agent cost efficiency. By offloading the localization sensors and only requiring communication receivers, the agents become more inexpensive. If the agents are built with smaller batteries (option provided by the increased power efficiency), the cost reduction increases.

Algorithm 2 provides the following specific contributions, additional to the common contributions:

- We provide theoretical results stating that the algorithm leads all clusters of agents to asymptotically either form a m-sided polygon formation or converge to a local maximum;

- No requirement for a standardized coordinate system across all agents. In *Algorithm 1*, the positioning data is broadcasted to multiple agents simultaneously, which requires all agents to interpret the positions identically: the agents' frames of reference must be identical. In *Algorithm 2*, the measurements are taken by the agents according to their individual coordinate system, so a common coordinate system is not required.
- No computational limitations to the number of agents or associated decrease in algorithmic efficiency (after a finite upper bound on the number of agents is reached). Due to the proposed implementation being based on the agents' local space, after an agent's local area is sufficiently full of agents, to the point no other agent can be in its area, the agent's computationally efficiency reaches its lower bound, not decreasing even if more agents enter the network.

1.4 Document Structure

The remainder of this document is organized as follows. Chapter 2 presents *Algorithm 1* tailored to the partially-decentralized setup and introduces the strategy for collision avoidance. This was a stepping stone towards the fully-decentralized algorithm presented in Chapter 3. General conclusions are offered in Chapter 4.

1.5 Notation

This section provides an extensive list of the notation used in this dissertation. Further details will be presented as necessary.

1.5.1 Mathematical Notation

$\|x\| =$ ℓ_2-norm of vector x

$\mathbb{N} =$ natural numbers excluding zero

$\mathbb{N}_0 =$ natural numbers including zero

$\mathbb{R}^3 =$ three-dimensional space of real numbers

$\mathbb{R}^n =$ n-dimensional space of real numbers

$\mathbb{R}^+ =$ set of positive real numbers

$\mathbb{R}^- =$ set of negative real numbers

$\mathcal{A} \setminus \mathcal{B} =$ set difference: set of elements in $\mathcal{A}$ not in $\mathcal{B}$

1.5.2 Agents' Properties Notation

$q_i =$ position of agent i

$v_i =$ velocity of agent i

$u_i =$ control vector of agent i

$\mathcal{X}_i =$ position-estimate polytope of agent i

$\mathcal{N}_i =$ neighbor set of agent i

$\mathcal{N}_{i,d} =$ set of agent i's neighbors within d distance units of i

1.5.3 Agent Movement Notation

$u_i(k) =$ desired movement vector (for agent i at time step k)

$u_i^s(k) =$ separation component

$u_i^c(k) =$ cohesion component

$u_i^a(k) =$ alignment component

$u_i^{attr}(k) =$ attraction component

$u_i^u(k) =$ utility component

$u_i^f(k) =$ formation component

$u_i^r(k) =$ randomness component

$D_{max} =$ maximum allowed distance for an agent per update step

$D =$ maximum allowed distance for an agent determined by the collision avoidance mechanism

1.5.4 Communication Notation

$\alpha =$ communication towers' strip angle

$\mathcal{T}_i =$ communication tower with index i

$\mathcal{N}(k) =$ neighbor set for a broadcast strip at the discrete time step k

$\mathcal{N}(k, \mathcal{T}_1) =$ neighbor set for the broadcast strip of tower $\mathcal{T}_1$ at the discrete time step k

$SR =$ agent sensing radius

2

Rendezvous with Communication Towers

Contents

2.1 Introduction

This Chapter presents a partially-decentralized solution to the rendezvous problem considering agents with limited communication capabilities and no localization sensors. In order to have a resilient algorithm to node failure in the network, agents are assumed to be identical and execute the same algorithm, thus voiding the need for leader selection protocols or specialized messages. Moreover, by pursuing a strategy based on local information to the agent, the algorithm can cope with several real-environment problems, such as asynchronous and delayed communication, packet losses, and agents arbitrarily entering and exiting the network.

In the envisioned scenario, the lack of localization sensors requires the existence of monitoring towers that use directional communication to transmit noisy position and velocity data to the agents. As a consequence, nodes are listening to the shared medium and will also receive information of nearby nodes. Upon receiving a broadcast from the tower, the movement of the agent is triggered asynchronously.

In the literature, the usual approach is based on consensus methods that generally require the agents to communicate with each other. In comparison, the solution described in this Chapter aims at attaining the following objectives:

- networks of large size, which is computationally infeasible with broadcast consensus;
- power efficiency of the standard agents (better autonomy), by offloading most of the sensors to the communicating towers;
- cost efficiency for the standard agents that have little payload in terms of sensors.

These objectives result in the system being composed of inexpensive mobile agents and more expensive communication towers. In practice, these can be implemented as mobile agents that become static to save power and carry all the necessary equipment to provide the standard agents with localization.

The contributions presented in this Chapter have been published in [11] and are in review in [12].

2.2 Previous Work Review

Various components of our proposed solution are related to topics in the literature: consensus with a switching topology network, rendezvous algorithms, flocking movement, collision avoidance mechanisms, glow-worm optimization, gradient source seeking, and polytope propagation. In the remainder of this section, we present related work from the various areas that find an intersection with our method.

2.2.1 Switching Topology

In [13], it is addressed the problem of reaching consensus for networks of agents with fixed and switching topologies. The network is defined as a graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k), \mathcal{A}(k))$, with $\mathcal{V}$ as the set of nodes, $\mathcal{E}(k)$ the set of edges, and $\mathcal{A}(k) = [a_{ij}(k)]$ the non-negative weighted adjacency matrix, where $a_{ij}(k) = 0$ if $(i, j) \notin \mathcal{E}(k)$. A switching topology can model, among other events, a link failure by setting the correspondent entry in $\mathcal{A}$ to zero. However, given the nature of a consensus algorithm, there is still an assumption of a connected topology over time.

The work in [13] defines a general function $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ to be implemented by the algorithm and taking the variables, $x_1, \ldots, x_n$, to some weighted combination of the initial state $x(0)$. A particular case of $\mathcal{F}-\text{consensus}$ corresponds to having $\mathcal{F}$ be the arithmetic average, which is known in the literature to be average consensus. The discrete-time protocol in [13] with a switching topology and zero communication time-delays is given by the following linear input:

$$u_i = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i) \tag{2.1}$$

where $\mathcal{N}_i$ is the set of neighbors of node i. Intuitively, this control law minimizes the differences between the node and neighbor states. Provided that all nodes follow (2.1), all agents asymptotically converge to the average for networks with a switching topology if the matrix $\mathcal{A}$ is appropriately selected [13]. The result naturally poses the assumption of a connected topology over time since each isolated cluster would converge to their specific value. In contrast, the proposed algorithm will remove this assumption and achieve convergence to a single point provided there exists enough concentration of nodes or sufficient time in the case of a single rendezvous point.

2.2.2 Rendezvous via Proximity Graphs

In [14], the objective is to achieve rendezvous for a dynamic network topology based on the proximity of the nodes. This setup is similar to the scenario considered in this thesis, where the agent position influences which messages it will receive, and as a consequence of the network topology. The edge set definition for a proximity graph is denoted by $\mathcal{E}_r$ to emphasize the dependence on the radius threshold for two nodes to be neighbors. In this case, the graph is denoted by $\mathcal{G}_r = (\mathcal{V}, \mathcal{E}_r)$. More precisely, $\mathcal{E}_r$ is defined as:

$$\mathcal{E}_r = \{(i, j) : \|p_i - p_j\| \leq r, i \in \mathcal{V}, j \in \mathcal{V}\}. \tag{2.2}$$

In [14], the *Circumcenter Algorithm* is composed of three tasks executed independently by each agent:

1. Detect neighbors within r distance;
2. Compute the circumcenter of the positions of the neighbors and the agent itself;
3. Steer towards the calculated circumcenter while maintaining connectivity with all its neighbors.

The circumcenter $\mathcal{S}$ is defined for a set of points in $\mathbb{R}^d$ as the center of the smallest d-sphere enclosing all points. In order for an agent to maintain connectivity with its neighbors, [14] proposes an allowed set of positions to which the agent can move. If agents i and j are neighbors, then their positions have to be contained in the closed ball $\overline{B}((p_i+p_j)/2, r/2)$. Generalizing for l neighbors, the set of allowed positions for agent i is constructed with:

$$\mathcal{C}_{p_i,r}(\{p_1,\dots,p_l\}) = \left(\bigcap_{q\in\mathcal{N}_i} \overline{B}\left(\frac{p_i+p_q}{2}, \frac{r}{2}\right)\right) \cap \overline{B}(p_i, s_{max}) \tag{2.3}$$

where s_{max} is the maximum distance each agent is allowed to move per update step, and $\overline{B}(p_i, s_{max})$ defines the closed ball that bounds the movement to s_{max}. To restrict the movement to the calculated set in (2.3), the agent solves the optimization problem:

$$\begin{aligned} \max_{\lambda} \quad & \lambda \\ \text{s.t.} \quad & \lambda \leq 1, \\ & (1-\lambda)q_0 + \lambda q_1 \in \mathcal{Q} \end{aligned} \tag{2.4}$$

The control input is determined with (2.4), which calculates the maximum distance agent i can move from its position, p_i (q_0 in Equation (2.4)), towards the circumcenter (q_1) while remaining contained in the allowed position set to maintain its connectivity ($\mathcal{Q}$).

Considering n mobile agents following the $CircumcenterAlgorithm$, with maximum update-step-length $s_{max} \in \mathbb{R}^+$, communication radius $r \in \mathbb{R}^+$, and a proximity graph with the same connected components as $\mathcal{G}_r$, [14] shows that:

1. If the positions of two agents belong to the same connected component of $\mathcal{G}_r$ for some $k \in \mathbb{N}_0$, then they remain in the same component for all $m \geq k$;
2. There exists $\mathcal{P}^* = (p_1^*, \dots, p_n^*) \in (\mathbb{R}^d)^n$ with the following properties:
   (a) $\mathcal{P}_m \to \mathcal{P}^*$ as $m \to \infty$;
   (b) $\forall i,j \in \{1,\dots,n\} : p_i^* = p_j^* \vee \|p_i^* - p_j^*\| > r$;
3. If $\mathcal{G} = \mathcal{G}_r$, then there exists $k \in \mathbb{N}$ such that $\mathcal{P}_m = \mathcal{P}^*$, for all $m \geq k$.

These results show that nodes remain in the same connected component, and when the graph is connected, the agents converge to a single position in finite-time. Rendezvous is guaranteed if and

only if the resulting communication graph contains a single connected component at least once every finite number of time instants. Whenever the graph is indefinitely partitioned, the algorithm will result in a different convergence position in each cluster. As our solution uses a movement behavior that enables isolated clusters of agents to form new connections, multiple components will be joined to a single cluster - depending on the initial state and node density - and the results of [14] will guarantee a finite-time rendezvous. However, it is noted that the work in [14] does not enforce collision avoidance, which will be a major difference to our proposal.

2.2.3 Multi-point Rendezvous

The problem of conveying mobile agents to a set of static points is addressed in [3] through the use of a leader/follower dynamic where the elected leaders are aware of the intended rendezvous points. Such an approach adds complexity to the solution due to the need of a leader selection procedure and also differs from our proposal in the sense that our rendezvous areas are assumed to be unknown. In [3], these m points are fixed and known a priori. The algorithm first divides the agents into m groups and then it selects a leader for each group, which is defined as the agent closer to all other agents within the group. A shortest-path tree is computed that serves as the hierarchy within that clique. At this point, the leader drives to the assigned rendezvous point maintaining connectivity, whereas the followers use a control law based on consensus dynamics. All nodes asymptotically reach the desired location [3].

One of the design objectives for the formation control presented in this thesis is to have all nodes use the same control law instead of using any hierarchical structure. In doing so, other nodes can join the network and behave accordingly without the need to rerun the leader selection. Moreover, rendezvous points in our scenario are defined to be points of maximum utility that are not known by the agents. Despite the rule to increase connectivity in the initial flocking algorithm proposed in the literature, we will present a collision avoidance mechanism with guarantees of connectivity maintenance regardless of noisy measurements and disturbances.

2.2.4 Gradient Source Seeking

In [4], it is proposed an algorithm for a group of mobile agents to locate an unknown maximum of a scalar field, based on flocking and Glowworm Swarm Optimization [15]. Comparable to our mission plane, the scalar field is defined with $\psi : \mathbb{R}^m \to \mathbb{R}$. The assumption of a unique maximizer and a known bound on the gradient is rather common as the algorithm is similar to a step of gradient ascent. The maximum point, q_s, is formally defined by:

$$\nabla(\psi(q_s)) = 0 \qquad \psi(q_s) \geq \psi(q) \; \forall q \in \mathbb{R}^m \tag{2.5}$$

Additionally, [4] assumes an agent at q_i has the capability to sense the value $\psi(q_i)$ and communicate with all agents j within a sensing radius r_s. This is similar to our proposal where towers broadcast the utility values $\psi(q_i)$ and $\psi(q_j),\ \forall j \in \mathcal{N}_i$. To simulate the attraction force to a higher-valued agent, [4] defines the movement component u_i as:

$$u_i = \rho \sum_{j \in \mathcal{N}_i} (\psi_j - \psi_i) n_{ij} \tag{2.6}$$

where $\rho \in \mathbb{R}^+$ is a tuning parameter and n_{ij} is a vector from node i to j. This methodology will also appear in our flocking rules to simulate attraction towards agents in more desirable positions of the mission plane.

2.2.5 Flocking

The results presented in Sections 2.2.1 and 2.2.2 assume a single connected component to achieve consensus over all the agents in the network. Another type of approach is the definition of flocking-based movement rules and allow the agents to explore the mission plane while maintaining connectivity to their current neighbors. The work in [5] proposes three components to emulate the behavior of biological flocks, such as a flock of birds or a school of fish. The action of each agent is selected in a decentralized manner by using local sensor data and without any message exchange with the neighbors. The three rules of movement aim to accomplish different objectives:

1. Collision Avoidance - each node tries to prevent getting too close to the others;
2. Velocity Matching - each node will try to match the velocity of its neighbors such that the flock can move as one entity;
3. Flock Centering - each node tries to remain closer than a maximum distance from nearby nodes to avoid forming independent clusters.

It is noted that in the majority of the literature, these rules are named Separation, Alignment, and Cohesion, respectively. In this thesis, we will be adopting this convention.

The collision avoidance method employed in typical flocking algorithms assumes static nodes by only considering their positions, disregarding their velocity, which can result in future collisions. It is a simplified version of dynamic collision avoidance, which would predict the positions of neighbor agents and verify their separation. The solution involves combining the three movement components using strict priority ordering. The component vectors are successively added until a maximum acceleration magnitude is reached. Intuitively, this technique aims at minimizing collisions by prioritizing the collision avoidance component.

In [5], the agents only have a localized perception of the flock, so the set of neighbors for an agent is constructed with all agents within a spherical zone with a fixed radius centered at the agent. This neighbor set definition is equivalent to *r-disks* proximity graphs presented in Section 2.2.2. Each neighbor affects the agent with a magnitude defined as an inverse exponential of the distance between them - to simulate a more significant influence from near neighbors.

2.2.5.A Flocking Implementation

In [16], three flocking algorithm implementations are presented: the first is a realization of original flocking (defined in [5]), the second adds a group objective component, and the third adds dynamic-obstacle avoidance. Similar to [5], the proximity graphs are undirected. In [16], the flock geometry is set to a lattice-type structure by enforcing all agents to be equally distant from their neighbors, i.e.,

$$\|q_j - q_i\| = \alpha \qquad \forall j \in \mathcal{N}_i \tag{2.7}$$

where $\alpha \in \mathbb{R}^+$, q_i is the position of agent i and $\mathcal{N}_i$ is its neighbor set. A flocking agent is referred to as an α-agent because it seeks to form an α-lattice as in (2.7). In order to accomplish this behavior, the authors propose the control law:

$$u_i = u_i^\alpha = \sum_{j \in \mathcal{N}_i} \phi(\|q_j - q_i\|_\sigma) n_{ij} + \sum_{j \in \mathcal{N}_i} a_{ij}(q)(p_j - p_i) \tag{2.8}$$

where n_{ij} is a vector along the line connecting q_i to q_j, p_i and p_j are the respective velocities of agents i and j, and $\phi(\|q_j - q_i\|_\sigma)$ is a pairwise potential function that represents how much agent i is attracted/repelled by agent j based on some σ-norm between the two positions.

2.2.5.B Flocking with Group Objective

In order to have nodes following a group objective, a modification to (2.8) is proposed in [16]:

$$u_i = u_i^\alpha + u_i^\gamma = u_i^\alpha + f_i^\gamma(q_i, p_i, q_r, p_r) \tag{2.9}$$

where $f_i^\gamma(q_i, p_i, q_r, p_r)$ is a linear combination of the relative differences of agent i's position and velocity to the desired rendezvous point modeled as a γ-agent with state $(q_r, p_r) \in \mathbb{R}^m \times \mathbb{R}^m$. This corresponds to the differences:

$$-c_1(q_i - q_r) - c_2(p_i - p_r) \qquad , \; c_1, c_2 \in \mathbb{R}^+ \tag{2.10}$$

These γ-agents are virtual nodes representing the rendezvous areas and are known to the α-agents.

In this thesis, we remove this assumption given that each node will not have prior knowledge of the rendezvous areas.

2.2.6 Set-Valued Observers

The problem of reaching consensus when position readings are corrupted by noise has been addressed in [17]. Using the concept of Set-Valued Observers (SVOs), each agent updates locally the received set-valued estimates for its position and that of its neighbors. The SVO framework produces a convex polytope containing all possible valid positions according to the received measurements. Then, the agents follow a standard control update rule:

$$X_i(k+1) = \alpha X_i(k) + (1-\alpha)\frac{1}{|N_i|}\sum_{j\in N_i} X_j(k) \tag{2.11}$$

where the true states are replaced by the set-valued estimates of these quantities X_i and X_j, respectively. In doing so, the consensus algorithm can cope not only with the noisy measurements but also with the fact that not all of them were taken simultaneously.

In [17], it is proven that in a network of agents following a linear consensus update rule, each connected component of the network converges to a point in the convex hull of the clique with its neighbors. Depending on the initial state, the node density in the mission plane, and the communication pattern, agents might rendezvous at a single point. [17] provides no guarantee of convergence for networks with disconnected topologies, which is a motivation to our work.

2.2.7 Collision Detection

The algorithms in [5] and [16] have *best effort* collision avoidance components that mitigate collisions between the mobile agents by adding a simulated repulsion factor to their behavior. The problem with these proposals is the weight adjustment needed to obtain the correct result. Moreover, this method does not produce a perfect mechanism to avoid collisions. This section describes methods for virtual collision detection used in the area of Computer Graphics that can be adapted to detect collisions between agents.

We present a standard basic method to detect collisions, Bounding Spheres, an improved version with a less conservative bounding volume, Bounding Cylinders, and an advanced method for polytope collision detection, Ray-Tracing. Other methods targeted for polytopes are available in the literature, such as calculating if there exists a separating plane between the polytopes [18], or Discrete Orientation Polytopes [19], which assumes only K possible rotational angles for the polytope and performs an interval overlap test. However, these do not provide the first collision point or the distance of overlap

considering the desired movement (direction and distance). The justification for the requirement of the first collision point will be presented in Section 2.5.4.C.

2.2.7.A Bounding Spheres

This method consists of bounding each agent with a virtual sphere, centered on the agent's position and with a radius such that it encloses the agent's volume or its position estimate in the form of an SVO. The detection between two spheres is calculated with:

$$\|c_i - c_j\| <= 2r \tag{2.12}$$

where c_i and c_j are the positions of the centers of the spheres bounding agents i and j, respectively. To calculate the distance agent i would have moved (while overlapping spheres) since the first intersection point, d is calculated with:

$$d = 2r - \|c_i - c_j\| \tag{2.13}$$

This method is the simplest to implement and the most computationally efficient; however, it leads to false positives due to its conservativeness - the spheres can be intersecting while the polytopes are separated.

2.2.7.B Ray-Tracing for Collision Detection

Using ray-tracing to detect collisions between convex bounding polygons provides exact results in two-dimensional space: if there is a collision, then the algorithm detects it without false positives. [6] uses ray-tracing to detect collisions between deformable bodies. Despite our bodies being rigid, the detection method is still applicable due to the purpose of the deformability mechanism being for reacting to collisions. To detect collisions, [6] traces a ray from each vertex of object i and verifies if it intersects with any point in object j. In cases where the bounding volumes are convex polygons, the first collision point between two volumes corresponds to the position of one of their vertices. With this result, it is guaranteed the collision detection method is exact, as it verifies if any possible future position of each vertex is intersecting with any point of the other volume.

An optimization technique to improve the efficiency of the detection method is given by [20]. *Back-face culling* can be used to avoid checking unnecessary boundary elements of the bounding volume. With this purpose, this technique does not consider vertices or edges that cannot be involved in the collision, given the direction of movement. An edge is culled if its normal has a negative projection on the motion vector, and only vertices in edges that have not been culled are considered. Informally, if

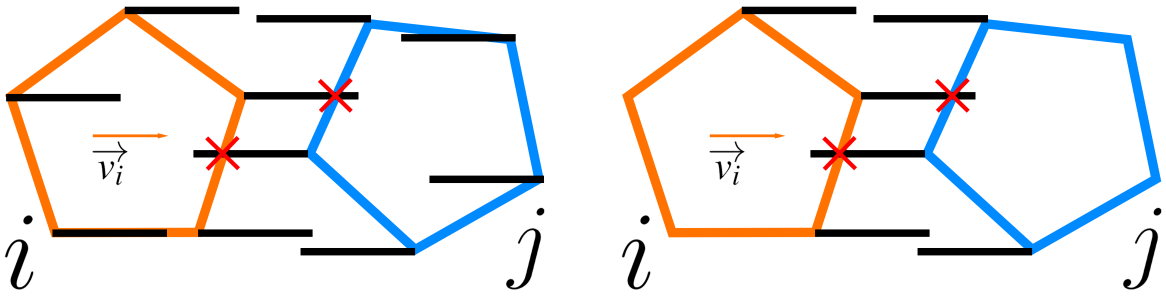


Figure 2.1: Ray-tracing collision detection. Rays being traced from all vertices (left), and only from relevant vertices using *back-face culling* (right). Not at scale.

an edge is in the back of the bounding polygon, the back edge is not considered and the back edge's vertices will not be used to trace rays.

Figure 2.1(a) illustrates the ray tracing process and the future collisions detected (marked by the red Xs). The black lines represent the rays, being cast in the direction of the movement, $\overrightarrow{v_i}$, from the bounding volume i, and in the opposite direction from the bounding volume j's vertices. Figure 2.1(b) illustrates the optimized process: only vertices relevant for future collisions - considering the movement direction - are used to cast rays.

2.2.7.C Bounding Cylinders

The ray-tracing method described is only exact in two-dimensional space and Bounding Spheres are too conservative in three-dimensional space. [7] uses Bounding Cylinders to model obstacle structures, that albeit still conservative, provide more precise collision detection than Bounding Spheres.

Similar to the modeling of the sphere (center position and radius), a cylinder can be defined by a radius and a line in $\mathbb{R}^3$ defined by an initial point and a direction vector.

This method is more complex to implement than Bounding Spheres and more inaccurate than Ray-Tracing; however, it is the best option of the three if the algorithms must consider three-dimensional space.

2.2.8 Environmental Collision Avoidance

To handle collision avoidance from environmental obstacles, [5] proposes two techniques: force field and steer-to-avoid.

The force field technique consists of a simulated repulsion force originating from the obstacle, used by the agents as a direct avoidance acceleration. This approach has two pitfalls, caused when an agent approaches an obstacle (surrounded by the simulated force field):

1. at an angle precisely opposite to the direction of the force field;
2. perpendicular to the force field;

In the first case, the agent will not steer away from the object and will only decelerate. Oppositely, in the second case, the agent will steer away unnecessarily, as it would not collide. The steer-to-avoid technique only considers obstacles relevant to the agent's path. If an agent detects an obstacle, it calculates the closest point of collision based on its current velocity and computes an acceleration vector to steer away from the calculated point.

The third algorithm in [16] extends *Algorithm 2* by adding environmental obstacle avoidance. To model this additional component in α-agents, a new agent type, β-agent, is created to represent an environmental obstacle. β-agents are virtual representations internal to α-agents created whenever there is a nearby obstacle. The avoidance method is given by the sequential tasks:

1. Determine the indices $\mathcal{N}_i^\beta$ of all obstacles $\mathcal{O}_k$ that are neighbors of i;
2. Create a virtual β-agent at q_{ik} on the boundary of $\mathcal{O}_k$ where $q_{ik} = argmin_{x \in O_k} \|x - q_i\|$;
3. Add the term $\psi_\beta(\|q_{ik} - q_i\|_\sigma)$ to the potential function of agent i corresponding to each virtual β-agent created to simulate an environmental obstacle.

where i must be an α-agent, k is the index of the obstacle (unrelated to our definition of k that represents the discrete time instant), and $\mathcal{N}_i^\beta$ is defined by:

$$\mathcal{N}_i^\beta = \{k \in \mathcal{V}_\beta : \|q_{ik} - q_i\| < r\prime\} \tag{2.14}$$

where $r\prime \in \mathbb{R}^+$ is the interaction range of an α-agent with a β-agent, and $\mathcal{V}_\beta$ denotes the set of indices of all obstacles in the mission plane.

Algorithm 3 (in [16]) extends *Algorithm 2* (in [16]) with the collision avoidance component:

$$u_i^\beta = c_1^\beta \sum_{k \in \mathcal{N}_i^\beta} \phi_\beta(\|q_{ik} - q_i\|_\sigma) n_{iik} + c_2^\beta \sum_{k \in \mathcal{N}_i^\beta} b_{ik}(q)(p_{ik} - p_i) \tag{2.15}$$

where c_1^β and c_2^β are positive constants, ϕ_β is a purely repulsive potential function for an α-agent and a β-agent, with $b_{ik}(q)$ being their adjacency weight. The position and velocity of the obstacle, represented by the β-agent created by the α-agent, are defined by q_{ik} and p_{ik}. The vector n_{iik} is a vector along the line connecting q_i to q_{ik}.

For the discovery of β-agents, [16] assumes α-agents are equipped with range sensors that measure the distance to the closest point of an object. Through simulations, [16] shows *Algorithm 3* can be used for automated rendezvous in mission planes containing static or dynamic environmental obstacles if these can be modeled as connected convex regions with boundaries that are smooth manifolds.

Apart from the perimeter walls, our proposed scenario includes statically-positioned environmental obstacles, so our environmental collision avoidance does not require sophisticated methods such as *Algorithm 3* in [16].

2.3 Contributions

The solution proposed in this Chapter presents a novel method to achieve rendezvous to multiple dynamic desirable areas without assuming the connectedness of the network topology while providing guarantees for an environment without agent-agent collisions in the presence of noisy measurements:

- There is no apriori information regarding the number, positions, and movement dynamics of the rendezvous areas. Additionally, this number is not fixed: new rendezvous areas are created arbitrarily and areas are removed from the mission plane by being explored by the agents.
- The network topology does not need to be initially connected. Section 2.7.3 provides simulation examples where the initial topology is disconnected and the agents reach a single cluster.
- Due to the position and velocity measurements being corrupted by noise, they are estimated as worst-case sets in order to achieve effective collision avoidance.
- The collision avoidance method implemented provides guarantees of an environment without agent-agent collisions with each agent predicting the worst-case set of all possible positions for its neighboring agents considering their past known positions by extending the worst-case sets estimated during the measurement process.

Unlike the literature examples presented in Section 2.2, the proposed solution does not require apriori knowledge of the number and position of agents, leader/follower protocols, direct agent-to-agent communication, or agents to localize themselves or transmit information. Additionally, our implementation focuses on the following objectives:

- Deploying to networks of large size. There is no computational limit on the number of allowed agents in the network or a limit to the mission plane size - although the convergence efficiency will generally decrease if the size of the mission plane is larger than the range of the communication towers. This contribution is computationally infeasible with pure-consensus approaches.
- Power efficiency, which increases the agents' power autonomy. By offloading the localization sensors and only requiring communication receivers, the agents conserve more power, which can result in increased battery autonomy, or an inferior cost if the agents are equipped with smaller batteries - which additionally reduces weight.

- Cost efficiency by requiring fewer sensors, the agents become more inexpensive. If the agents are built with smaller batteries (option provided by the increased power efficiency), the cost is additionally reduced. This contribution motivates networks with more agents.

The contributions presented in this Chapter have been published in [11] and are in review in [12].

2.4 Problem Statement

In this Chapter, it is addressed the problem of a multi-agent system converging and following multiple dynamic rendezvous areas using a partially-decentralized control law. The mobile agents have no localization sensors and receive position and velocity updates from directional broadcasts by the communication towers.

The mission plane is assumed to be defined by a time-varying utility function which attributes a numerical value to each position of the mission plane, regarding its quality to the mission objective. In this algorithm, this function is known or inferred by the communication towers and included in the broadcast to the agents.

The communication towers are responsible for measuring and broadcasting the positions and velocities of the agents. For this purpose, the towers are equipped with directional antennae.

The design choice of having directional antennae is justified by the design goal of being able to deploy this algorithm in networks of large size. An antenna that measures and broadcasts the physical state of every agent in a radius can be too computationally intensive - depending on the crowdedness of the mission plane - which would require more expensive hardware and higher power usage, opposing the premises of an inexpensive and power-efficient system. As the implementation of a communication tower can be seen as a fixed structure bounded to the mission plane's boundaries or a more advanced mobile agent, power efficiency is a priority.

An immediate consequence of a directional antenna is its broadcast is only transmitted to a strip of the mission plane. It is assumed the antennae can rotate, so their transmission strips are not static. These strips are defined as circle segments with a fixed angle originating in their tower's position and extend to the mission plane's perimeter.

The measurements taken by the towers are noisy. As it is required for this algorithm to guarantee an environment without agent-agent collisions - since collisions could cause the loss of agents - the towers transmit each agent's position in the form of a set-valued estimate, which is assumed to be equivalent to a convex polytope containing the exact position. The agents consider this polytope in their collision avoidance - planning for the worst case. The velocity is also noisy; nevertheless, it is transmitted as a non-exact two-dimensional vector due to not being required to consider collisions.

The agents have no localization capabilities, only having sensors to verify if the desired movement

is executed, and limited communication capabilities: an agent can only receive external transmissions from the communication towers, which include the agent's position and velocity. An agent's dynamic model is given by the discrete double integrator:

$$\begin{aligned} q_i(k+1) &= q_i(k) + v_i(k), \\ v_i(k+1) &= v_i(k) + u_i(k), \\ i &= 1, 2, \ldots, n \end{aligned} \tag{2.16}$$

where q_i, v_i, and u_i represent agent i's exact position, velocity, and control input, respectively.

2.5 Proposed Solution

2.5.1 Mission Plane

The mission plane defines the area in which the agents can traverse. An initial setup step when an agent is added to the network notifies the agent of the position of the boundaries. To prevent the agents from moving out-of-bounds, they are virtually repelled from the perimeter - similarly to the *Separation* component in Section 2.5.3.

Additionally, the mission plane is responsible for representing rendezvous (desired) areas, undesired areas, and unauthorized areas, through its utility function. In order to have a mathematical representation of areas, instead of points, the utility function is a sum of paraboloids, each representing an area. Having paraboloids allows for the creation of multiple areas with different desirability (utility) values, different areas of effect, or different utility declivities. Formally, the paraboloids used are defined as:

$$\frac{\kappa}{(\phi x)^2 + (\tau y)^2 + \omega} \tag{2.17}$$

where κ, $\omega \in \mathbb{R}^+$ modulate the paraboloid's height, and ϕ, $\tau \in \mathbb{R}^+$ modulate its slope.

Rendezvous Areas Rendezvous areas are defined by paraboloids with positive utility values, $\kappa \in \mathbb{R}^+$.

Undesired Areas Contrarily, undesired areas are defined by paraboloids with negative utility values. Formally, the κ parameter that defines their paraboloids is negative, $\kappa \in \mathbb{R}^-$. Albeit areas with no value for the rendezvous mission, they are still valid for the agents to traverse.

Unauthorized Areas Unauthorized Areas consist of areas in which the agents are not allowed to

traverse, simulating real-world obstacles. These environmental obstacles are statically-positioned, and their boundaries are transmitted to the agents in the initial setup step - with the mission plane's boundaries. These areas are not defined by paraboloids, being defined by attributing a predetermined value to their areas in the utility function. While rendezvous areas have different utility values based on the distance to its center, which justifies the use of paraboloids, the unauthorized areas have a single utility value.

2.5.1.A Parameter Adjustability

The paraboloid-defined areas are dynamic in terms of position and utility value. When an agent explores a rendezvous area, its utility value decreases continually until the agent stops exploring or the utility value reaches the threshold for removal - at which point the paraboloid is removed from the utility function. Oppositely, the value of a rendezvous area increases while no agent is exploring it until it reaches a predetermined maximum. Similarly to the utility value, the position of these areas is dynamic, changing arbitrarily.

2.5.2 Communication Towers

Due to the agents not having localization capabilities, they rely on the transmissions from the communication towers to determine their positions in the mission plane. Thus, the towers are responsible for measuring and broadcasting the positions and velocities of the agents.

As a result of their broadcasts only being transmitted to a strip of the mission plane, the towers only measure the physical state - position and velocity - of the agents that will receive the transmission, i.e., agents within the strip.

Figure 2.2(a) illustrates a tower's strip in blue. The tower, $\mathcal{T}_1$, targets agent i as the center of its current broadcast, measures the position (calculating the estimate polytope) and velocity of agents i and j, and broadcasts the data to these agents. In the figure, α represents the strip's angle: angle between the boundary vectors that define the strip's two-dimensional projection's lateral limits, $\mathcal{T}_{\alpha_1}$ and $\mathcal{T}_{\alpha_2}$. Figure 2.2(b) shows a communication strip in a simulated environment: the four agents within the strip receive the broadcast, despite the presence of environmental obstacles (represented by the black squares) between the tower and the agents.

2.5.2.A Communication Model

At each discrete time instant, a tower targets a new agent in which to center its transmission strip, measures the physical state of the agents within the strip, and broadcasts the data.

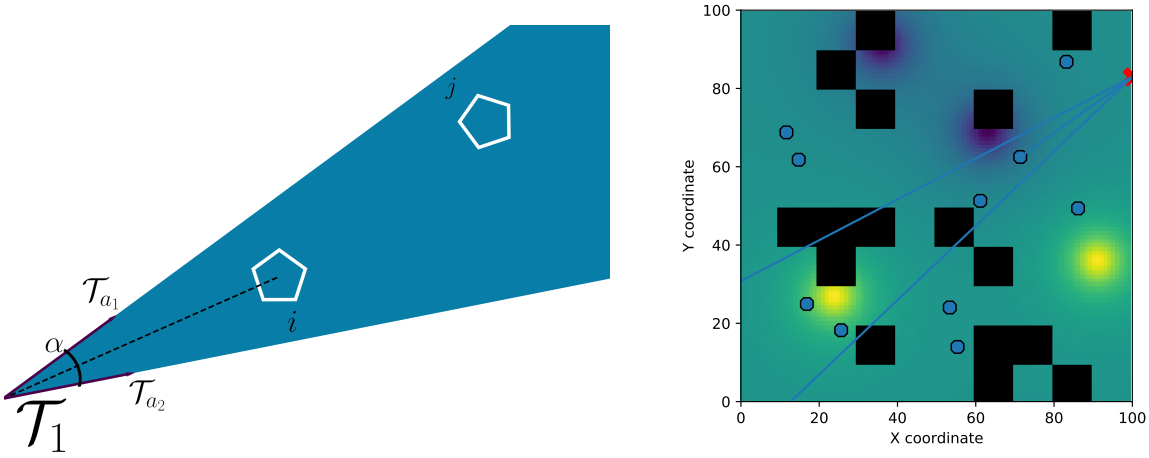


Figure 2.2: Communication tower notation (left - not at scale) and illustration of the broadcast strip and target in a simulated environment with the tower marked by the red X (right).

2.5.2.B Targeted Agent

Each tower iterates through the agent set in a round-robin fashion, targeting each agent. In environments with multiple towers, each tower follows the former process independently. It is left for future work the creation of a heuristic to choose a suitable agent to target - prioritizing areas with sparser agents over areas with dense clusters - and the implementation of a communication protocol between towers - to mitigate redundant or duplicate broadcasts. These functionalities increase the convergence rate of the algorithm and decrease the power usage of the system by minimizing the unnecessary use of the towers.

2.5.2.C Neighbors Definition

For an agent to be considered in a broadcast, it must be within the transmission strip, which requires the agent's position-estimate-polytope to be contained by the boundaries of the circle segment that defines the strip's projection on the mission plane. The set of agents in the strip - centered on agent i at the discrete time instant k - is defined as:

$$\begin{aligned} \mathcal{N}(k) = \{ j \in \mathcal{V} : \\ & \measuredangle(\overrightarrow{t_i}, \overrightarrow{t_j}) \leq \frac{\alpha}{2} \\ & X_j(k) \cap \mathcal{T}_\alpha = \emptyset \} \end{aligned} \tag{2.18}$$

where α is the transmission angle, $\overrightarrow{t_i}$ and $\overrightarrow{t_j}$ are vectors starting in the tower's position, ending in agent i's and j's polytopes' centroids, respectively, with $\measuredangle(\overrightarrow{t_i}, \overrightarrow{t_j})$ being the angle between the two

vectors. $\mathcal{X}_j(k)$ is agent j's position-estimate-polytope which cannot intersect the boundaries of the strip's projection on the mission plane, $\mathcal{T}_\alpha$, created by the transmission angle. Informally, for agent j to belong to $\mathcal{N}(k)$, it has to be fully contained between the strip's boundaries, without overlap. Figure 2.2(a) illustrates the definition of α and the limits of the strip, $\mathcal{T}_{\alpha_1}$ and $\mathcal{T}_{\alpha_2}$, grouped in $\mathcal{T}_\alpha$.

The neighbor set for every agent i in the strip will be the full set of agents in the strip: $\mathcal{N}_i(k) = \mathcal{N}(k)$, which includes agent i.

In environments with multiple towers, each tower will have a different strip at k. For notation simplicity, we refer to the neighbor set of a single tower, at k, with $\mathcal{N}(k)$. When necessary, $\mathcal{N}(k, \mathcal{T}_1)$ and $\mathcal{N}(k, \mathcal{T}_2)$ will be used to differentiate two neighbor sets at time k from any two towers, $\mathcal{T}_1$ and $\mathcal{T}_2$.

2.5.2.D Broadcast Data

The data broadcasted by a tower, $\mathcal{T}_1$, is composed by:

- The position-estimate-polytope and noisy velocity vector of each agent in the strip, $i \in \mathcal{N}(k, \mathcal{T}_1)$
- The utility function. It must be broadcasted every time step because it is time-varying.

2.5.3 Individual Agent Model

In the literature, the rendezvous problem is commonly solved with consensus-based approaches. Whenever the network is composed of multiple isolated connected components, each part achieves different consensus values. In order to circumvent this problem, most implementations rely on the agents having full knowledge of the network to accomplish a single final consensus value. One design goal of this algorithm is the ability to be deployed in networks of large size, as such, agents knowing the full network topology would be infeasible. Consequently, to avoid the problem of multiple final consensus values, the implementation of our agent movement is based on flocking.

By using a flocking-based movement solution, the agents are continually nearing their neighbors and dynamically exploring new areas to find previously unknown clusters. This reduces the final number of agent clusters, possibly to one in certain conditions, in the presence of initially disconnected network topologies.

The agent movement extends the original flocking rules with three additional components tailored for the proposed environment.

Separation Component The *Separation* component is responsible for repelling agents as a naive implementation of a collision avoidance mechanism. Formally, $u_i^s(k)$ is defined as:

$$u_i^s(k) = \frac{1}{|\mathcal{N}_{i,d}(k)|} \sum_{j \in \mathcal{N}_{i,d}(k)} c_i(k) - c_j(k) \tag{2.19}$$

where $\mathcal{N}_{i,d}$ is the set of agent i's neighbors within d unit distances.

Cohesion Component The *Cohesion* component is responsible for decreasing the overall distance between neighbors and preventing agents to excessively separate. Formally, $u_i^c(k)$ is defined as:

$$u_i^c(k) = \frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} c_j(k) - c_i(k) \tag{2.20}$$

In *Separation*, only neighbors closer than d are considered. This is justified by separating from distant agents being counterproductive to the mission objective of rendezvousing. As *Cohesion* aligns with this objective, all neighbors are considered.

Alignment Component The *Alignment* component causes a cluster of agents to align their direction vectors. Formally, $u_i^a(k)$ is defined as:

$$u_i^a(k) = \frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} v_j'(k) - v_i'(k) \tag{2.21}$$

where v_i' and v_j' are the velocities of agents i and j as received in the broadcasts, respectively.

Attraction Component *Attraction*, along with *Utility* and *Randomness*, is a custom component, supplementary to the original flocking rules. It is responsible for moving clusters to higher-utility positions. For this purpose, each agent is attracted to neighbor high-utility agents and repelled from neighbor lower-utility agents. A higher-utility agent is an agent in a higher-utility position relative to the agent computing the control law. Formally, $u_i^{attr}(k)$ is defined as:

$$u_i^{attr}(k) = \frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} (h(c_j(k)) - h(c_i(k)))(c_j(k) - c_i(k)) \tag{2.22}$$

where $h(c_i(k))$ and $h(c_j(k))$ are the utility values at agent i's and agent j's centroid positions, respectively.

Utility Component The *Utility* component is responsible for moving the agents in the direction that locally maximizes the utility function by following the function's gradient in their position, which results in agents converging in higher-utility areas. When the utility function is not diferentiable at the agent's position, this component has a zero-vector value. Formally, $u_i^u(k)$ is defined as:

$$u_i^u(k) = \nabla h(c_i(k)) \tag{2.23}$$

Randomness Component In order to avoid indefinite movement deadlocks, the *Randomness* component, $u_i^r(k)$, adds a unit vector with a random angle to the agent acceleration vector.

2.5.3.A Desired Movement Law

The desired movement law is constructed as a weighted average of the multiple components. Each weight translates to the importance of the respective component for the mission objective. For instance, if the mission objective is to explore the maximum number of rendezvous points without the need for agent convergence, then the *Utility* component's weight will be higher than its *Cohesion* counterpart.

Formally, the desired movement law, $\hat{u}_i(k)$, is defined with:

$$\hat{u}_i(k) = \theta \cdot u_i^s(k) + \beta \cdot u_i^c(k) + \gamma \cdot u_i^a(k) + \delta \cdot u_i^{attr}(k) + \epsilon \cdot u_i^u(k) + \zeta \cdot u_i^r(k) \tag{2.24}$$

where all weights $\theta, \beta, \gamma, \delta, \epsilon, \zeta \in \mathbb{R}^+$, with all component vectors normalized.

The process of tunning the weights starts with understanding the mission objective and how each component contributes to it. Depending on their importance, their weights are attributed, decreasing in value with decreasing priority. To calibrate these values, multiple simulations in a variety of environments are executed, verifying if the desired emergent behavior is achieved.

2.5.4 Agent Collision Avoidance

The implemented collision avoidance mechanism has two stages: collision detection (Polytope Propagation and Planned Collision Detection) and Movement Restriction. The former calculates if the moving agent will collide with any of its neighbors if it moves the maximum allowed distance per update - D_{max} - in its desired direction. The latter culls the desired movement to prevent collisions.

2.5.4.A Polytope Propagation

Before detecting future collisions, the calculating agent's internal representations of the position polytopes of its neighbor agents are extended to contain all the positions to which they could have moved since their position was transmitted in the broadcast. This extended polytope includes all points whose distance to any position in the polytope is less than D_{max}.

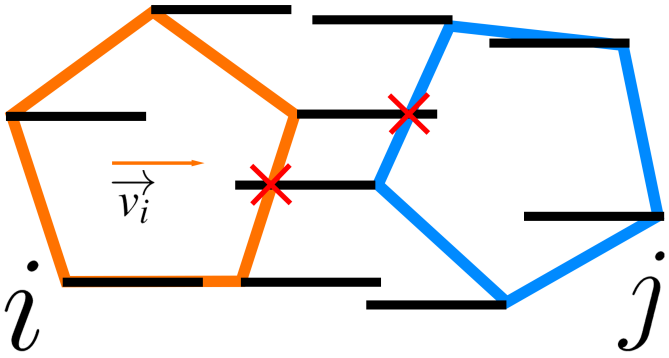


Figure 2.3: Ray-tracing collision detection. Rays are traced from both polytopes: in the direction of the desired movement, $\overrightarrow{v_i}$, from agent i's polytope (orange shape), and in the opposite direction from agent j's extended polytope (blue shape). The future collisions are marked by the red Xs. Not at scale.

2.5.4.B Planned Collision Detection

The collision detection implementation utilizes the propagated polytopes, and detects possible collisions between every agent pair (i, j), with i being the calculating agent and $j \in \mathcal{N}_i(k) \setminus i$.

A ray is cast from each vertex of agent i's polytope, with a length equal to the maximum allowed distance per update and direction equal to the desired movement direction. An intersection between this ray and agent j's polytope represents a possible future collision. This ray casting process is repeated from the vertices of agent j's polytope, with rays in the opposite direction. Figure 2.3 illustrates the process: agent i is represented by the orange shape, agent j by the blue shape, and the possible future collision points by the red Xs.

The final step of this phase is to calculate the minimum distance from the origin of each ray to its intersection with the opposite polytope, which represents the maximum distance the agent can move in its desired direction without collisions, D. Figure 2.4(a) presents a visual definition of D.

The implemented ray-tracing method implemented for collision detection only provides exact results in two-dimensional space. In two dimensions, the first collision point between two convex polytopes corresponds to a vertex (from either polytope), and as such, by tracing the future path of each vertex and verifying if it intersects with another convex polytope, this method determines if there will be a collision. In three dimensions, convex polytopes can have edge-edge collisions - casting rays from the vertices would not detect this type of collision. For the proposed system to be used in three-dimensional space while guaranteeing an environment without agent-agent collisions, the collision detection method must be adapted or replaced. For three dimensions, the ray-tracing method could be modified to consider the two-dimensional projection of the three-dimensional polytopes in the desired movement plane. By using the projections, the implemented method would remain unchanged, and the guarantee of exactness would still apply. Oppositely, two possible replacements are described in Section 2.2.7: *Bounding*

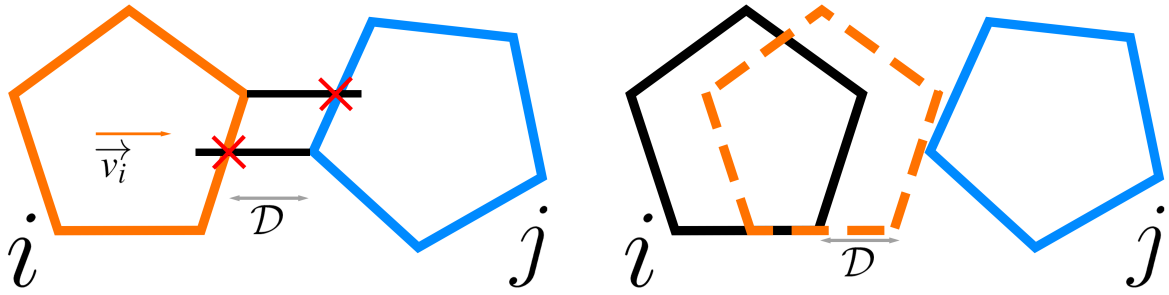


Figure 2.4: Minimum distance between a ray's origin and its intersection with the opposite polytope, represented by D (left). Position of agent i's polytope after moving D distance units in the desired direction, $\overrightarrow{v_i}$ (right). Not at scale.

Spheres and *Bounding Cylinders*. These methods do not result in undetected collisions, although, due to being conservative, can result in false positives.

2.5.4.C Movement Restriction

In order to guarantee an environment without agent-agent collisions, the magnitude of agent i's control input is culled to equal D - the distance calculated in the previous step. Figure 2.4(b) illustrates the position of the agent's polytope if the desired movement magnitude is equal to D. It is shown D is the maximum distance agent i can move in its desired direction, $\overrightarrow{v_i}$ (as represented in Figure 2.4(a)), without colliding with agent j.

A pair of neighbor agents sufficiently distanced to cause a possible collision will generally desire to move in a similar direction, mostly due to the *Utility* component, or in a direction that separates them, mostly due to *Separation* and *Formation*. As such, agent i's movement distance is restricted, because, generally, the obstacle-agent in its path will eventually move.

2.5.5 Environmental Collision Avoidance

We consider two types of environmental obstacles: the mission plane's perimeter walls and generalized statically-positioned obstacles. The first type is avoided using the force field method presented in Section 2.2.8 along with its pitfalls when an agent approaches an obstacle surrounded by a force field:

- at an angle precisely opposite to the direction of the force field, which will only cause it to decelerate and not steer away from the obstacle.
- perpendicular to the force field, which will cause it to steer away unnecessarily, as it would not collide.

In the case of the mission plane's walls: the first pitfall is mitigated by the agent being attracted towards its neighbors from near the perimeter, and the second pitfall is generally desirable - it mitigates the occurrence of the first pitfall and increases the exploration factor by moving the agent towards a position surrounded by available mission plane areas (instead of being close to the wall). Our agents cannot use the steer-to-avoid method presented in 2.2.8 because the walls cannot be circumvented.

The method for avoiding the second type - presented in the remainder of this section - is analogous to the agent collision avoidance method in Section 2.5.4. Instead of maximizing the distance that results in a collision-free movement, this method prevents the agent from moving if it would result in a collision with an environmental obstacle, as there is no interest for the mission objective for the agent to move closer to the obstacles.

2.5.5.A Polytope Propagation

The environmental obstacles are static, so this step is not required.

2.5.5.B Planned Collision Detection

To detect a collision with the environment, the agent calculates if the future position resulting from its desired movement is inside an unauthorized area. As demonstrated in simulations that consider mission planes with unauthorized areas, this method does not provide exact results: a subset of points of the polytope can be intersecting the static environmental obstacle, with the polytope center - which is the reference point for the future positions - not colliding. This is explained in more detail in Section 2.8.1 with additional limitations of this algorithm and respective possible solutions.

2.5.5.C Movement Restriction

Unlike the analogous in agent collision avoidance - in Section 2.5.4.C - it is not advantageous to the mission objective for an agent to minimize its distance to an environmental obstacle. As such, when an agent detects its desired movement will result in a collision with an environmental obstacle, it stops. This movement restriction is justified by the two following general cases:

- if the agent does not have neighbors, its Randomness component will eventually cause it to move away from the obstacle.
- if the agent has neighbors, they will eventually attract it away from the obstacle.

Formally, the result of this method only has two possible values: $D_{env} \in 0, 1$. If the desired movement would result in a collision with an environmental obstacle, $D_{env} = 0$.

2.5.6 Final Control Law

The final control law will consider the agent's desired movement direction and the maximum distances allowed by the collision avoidance methods. The control input given to the agent will have the desired direction and a magnitude equal to the minimum between the resulting distances from the agent and environmental collision avoidance methods. Formally, $u_i(k)$ is defined as:

$$u_i(k) = \frac{\hat{u}_i(k)}{\|\hat{u}_i(k)\|} \cdot \min(D, D_{env}) \tag{2.25}$$

with $\hat{u}_i(k)$, D, and D_{env} being defined in Equation (2.24) and Sections 2.5.4.B and 2.5.5.B, respectively.

2.6 Convergence Analysis

In this section, we give a convergence result regarding the proposed flocking algorithm by reformulating it as a consensus system with a virtual leader given by the utility values of the players. Given the proposed collision avoidance mechanism, the value θ accounting for the weight of the *Separation* velocity vector can be negligible as the nodes will not collide. Moreover, given that positions are corrupted by noise, the theoretical results have to be derived for the centroids of the polytopes. This means that not considering collisions in our analysis and proving convergence translates to convergence to a ball around the target that is determined by the noise magnitude and the number of nodes (since when agents cannot collide they will prevent others from entering within their polytope).

Given the above considerations, we now present the main theorem of this Chapter where convergence is proved when disregarding the *Separation* component and the collision avoidance method and where the symbol will be used to denote the Kronecker product.

Theorem 1. *Consider n nodes with the centroids of $\mathcal{X}_i(k)$ given by $c_i(k), \forall i \leq n$ following the algorithm in (2.24) with a sufficiently small ϵ gradient step, in a mission characterized by a concave utility function h. Then, the velocity iteration is equivalent to a consensus algorithm perturbed by a virtual leader, i.e.,*

$$v(k+1) = Wv(k) + z^{\text{virtual}}(k) \tag{2.26}$$

where W corresponds to a consensus matrix and z^{virtual} to an input of the system. Moreover, the position of the nodes evolves as a perturbed gradient ascent algorithm and therefore $\lim_{k\to\infty} \|x(k) -_{2n} x^\star\|_2 \leq \varphi(\zeta, \eta)$ where $x^\star := h(x)$ and $\varphi(\zeta, \eta)$ is some constant dependent on the random input magnitude and the noise level of the measurements η.

Proof. We first write the velocity iteration equation as:

$$v_i(k+1) = v_i(k) + \beta \cdot \mathcal{U}_i^c(k) + \gamma \cdot \mathcal{U}_i^a(k) + \delta \cdot \mathcal{U}_i^{attr}(k) + \epsilon \cdot \mathcal{U}_i^u(k) + \zeta \cdot \mathcal{U}_i^r(k) \tag{2.27}$$

$$= (1-\gamma)v_i(k) + \gamma \frac{1}{\mathcal{N}_i(k)} \sum_{j \in \mathcal{N}_i(k)} v_j(k) + z_i^{\text{virtual}}(k) \tag{2.28}$$

where $z_i^{\text{virtual}}(k) = \frac{1}{\mathcal{N}_i(k)} \sum_{j \in \mathcal{N}_i(k)} (\beta + \delta(h(c_j(k)) - h(c_i(k))))(c_j(k) - c_i(k)) + \epsilon \nabla h(c_i(k)) + \zeta \mathcal{U}_i^r(k)$. The iteration can be expressed in matrix form as:

$$v(k+1) = \underbrace{W_\gamma v(k)}_{\text{consensus}} + \underbrace{z^{\text{virtual}}(k)}_{\text{virtual leader}}. \tag{2.29}$$

Notice that the input $z^{\text{virtual}}(k)$ without the noise term would only be zero for the maximum of function h and when nodes are all in the same position. Moreover, this vector has bounded norm and converges to zero since function h is concave. It is well known in the theory of consensus, that (2.29) converges to consensus state such that the zero of the input signal is achieved provided that the support graph of W_γ is connected, i.e., velocity zero for all nodes. However, in general, this does not hold and the iteration (2.29) will make the various clusters converge to their own consensus velocity vector. Given the existence of a random component, instead of convergence, we have a bounded $v(k)$ with a norm dependent on ζ.

The position iteration is given by:

$$x_i(k+1) = x_i(k) + v(k) \tag{2.30}$$

where $v(k)$ is a perturbed gradient signal. This is a gradient ascent iteration that converges to the maximum of a concave function provided an appropriately sufficiently small step size is selected. Due to the existence of noise and the random input, the conclusion

$$\lim_{k \to \infty} \|x(k) -_{2n} x^\star\|_2 \leq \varphi(\zeta, \eta) \tag{2.31}$$

follows since each cluster can be viewed as independent gradient ascents. □

2.7 Simulation Results

To evaluate the algorithm created, we executed multiple simulations across a variety of environments. Sections 2.7.1 and 2.7.2 present the common goals and environments over all simulations, with 2.7.3 providing an analytical analysis of the results, and 2.7.4 comparing them to the desired outcome. The

simulation figures and videos are available on GitHub [21].

2.7.1 Simulation Goals

The simulations presented in this section are used to verify the following design objectives:

1. agents move towards and remain in the rendezvous areas while these are in the utility function;
2. agents are repelled by the undesired areas;
3. agents do not enter unauthorized areas;
4. agents do not leave the mission plane;
5. agents do not collide;
6. indefinite movement deadlocks do not occur;
7. agents eventually leave low-utility areas.

2.7.2 Simulation Environment

Every simulation presented in Section 2.7.3 utilizes identical agents, including the control laws' weights, defined in Equation (2.24). Additionally, simulations generally use the default 100×100 mission plane size, 10 agents, and 1 communication tower with a 16-degree strip angle. These parameters are modified in specific simulations when explicitly stated.

The simulations have a pre-imposed hard time limit (generally equivalent to 500 discrete time steps) to be able to be recorded and analyzed. The time limit can result in the configuration of agents presented in the final figure of a simulation to imply that some agents are in a deadlocked state or in an irrelevant position to the mission objective. This misinterpretation is primarily manifested in simulations with dynamic-valued rendezvous areas: the agents reach an area, explore it, the area is removed from the mission plane, and the final configuration illustrates the agents in an area with no rendezvous area. To mitigate this occurrence, each simulation is presented with multiple snapshots of configurations across the simulation duration. Additionally, in simulations with intermediate time steps that do not present useful information for the system analysis - such as the simulation with a single rendezvous area (whose purpose is to test if the agents reach the desired area from an initially disconnected network topology) - only two figures are presented: the initial configuration (at $k = 0$) and the configuration at a time step in which the agents are considered to have fulfilled their objective - generally different from the $k = 500$ hard limit. The fulfillment rule developed to detect if the agents have completed their objectives at time step k requires the following:

- All agents are in a rendezvous area;
- The sum of the distances moved by all agents between time steps $k-1$ and k is below a threshold for two consecutive time steps.

The first requirement is satisfied when each of the agents is at most at 10 distance units from its closest rendezvous area. This value was calculated with the following:

$$d_{rendezvous} = \left\lceil \left\lceil \frac{n}{4} \right\rceil \cdot 1.6 \cdot 2 \right\rceil \tag{2.32}$$

where n is the number of agents (generally 10) and 1.6 is the radius of each agent's polytope. $d_{rendezvous}$ is an upper bound on the maximum distance at which an agent can be from a rendezvous point if all n agents are equally divided in each of the four cardinal directions around a single rendezvous area. In the simulated environments where this maximum distance requirement is increased, it will be explicitly stated. The second requirement is satisfied when the sum of the distances moved by all the agents between a time step $k-1$ and k is less than 3 distance units - 30% of the sum of the maximum allowed distances for the 10 agents - for two consecutive time steps, i.e., this sum is less than 3 between $k-2$ and $k-1$ and between $k-1$ and k. This fulfillment rule - to determine when a simulation is completed - is used for simulations that do not contain moving rendezvous areas or dynamic values, as in those simulations, the agents will never reach a state of completion. In the former, the agents will continuously move towards the moving rendezvous areas for the simulation duration. In the latter, when the agents reach a rendezvous area, the area is considered explored and removed from the mission plane, which results in the agents moving to a new rendezvous area. In the particular scenario with a single statically-positioned undesired area (and no explicitly-defined rendezvous areas), the fulfillment rule cannot be used, as the agents will explore the mission plane for a more desirable area for the simulation duration.

Finally, the simulation videos demonstrate that the behavior of the agents in the final time steps is as expected.

2.7.3 Analytical Analysis

The scenarios included are:

1. Single-maximum utility function - to account for rendezvous missions:
   (a) Static rendezvous area;
   (b) Moving rendezvous area;
2. Single-minimum utility function - to illustrate escape missions:
   (a) Static minimum area;

(b) Moving minimum area;

3. Multiple maxima and minima - to depict the cases of conflicting objectives:

   (a) Static rendezvous areas;

   (b) Static rendezvous areas in a mission plane with unauthorized zones;

   (c) Static rendezvous areas with dynamic values;

   (d) Static rendezvous areas with dynamic values in a mission plane with unauthorized zones;

   (e) Static rendezvous areas with dynamic values and random maxima/minima creation and destruction;

   (f) Static rendezvous areas with dynamic values and multiple communication towers;

   (g) Static rendezvous areas in a 200×200 mission plane with 30 agents and 10 communication towers.

The *"static"* attribute is related to the time-invariant position; when the utility values change, it is explicitly stated with *"dynamic values"*.

In the cases of multiple maxima and minima - case 3 - the minimum areas are also static, and in 3e, they are also created and destroyed randomly. The simulation figures will omit the representation of the communication strip and the target agent (at the time instant the figures illustrate) for simplicity because they do not provide useful data for the analysis of the results. In order to represent the utility function, the simulation figures use a relative color scale: yellow represents positions with the highest utility value, and oppositely, purple corresponds to the lowest utility value. Positions with intermediate values have an intermediate color. As such, in simulations with rendezvous areas with dynamic values - directly correlated to a change in values of the utility function - the colors of the mission plane's positions will change as the simulation progresses.

2.7.3.A Single Static Rendezvous Area

Figures 2.5(a) and 2.5(b) illustrate the configurations at ($k = 0$) - initial configuration - and ($k = 294$) - final configuration - with a single static rendezvous area (simulation type 1a) at the center of the mission plane - represented in yellow. This first example demonstrates that convergence can be achieved in a simple mission plane with a single communication tower, even with an initial disconnected network topology (objective 1). Additionally, the agents do not leave the mission plane (objective 4), the final configuration depicts the effectiveness of the proposed collision avoidance method (objective 5), and there are no indefinite movement deadlocks (objective 6). The realization of objectives 4, 5, and 6 will generally not be explicitly stated in the remainder of the analysis of the simulations for simplicity, and due to always being achieved.

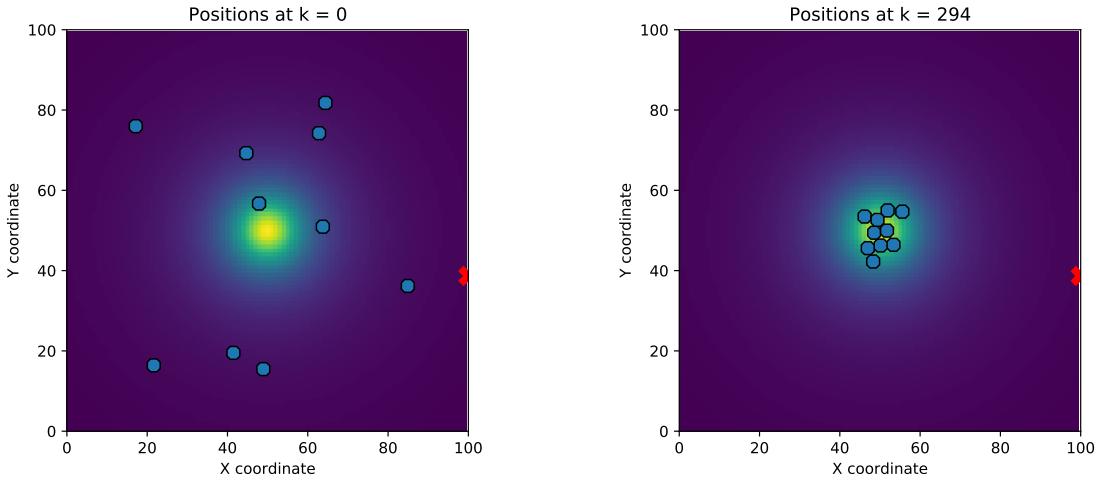


Figure 2.5: Agents configurations at time steps $k = 0$ (left) and $k = 294$ (right) for a simulation with a single static maximum at the center of the mission plane with one communication tower (marked by the red X). Using *Algorithm 1*.

2.7.3.B Single Moving Rendezvous Area

Figures 2.6(a) and 2.6(b) illustrate a similar scenario with a randomly moving single maximum, initially at the center of the mission plane. The agents rendezvous on the desired area, and dynamically track it to its new position.

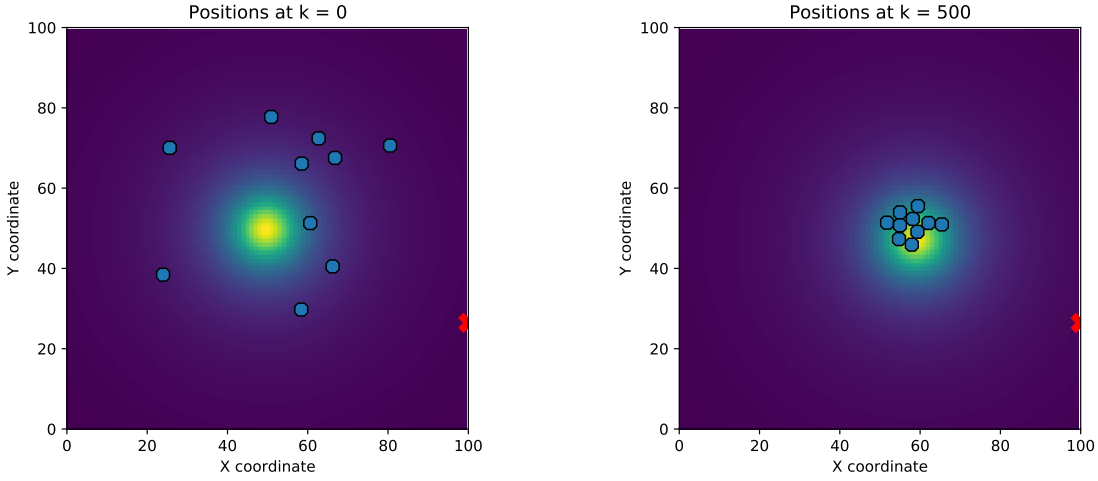


Figure 2.6: Agents configurations at time steps $k = 0$ (left) and $k = 500$ (right) for a simulation with a single moving maximum at the center of the mission plane with one communication tower (marked by the red X). Using *Algorithm 1*.

2.7.3.C Single Static Minimum Area

Figures 2.7(a) and 2.7(b) represent the initial and final configurations for the ten agents in the case of a single static minimum area - type 2a. This simulation type and the simulation in Section 2.7.3.D depict escape missions (from the undesired area). Due to the existence of multiple equally-desirable areas that correspond to the areas not affected by the central minimum, the agents do not create a final single cluster, being repelled by the minimum-desirability area (objective 2).

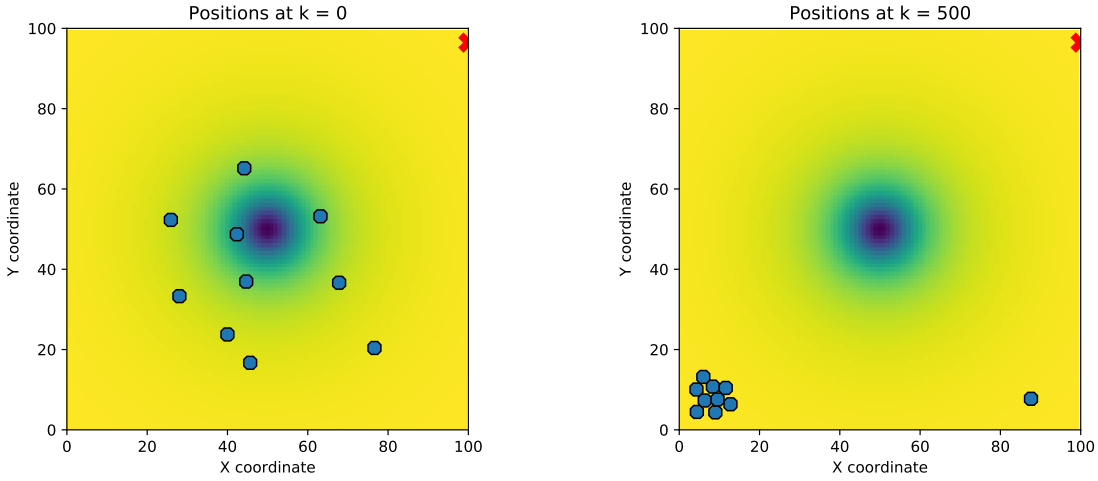


Figure 2.7: Agents configurations at time steps $k = 0$ (left) and $k = 500$ (right) for a simulation with a single static minimum at the center of the mission plane with one communication tower (marked by the red X). Using *Algorithm 1*.

2.7.3.D Single Moving Minimum Area

Similarly, for scenario 2b - single moving minimum area - the agents do not create a single final cluster. This simulation demonstrates that the agents move away from the undesirable moving area (objective 2). Figures 2.8(a) and 2.8(b) show the initial and final configurations.

2.7.3.E Multiple Static Rendezvous Areas

The simplest scenario with multiple rendezvous and undesired areas - type 3a - is illustrated in Figures 2.9(a) and 2.9(b). The agents are able to converge to the two static rendezvous areas while avoiding the undesirable (low-utility) areas (objectives 1 and 2). This simulation environment satisfies the requirements for the use of the fulfillment rule to verify if the agents have completed their objective (and for the simulation to be considered finalized). However, due to the particular configuration of agents reached (as shown in Figure 2.9(b)), the agents are not all considered to be closer than the maximum distance

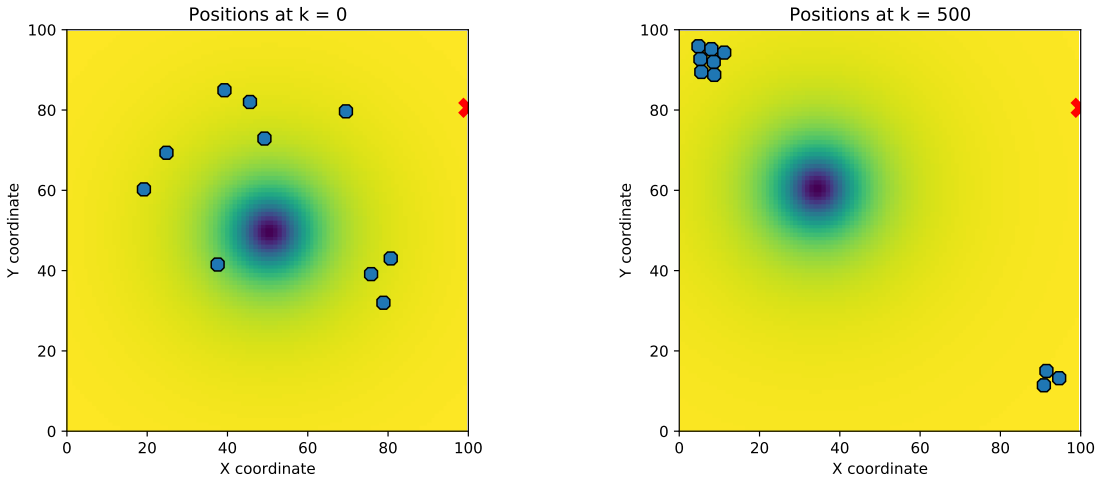


Figure 2.8: Agents configurations at time steps $k = 0$ (left) and $k = 500$ (right) for a simulation with a single moving minimum at the center of the mission plane with one communication tower (marked by the red X). Using *Algorithm 1*.

of a rendezvous area. As such, this simulation is considered to be concluded with the pre-imposed time limit, $k = 500$.

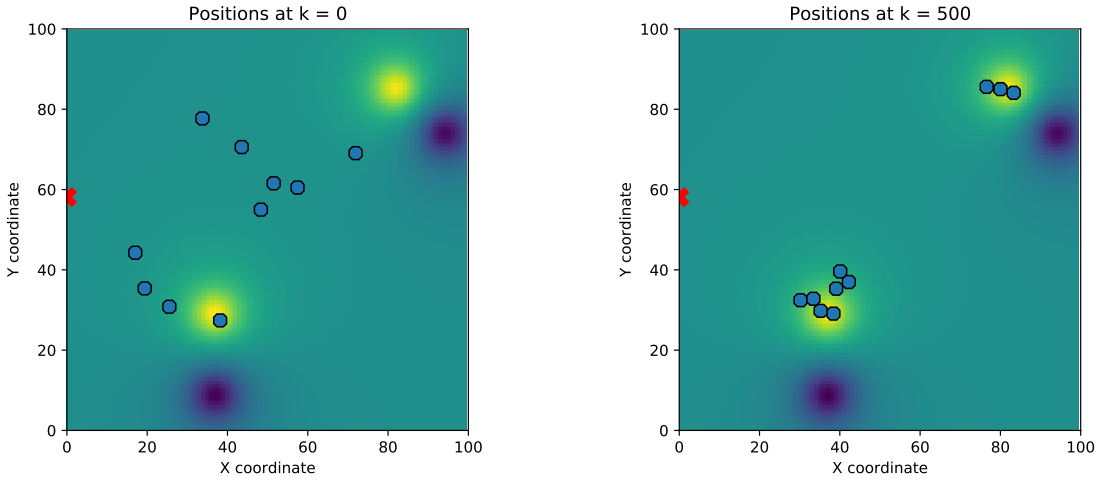
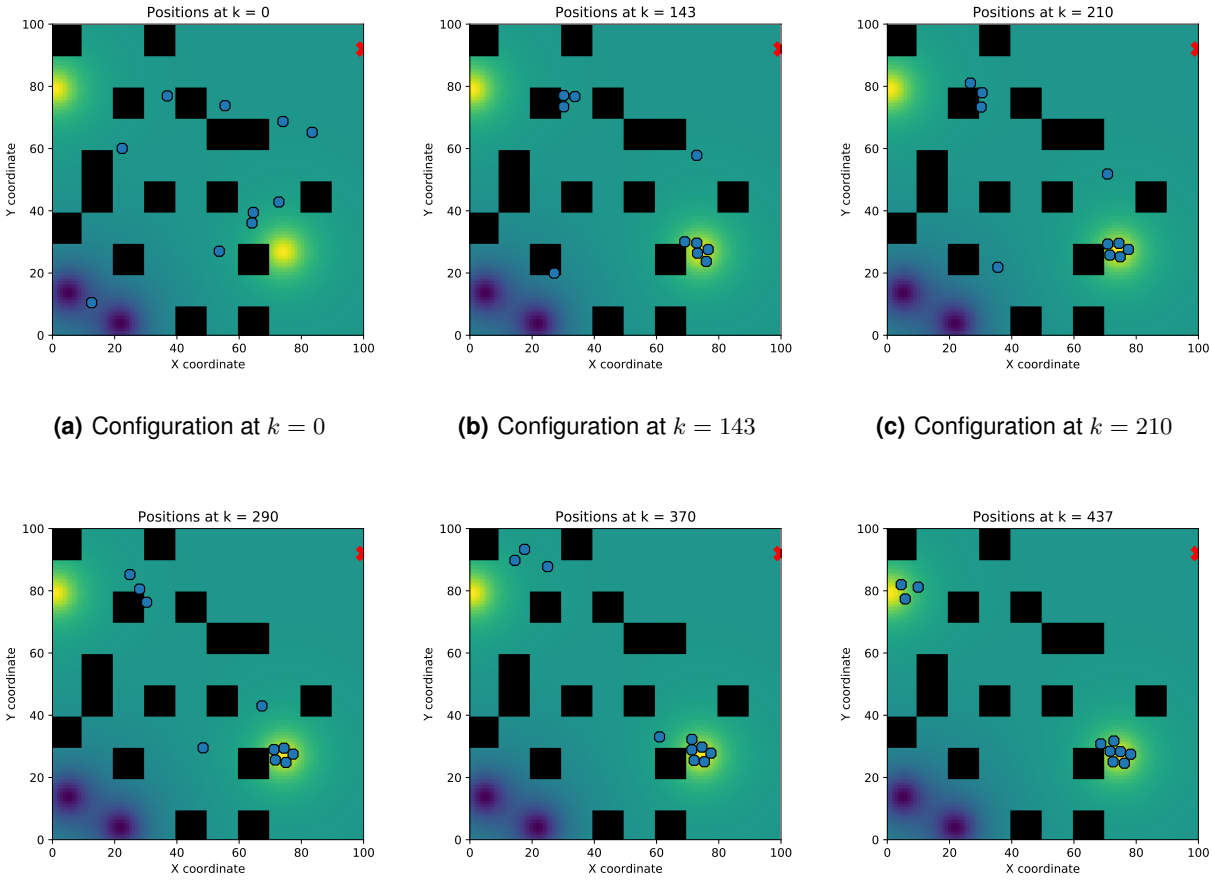


Figure 2.9: Agents configurations at time steps $k = 0$ (left) and $k = 500$ (right) for a simulation with a mission plane with multiple static maxima and minima and with one communication tower (marked by the red X). Using *Algorithm 1*.

2.7.3.F Multiple Static Rendezvous Areas in a Mission Plane with Unauthorized Areas

The simulation of type 3b - multiple static rendezvous and undesirable areas in a mission plane with unauthorized areas - is depicted in Figures 2.10(a) to 2.10(f). The black squares represent the unauthorized areas. Similarly to type 3a, the agents converge to the multiple rendezvous areas while avoiding the low-utility areas (objectives 1 and 2), and additionally, avoiding the unauthorized areas (objective 3). The three agents at the top of Figure 2.10(b) have their path to the rendezvous area obstructed by the unauthorized area. It can be seen in Figures 2.10(c), 2.10(d), and 2.10(e) that the agents circumvent the static environmental obstacle to reach the rendezvous area: illustrated in Figure 2.10(f).



(a) Configuration at $k = 0$ **(b)** Configuration at $k = 143$ **(c)** Configuration at $k = 210$

(d) Configuration at $k = 290$ **(e)** Configuration at $k = 370$ **(f)** Configuration at $k = 437$

Figure 2.10: Agents configurations for a simulation with a mission plane with multiple static maxima, minima, unauthorized areas (black squares), and with one communication tower (marked by the red X). Using *Algorithm 1*.

2.7.3.G Multiple Static Rendezvous Areas with Dynamic Values

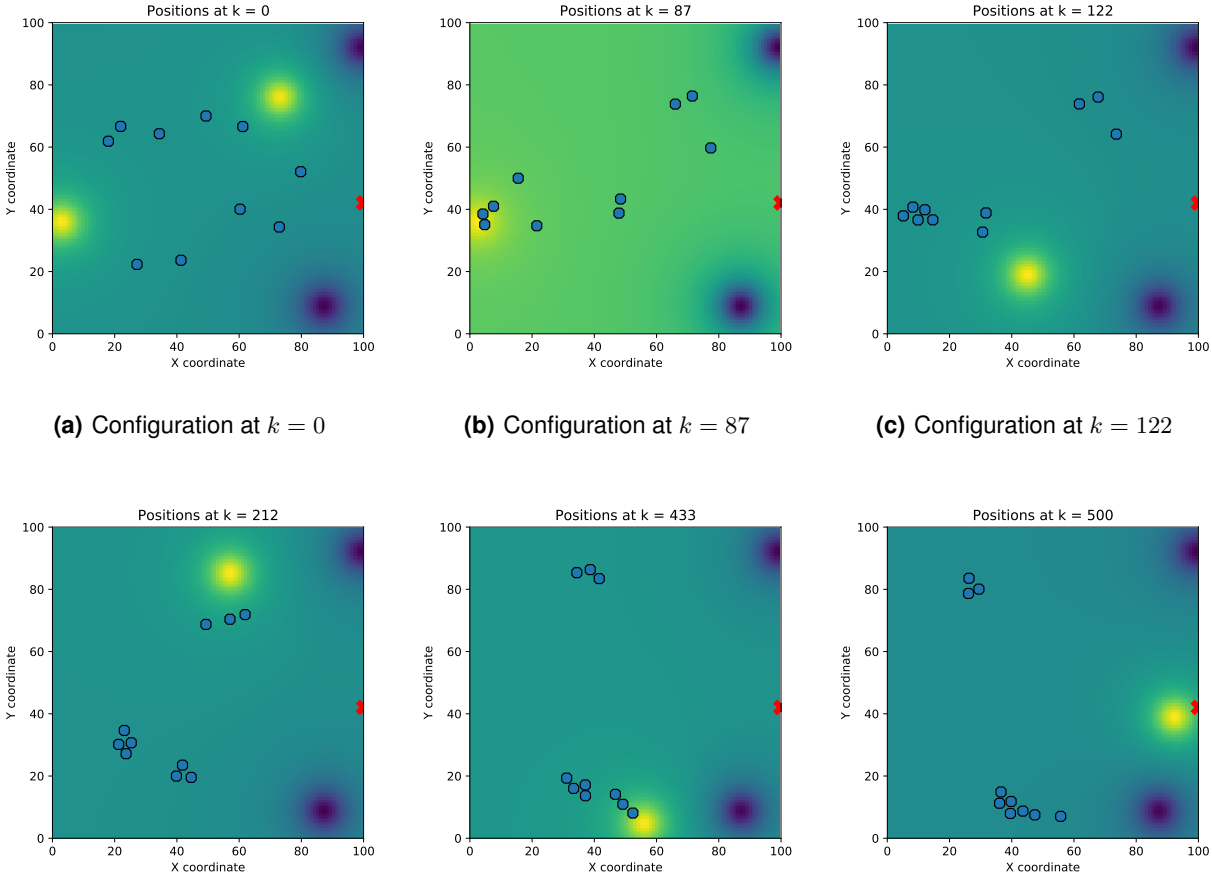
Figures 2.11(a) to 2.11(f) illustrate type 3c - multiple static rendezvous areas with dynamic values and multiple static undesired areas. The rendezvous areas' values are considered dynamic because their values are a direct result of agent exploration. While agents are not exploring a rendezvous area, its utility value increases (until a predefined maximum). Oppositely, while the agents are exploring a rendezvous area, its value decreases until it reaches a minimum - at which point the rendezvous area is removed from the mission plane (due to the paraboloid being removed from the utility function). This simulation demonstrates the agents converge to the multiple rendezvous areas and remain there until the areas are considered fully explored and removed (objective 1). The agents in a rendezvous area detect it has been removed (cluster on the lower left in Figure 2.11(c) detects area on the lower left in Figure 2.11(b) disappeared), and explore the mission plane until they converge to a new one (Figures 2.11(d) and then 2.11(e) - the cluster on the lower left finds a new rendezvous area). This process would have been repeated indefinitely, but the simulations have a pre-imposed time limit. Objective 2 is also achieved.

2.7.3.H Multiple Static Rendezvous Areas with Dynamic Values in a Mission Plane with Unauthorized Areas

Type 3d - multiple static rendezvous areas with dynamic values and multiple static undesired areas in a mission plane with unauthorized areas - is illustrated with Figures 2.12(a) to 2.12(f). The results of simulation 3c persist, and additionally, the agents do not enter the unauthorized areas and generally eventually leave low-utility areas (totaling to objectives 1, 2, 3, 4, 5, 6, and 7). The agent cluster at the lower center of Figure 2.12(e) can be seen avoiding the unauthorized area at its right to reach the rendezvous area at the lower right, almost arriving before the simulation ends (in 2.12(f)).

2.7.3.I Multiple Static Rendezvous Areas with Dynamic Values and Random Creation/Destruction of Rendezvous Areas

Figures 2.13(a) to 2.13(f) illustrate type 3e - multiple static rendezvous areas with dynamic values and multiple static undesired areas in a mission plane with random creation and destruction of maximum and minimum areas. In this type of scenario, in a predefined frequency, one rendezvous or undesired area is either added to or removed from the mission plane. In the presented simulation, six rendezvous areas were destroyed, and ten were created, and three and six low-utility areas were removed and added, respectively. Similarly to simulation types 3c and 3d, when an agent detects the rendezvous area it was seeking has been removed, it explores the mission plane in search of a new one (objective 1). In this simulation type, rendezvous areas are removed due to the random process and agent exploration.

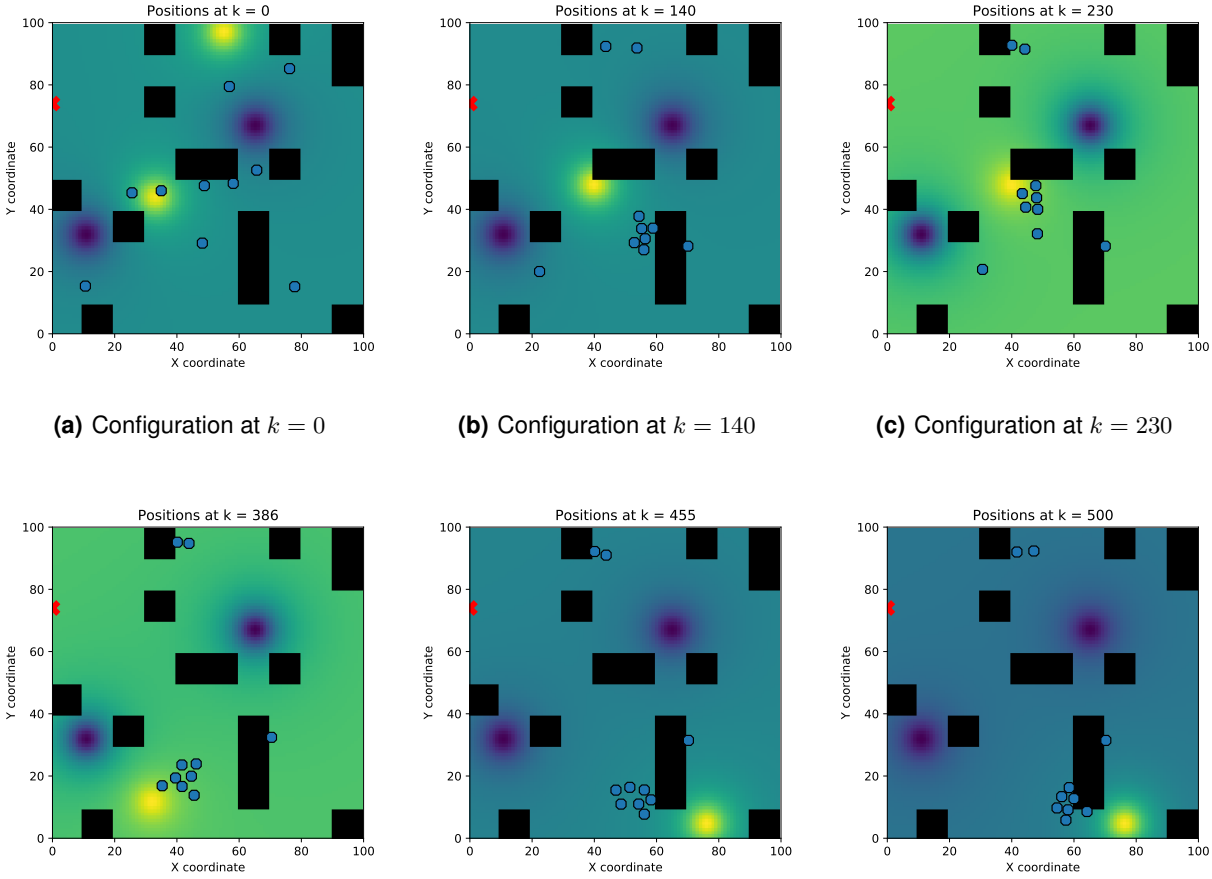


(a) Configuration at $k = 0$ **(b)** Configuration at $k = 87$ **(c)** Configuration at $k = 122$

(d) Configuration at $k = 212$ **(e)** Configuration at $k = 433$ **(f)** Configuration at $k = 500$

Figure 2.11: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values and multiple static minima; and with one communication tower (marked by the red X). Using *Algorithm 1*.

An agent's discovery of the removal of a rendezvous area is attributed to the change in gradient of the utility function at its location, used in the *Utility* component. In the simulation presented, the agents converge to multiple rendezvous areas, and when these are removed, the agents explore the mission plane for new desired areas: in Figure 2.13(c) the two agents at the lower left of the image are nearing a rendezvous area which is removed before Figure 2.13(d); when the agents detect the area is missing, they continue exploring until they reach the rendezvous area in the lower middle area in Figure 2.13(e).



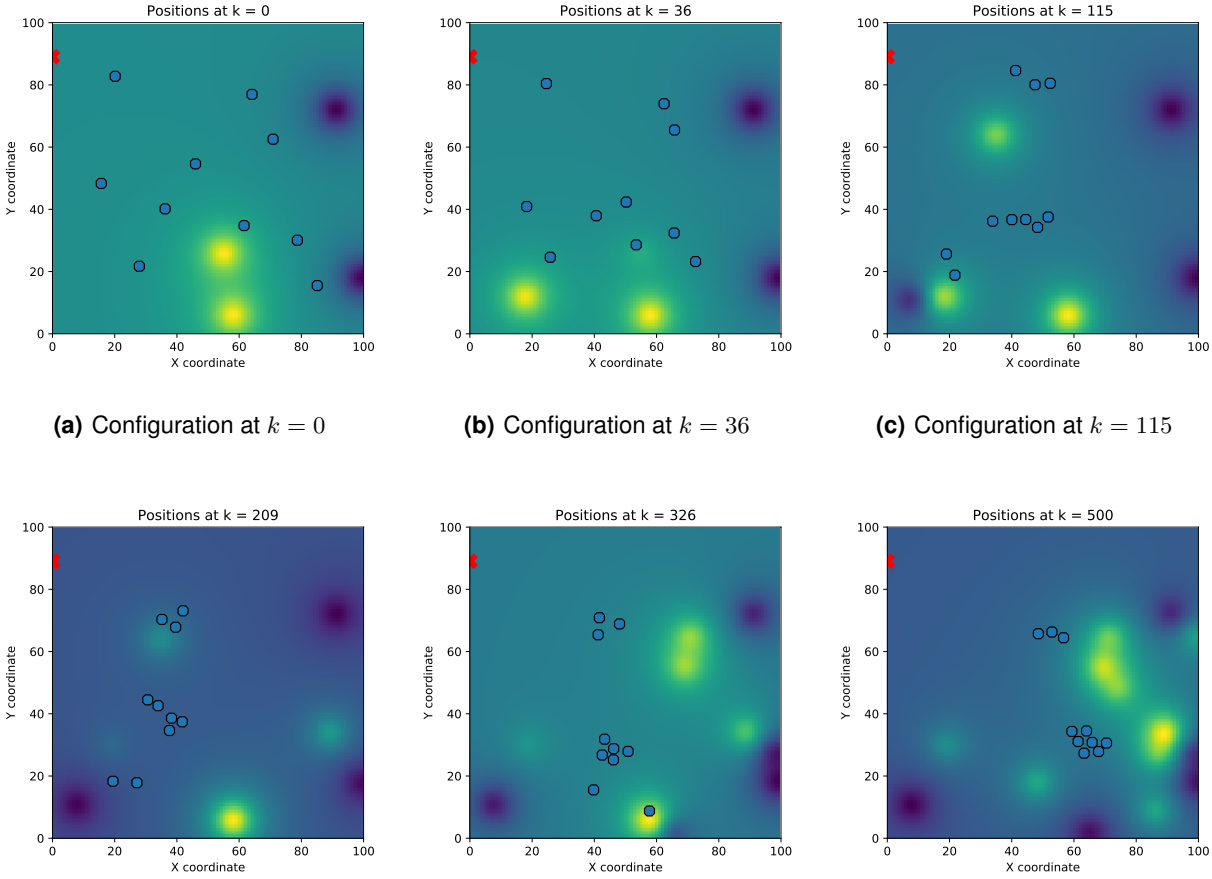
(a) Configuration at $k = 0$ **(b)** Configuration at $k = 140$ **(c)** Configuration at $k = 230$

(d) Configuration at $k = 386$ **(e)** Configuration at $k = 455$ **(f)** Configuration at $k = 500$

Figure 2.12: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values, multiple static minima, and unauthorized areas (black squares); with one communication tower (marked by the red X). Using *Algorithm 1*.

2.7.3.J Multiple Static Rendezvous Areas with Dynamic Values and Multiple Communication Towers

The simulation type 3e - multiple static rendezvous values with dynamic values, multiple static minima, and multiple communication towers - is illustrated in Figures 2.14(a) to 2.14(f). The simulation presented has two randomly statically-positioned towers, marked by the red Xs. This simulation demonstrates that the agents can receive information from multiple towers to achieve rendezvous to the multiple dynamic desired areas (objective 1). Objectives 2, 4, 5, 6, and 7 are also achieved. This scenario does not contain unauthorized areas, so objective 3 is invalid.



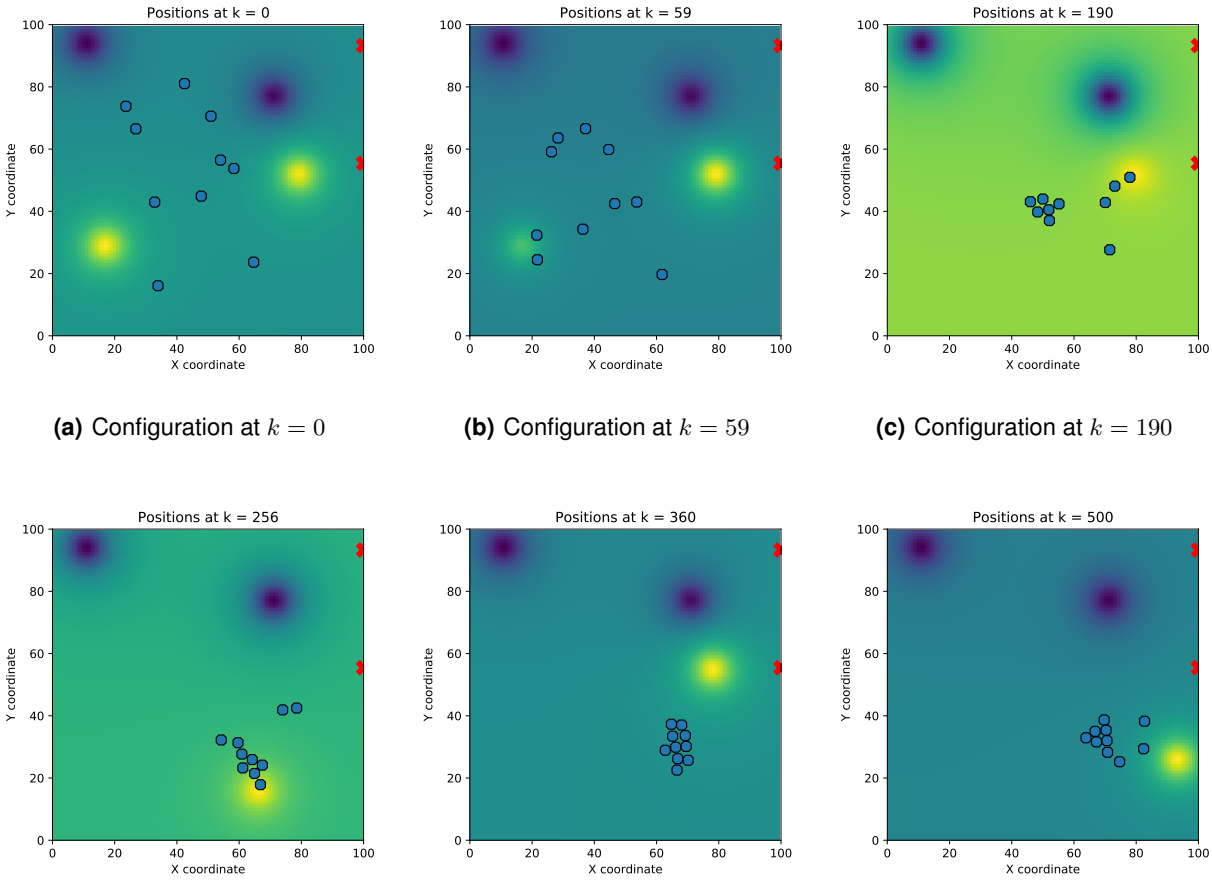
(a) Configuration at $k = 0$ **(b)** Configuration at $k = 36$ **(c)** Configuration at $k = 115$

(d) Configuration at $k = 209$ **(e)** Configuration at $k = 326$ **(f)** Configuration at $k = 500$

Figure 2.13: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values and multiple static minima; with one communication tower (marked by the red X). During the simulation, multiple rendezvous and undesired areas are randomly added or removed. Using *Algorithm 1*.

2.7.3.K Multiple Static Rendezvous Areas in a 200×200 Mission Plane with 30 Agents and 10 Communication Towers

Figures 2.15(a) and 2.15(b) illustrate type 3g - multiple static rendezvous areas in a 200×200 mission plane with 30 agents and 10 communication towers (randomly statically-positioned). This simulation demonstrates the agents can rendezvous to the multiple desired areas in an extensive mission plane with a large number of agents, initially forming a disconnected network topology (objective 1). All valid objectives for this scenario were achieved: 1, 2, 4, 5, 6, and 7. Due to the increased size of the mission plane and increased number of agents, the requirements to verify if the agents have completed their objective (and for the simulation to be considered finalized) have been modified: the maximum distance the agents can be from a rendezvous area has been increased to $26 \left(= \left\lceil \left\lceil \frac{30}{4} \right\rceil \cdot 1.6 \cdot 2 \right\rceil \right)$ distance units,



(a) Configuration at $k = 0$ **(b)** Configuration at $k = 59$ **(c)** Configuration at $k = 190$

(d) Configuration at $k = 256$ **(e)** Configuration at $k = 360$ **(f)** Configuration at $k = 500$

Figure 2.14: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values and multiple static minima; with two communication tower (marked by the two red X). Using *Algorithm 1*.

and the maximum sum of total distances between time steps $k-1$ and k (for two consecutive time steps) has been increased to $9\,(= 0.3 \cdot 30 \cdot 1)$.

2.7.4 Results Discussion

As presented in Section 2.7.1, the general design objectives to be tested by the simulations were the following:

1. agents move towards and remain in the rendezvous areas while these are in the utility function;
2. agents are repelled by the undesired areas;
3. agents do not enter unauthorized areas;

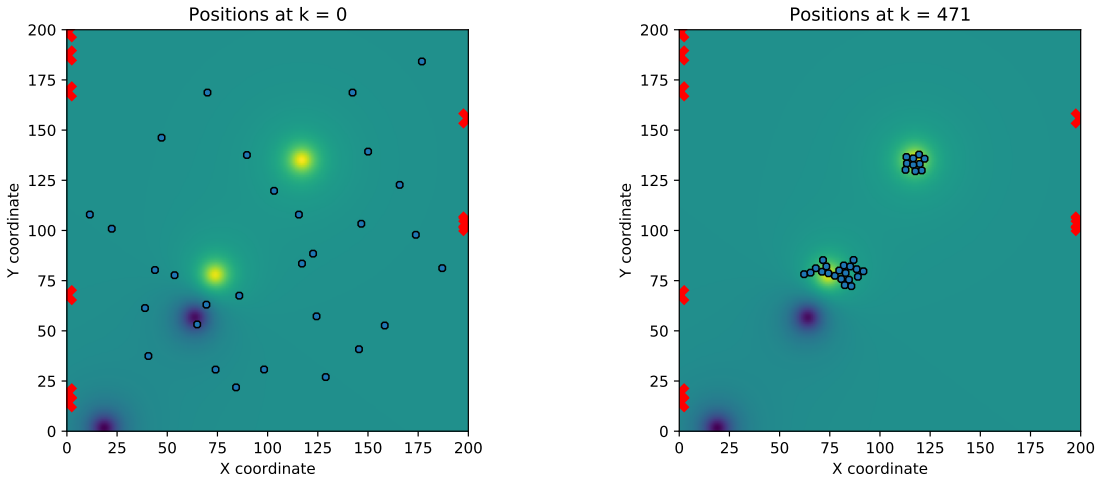


Figure 2.15: Agents configurations at time steps $k = 0$ (left) and $k = 471$ (right) for a simulation with 30 agents in a 200×200 mission plane with multiple static maxima and minima and with 10 communication towers (marked by the red Xs). Using *Algorithm 1*.

4. agents do not leave the mission plane;
5. agents do not collide;
6. indefinite movement deadlocks do not occur;
7. agents eventually leave low-utility areas;

Objectives 4, 5, 6, and 7 were verified across all simulation scenarios presented, and as such, are considered fully accomplished. Concerning Objective 6, as explained in Section 2.7.2, the final figures presented for each simulation might erroneously imply the agents are in a deadlocked state or in irrelevant positions for the mission objective. The simulation videos show the agents are not in a deadlocked state and that the positions of the agents were generally relevant.

The primary priority of the simulations presented in Sections 2.7.3.C and 2.7.3.D was to test objective 2 - testing if the agents are repelled from statically-positioned and moving undesired areas. From the simulation figures, it can be observed that this objective is fully accomplished.

Similarly, the simulations in Sections 2.7.3.F and 2.7.3.H were fundamentally used to verify the agents' collision avoidance of static environmental obstacles. As explained in Section 2.5.5.B, the environmental collision avoidance method implementation does not provide exact results: the agent calculates there will be no collision (with an environmental obstacle) if the future position of its polytope's centroid will be outside of any unauthorized area. This can result in the agent's polytope intersecting the area, as observed in Figure 2.12(f) (the lower right agents intersect the rectangle-shaped unauthorized area). Consequently, objective 3 is considered partially accomplished. This precision problem could

be solved by implementing a detection method based on the method used to detect collisions between agents: ray casting between the agent's polytope and the polytopes of the obstacles in the agent's area.

The primary objective of achieving rendezvous to desirable areas was a focus on all simulations. With different levels of success, the agents always reached rendezvous areas. In the simulations with a single rendezvous area (Sections 2.7.3.A and 2.7.3.B), all the agents achieved consensus on the rendezvous area. In the simulations with a single undesired area (2.7.3.C and 2.7.3.D), the agents avoided the undesired area, remaining in the positions with the highest utility value in the scenario. In simulations with multiple utility function maxima and minima (2.7.3.E to 2.7.3.K), the agents reached the rendezvous areas in the presence of environmental obstacles, random area destruction, and area removal due to agent exploration. Consequently, objective 1 is considered fully accomplished.

The simulations presented did not test how the system reacts to agents arbitrarily leaving and entering the network, because it is hypothesized the agent behavior would be identical to when agents discover new neighbors or lose connection to existing ones. This hypothesis results from the implemented local-agent approach: an agent dynamically adapts to the number of neighbors. The execution of simulations that test this behavior is left for future work.

2.8 Conclusions

This Chapter addressed the problem of having a group of mobile agents rendezvousing to multiple dynamic desired areas with a semi-decentralized approach. The number of existing rendezvous areas and their positions are unknown to all system entities: rendezvous areas are arbitrarily created and removed from the mission plane and are considered dynamic because their positions and utility values (interest to the mission objective) are time-varying. Agents have no localization sensors but are equipped with limited communication capabilities, receiving noisy positioning data from communication towers through directional broadcasts.

The proposed solution avoids the need for an initially connected topology, synchronous communication, and leader selection protocols by defining agent movement based on improved flocking rules that dynamically merge previously unknown agents in a single cluster while attracting the agents to areas with increasing interest to the mission objective - by considering the direction that maximizes the local gradient of the mission plane's utility function on each agent's position and by creating a simulated attraction force towards agents in positions with higher utility values.

Additionally, the system provides guarantees for an environment without agent-agent collisions, while in the presence of noisy positioning data sent by the measurement/communication towers: the position measurement of an agent is considered to be a worst-case-estimate polytope, containing all the possible positions in which the agent can be positioned.

The proposed solution was designed to be deployable to networks of large size, resulting in no computational limit on the number of allowed agents in the network or a limit to the mission plane size - although the convergence efficiency will generally decrease if the size of the mission plane is larger than the range of the communication towers. Secondary objectives of the system development considered power and cost efficiency: by offloading the localization sensors and only requiring equipment to receive communications, the agents become more inexpensive and conserve more power, which can result in increased battery autonomy or an inferior cost if the agents are equipped with smaller batteries.

The efficacy of the algorithm is illustrated using various simulated scenarios with sparse agent occupation. It is shown that agents generally achieve rendezvous while avoiding undesired and unauthorized areas.

Unlike the literature examples presented in Section 2.2, *Algorithm 1* rendezvous the mobile agents to multiple rendezvous areas, with unknown positions and movement dynamics, that represent interesting positions to the mission objective while not requiring apriori information regarding the number and position of the rendezvous areas, an initially connected network topology, apriori knowledge of the number and position of agents, leader/follower protocols, or direct agent-to-agent communication.

2.8.1 System Limitations

As a result of the agents not having localization sensors, they require the communication towers' broadcasts to obtain the positioning data regarding their neighbors. Due to all neighbor agents receiving the same broadcast, the positions received (and the velocities) are in a frame of reference that must be known by all the agents. As such, the agents need to have identical coordinate systems, which can be assumed to be given to an agent before it enters the system, in an initial setup step - in addition to the mission plane's boundaries and static environments' polytopes.

As discussed in Section 2.7.4, the agents can partially intersect the unauthorized areas, which simulate the environmental obstacles. This implementation limitation can be solved with the solution presented in Section 2.8.2.

2.8.2 Future Work

Objective 3 is considered to be only partially accomplished. As such, it is left for future work the implementation of an environmental obstacle collision detection method based on the method used to detect collisions between agents - ray casting between an agent's polytope and the polytopes of the obstacles in the agent's vicinity.

To improve the system's convergence efficiency, it is left for future work the development of an obstacle collision avoidance method based on circumventing obstacles and a heuristic for the decision of

target agent by the communication towers. The addition of agent movement without being triggered by the towers' broadcasts would also increase the convergence efficiency by having the agents move without receiving new information. The implementation would be based on polytope propagation: each agent propagates the polytopes of its neighbors to consider the positions to which they could have moved since the last broadcast (acknowledging the number of discrete time steps elapsed and their last known positions and velocities) and moves accordingly, with the agent collision avoidance method considering the propagated polytopes to create a fully collision-free environment.

For algorithmic performance improvements, it is left for future work the development of a communication protocol between the towers to mitigate redundant or duplicate broadcasts and a vertex culling method in the ray tracing process. Redundant or duplicate broadcasts can occur when two communication towers are close together and target an agent in adjacent time steps. The simulation presented in Section 2.7.3.K illustrates the situation of having multiple towers close together: in Figure 2.15(a), the right side of the mission plane's perimeter has four communication towers, with three being within approximately 10 distance units. The improved ray-tracing method would only trace rays from the relevant vertices for the collisions, i.e., vertices associated with edges whose normal vectors have a non-negative projection on the desired movement vector. Informally, if an edge is on the back of an agent's polytope, and the agent's desired direction is forwards, the vertices in the back edge are not considered. Figure 2.1(b) illustrates the result of applying back-face culling to the current implementation.

3

Rendezvous with Agents with Localization Capabilities

Contents

3.1 Introduction

As mentioned in Chapter 2, there exist centralized systems to achieve rendezvous in the literature - for instance, with the use of leader-selection protocols. *Algorithm 1* improves these systems by being partially-decentralized with milder assumptions on the known information than other proposals. Herein, we propose a fully-decentralized solution - *Algorithm 2* - to the rendezvous problem by building on the partially-decentralized solution. By decentralized, it is meant that all nodes do not require any communication capabilities and resort to local sensors to find their position and velocity. Such a proposal is ideal for communication-denied environments.

Algorithm 2 extends the algorithm presented in Chapter 2, solving the rendezvous problem considering agents with localization sensors and no communication capabilities. In order to avoid structured solutions, it is assumed identical agents executing a modified version of *Algorithm 1* that is both decentralized and allows for the additional objective of moving in dynamic formations. Thus, there is no need for leader selection protocols, specialized message transmissions, and single points of failure.

The envisioned scenario does not include any centralized structure (the communication towers) and agents are equipped with localization and sensing equipment that acquire position and velocity measurements of agents within a sensing radius. We remark that given the localized decision based on relative measurements, there is no need for a global coordinate system. These sensors are corrupted by noise and agents are equipped with observers that are able to construct worst-case estimates of the positioning data for each of their neighbors in the form of a convex polytope containing the exact unknown position, which, in the general case, will not match its centroid. We still require the solution to satisfy the following design objectives:

- no single point-of-failure;
- no maximum allowed number of agents due to computational limitations - to deploy in networks of unbounded size;
- computational and power efficiency - to increase the battery autonomy of the mobile agents.

The contributions presented in this Chapter are in review in [22].

3.2 Previous Work Review

This section presents literature examples regarding the formation component, which are sorted into three categories: Physical-Leader Formations (Section 3.2.1), Virtual-Leader Formations (Section 3.2.2), and Distance-Based Formations (Section 3.2.3). The requirement for a physical leader means that some entity must be elected to serve that role, be it an agent or a target. The alternative, a virtual leader, often

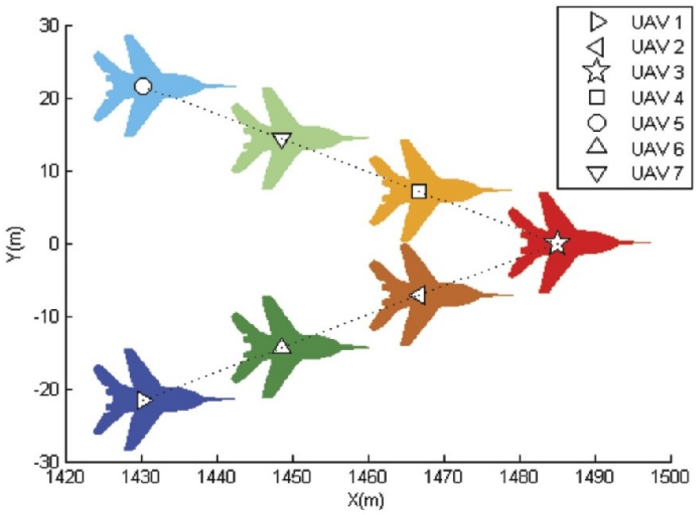


Figure 3.1: Physical-leader formation structure used in [1].

forces the dynamics of the leader to be known in advance or communicated to nearby agents that have the purpose of tracking it.

3.2.1 Physical-Leader Formations

In [8], the problem of trajectory tracking in a desired formation is addressed. The position of the target is assumed to be known by the agents and serves as a reference point from which the formation slots are constructed. These correspond to the vertices of a regular n-sided polygon centered at a relative position from the target position, with the identifier of each node being used for the slot assignment. The target agent in [8] is a physical agent, but it does not follow the same control rule as the agents in formation, and instead moves with predetermined dynamics.

In [1], the authors tackle this problem using the concept employed in pigeon flocks. The algorithm transforms an arbitrary connected topology into a leadership hierarchy that defines the immediate individual leaders of each agent. The primary leader has two direct followers, while the intermediary agents have a single immediate neighbor up and down the hierarchy, which is illustrated in Figure 3.1. Communication exists to have nodes achieve consensus regarding the structure. A critical issue with the application of such a technique to the considered scenario is the fact that agents i and j can have two different neighbor sets, $\mathcal{N}_i \setminus \mathcal{N}_j \neq \emptyset$. The procedure used in [1] to deal with nodes leaving the network requires that an agent starts following its previous leader's leader in case the immediate connection is lost. The decentralization objective precludes the adoption of such a strategy.

Moreover, the algorithm in [1] also has some shortcomings in our scenario:

- low fault tolerance since agents are only connected to their immediate leader, posing problems

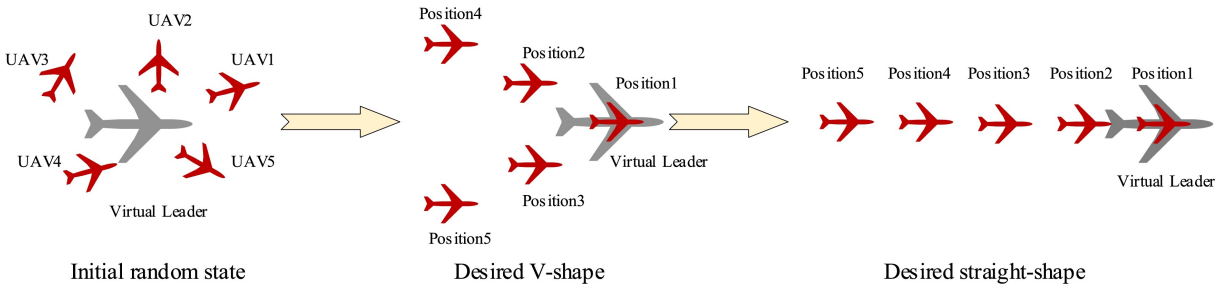


Figure 3.2: Virtual-leader formation structures used in [2].

whenever this structure changes;

- limited scalability due to the need for a consensus protocol with all agents in the network to determine the leadership hierarchy tree;

- the selection of the control parameters is manual.

3.2.2 Virtual-Leader Formations

In contrast with the examples in Section 3.2.1, [2] uses a virtual leader to address this problem. Instead of having a single physical agent controlling the formation, each agent has a virtual representation of a leader with a known desired path and velocity. This position is used as a reference point that allows each agent to find its relative location within the formation structure, which is illustrated in Figure 3.2.

The movement in [2] uses a control law with components that can be interpreted as the flocking rules of *Separation*, *Cohesion*, and *Alignment*, and a collision avoidance method based on potential fields. The actuation is therefore given by:

$$u_i = u_\gamma + u_{\alpha i} + u_{\beta i} + u_d \tag{3.1}$$

where u_γ is the acceleration of the virtual leader, $u_{\alpha i}$ and $u_{\beta i}$ are the components for the attraction/repulsion between agents and from objects, respectively, and u_d is the formation maintenance control law. The value of $u_{\alpha i}$ is given by:

$$u_{\alpha i} = -c_\alpha \sum_{j \in \mathcal{N}_{\alpha i}} \nabla_{x_i} \psi_\alpha(\|x_j - x_i\|_\sigma) \tag{3.2}$$

where c_α represents the weight coefficient of the force between agents, $\mathcal{N}_{\alpha i}$ is the neighbor set of agent i, and ψ_α simulates the attraction/repulsion forces between agents. The component $u_{\alpha i}$ equates to the *Separation* and *Cohesion* components of *Flocking*, presented in [5]. The value of $u_{\beta i}$ is defined as:

$$u_{\beta i} = -c_\beta \sum_{k \in \mathcal{N}_{\beta i}} \nabla_{x_i} \psi_\beta(\|x_{ok} - x_i - r_{ok}\|_\sigma) \tag{3.3}$$

where c_β represents the weight coefficient of the force between an agent and an obstacle, $\mathcal{N}_{\beta i}$ is a collection of obstacles in the vicinity of i, and ψ_β simulates the repulsion force from an obstacle. Lastly, the value of u_d is:

$$u_d = -k_1 \sum_{j \in \mathcal{N}_{\alpha i}} (x_i(t) - x_j(t) - r_{ij}) - k_2(x_i(t) - x_\gamma(t) - r_i) - k_3 \sum_{j \in \mathcal{N}_{\alpha i}} b_{ij}(t)(v_i(t) - v_j(t)) - k_4(v_i(t) - v_\gamma(t)) \tag{3.4}$$

where k_1, k_2, k_3, and k_4 are component weights. The components of k_3 and k_4 are responsible for the consensus on the velocity of the formation (considering the velocity of the virtual leader) and are equivalent to the *Alignment* component in *Flocking*. The components of k_1 and k_2 are responsible for moving an agent to its formation slot.

Similarly to [1], the consensus for slot assignment would not always provide desirable results in our proposed scenario, because any two neighbor agents i and j can have different neighbor sets, $\mathcal{N}_i \setminus \mathcal{N}_j \neq \emptyset$.

3.2.3 Distance-Based Formations

In [9], it is addressed the problem of formation stabilization through the use of potential functions. The algorithm considers a graph $\mathcal{G} := (\mathcal{V}_\epsilon, \mathcal{C}, \mathcal{D})$ to represent the network, where $\mathcal{V}_\epsilon$ and $\mathcal{C}$ are the vertices and adjacency matrix, respectively, and $\mathcal{D}$ is the distance matrix, whose elements are the desired distances between each pair of agents. Achieving the assigned separation is done through the minimization of $\phi_l(q_i, q_j)$ given by:

$$\phi_l(q_i, q_j) := \|q_i - q_j\| - d_{ij} \tag{3.5}$$

where d_{ij} is the element in $\mathcal{D}$ corresponding to the desired edge-length between agents i and j. The set of structure constraints is referred as $\Phi = 0$. The solution presented in [9] to achieve stable formations is to steer the agents in the direction that minimizes the error in the structural constraints. Our movement implementation has a similar flavor, although the formations in [9] are static and the single objective of an agent is to reach its slot. Directly adapting these strategies to a case where multiple objectives are present is not trivial. A related work in [10] also defines a fixed formation with the desired velocity of the centroid being available, which results in a leader/follower protocol with a single point-of-failure.

3.2.4 Previous Work Summary

The design choices and implementations in the literature examples presented in Sections 3.2.1 to 3.2.3 result in the following limitations:

1. The formation position is not dynamic, ([9]);
2. The agents need to know the position of the formation or a reference point, ([1], [2], [8], [9]);
3. The agents need to know the identity of their neighbors, ([1], [9], [10]);
4. The formation slots are pre-assigned, ([8], [9], [10]);
5. The formation is locked to new agents, ([1], [2], [8], [9], [10]);
6. The formation does not adapt to the loss of agents, ([8], [9], [10]).

Cases 1 and 2 are not desirable in our system since nodes need to move to rendezvous areas and, as such, the position of the structural representation of the formation must be dynamic. Moreover, the structure must be decided by each agent individually, as having a central node (such as the towers in *Algorithm 1*) transmitting it would result in a partially-decentralized approach.

Cases 3 to 6 do not comply with our assumptions of identical agents and dynamic formations: agents have no identity, and the formation structure is altered based on the number of nearby agents. The implemented solution considers the structure as a convex regular m-sided polygon (based on [8]) centered on the average position of the m agents participating in the formation - this center can be seen as the virtual leader in [2] - with each vertex being a slot assigned greedily in real-time.

The formation implementation and respective movement component are described in Sections 3.5.3 and 3.5.2, respectively.

3.3 Contributions

The solution proposed in this Chapter presents a novel method to achieve rendezvous to multiple dynamic desirable areas while the agents create and maintain non-rigid formations. These agents are aware of their position and can localize neighbor agents, only having a localized view of the network. The contributions presented in Chapter 2 remain valid: there is no apriori information regarding the rendezvous areas, there are no assumptions regarding the initial network topology, the position and velocity measurements are assumed to be corrupted by noise, and the collision avoidance method implemented provides guarantees for an environment without agent-agent collisions.

Unlike the literature examples presented in Sections 2.2 and 3.2, the proposed solution rendezvous the agents without requiring apriori knowledge of the number and position of agents, leader/follower

protocols, or communication between system entities. Moreover, it is resilient to faults by being fully-decentralized. Additionally, the implementation of this algorithm focuses on the following objectives:

1. No single point-of-failure. The system will only fail if all the agents experience failure.

2. Deploying to networks of unbounded size. There is no computational limit on the number of allowed agents in the network or a limit to the mission plane size - although the convergence efficiency will generally decrease if the sensing radii are not increased with the size of the mission plane. This is possible because the control law only uses local information in the vicinity of the nodes provided by the on-board sensors.

3. Computational and power efficiency.

The contributions presented in this Chapter are in review in [22].

3.4 Problem Statement

The algorithm presented in this Chapter is an extension of *Algorithm 1*, which includes the system components and assumptions unless explicitly stated otherwise. Due to the agents having additional sensor capabilities, the communication towers have been removed.

The agents are assumed to have on-board sensors that measure noisy positions and velocities of nearby agents. These are assumed to either be acquired by a global system, such as GPS, or by relative measurements. These also enable nodes to determine, once entering the network, if the mission plane boundaries are near and to find convex polytopes containing any environmental obstacle in the agent's local coordinate system.

For simplicity, the sensing region is assumed to be a circle with radius SR. Comparable to the communication towers, the position measurements taken by the agents are noisy and passed through an observer that returns convex polytopes that are guaranteed to contain the exact unknown positions of its neighbors. In order to navigate to areas of interest, agents can measure the utility function in this sensing radius. The definition of the sensing area of an agent and resulting neighbors is illustrated in Figure 3.3.

3.5 Proposed Solution

All implementation decisions and design goals not explicitly redefined in this section are sustained from the previous algorithm.

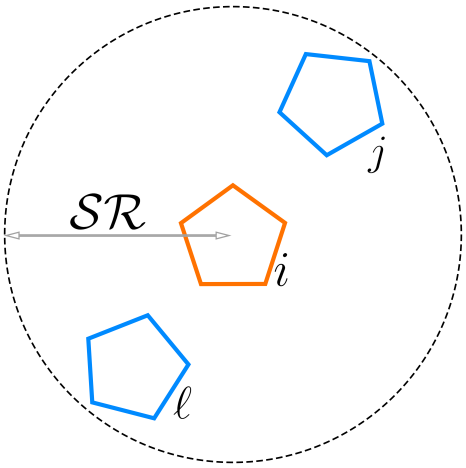


Figure 3.3: A not-at-scale illustration of the sensing radius, SR and the area where agent i can sense neighbors j and ℓ positions and velocities, as well as the utility values.

3.5.1 Neighbors Definition

The redefinition of neighbor agents is based on the sensing radius. Agent i's neighbors are all agents within SR distance units. Formally, the neighbor set is redefined as:

$$\mathcal{N}_i(k) = \{j \in \mathcal{V} : \parallel q_i(k) - q_j(k) \parallel < SR\} \tag{3.6}$$

3.5.2 Individual Agent Model

The movement model is a direct extension of the model presented in Section 2.5.3, with the addition of a component referent to agent formations.

Formation Component The *Formation* component is responsible to move the agents to their formation slots. Formally, $u_i^f(k)$ is defined as:

$$u_i^f(k) = f_i(k) - c_i(k) \tag{3.7}$$

where $f_i(k)$ is the position of agent i's slot in the formation. Section 3.5.3 describes the process of calculating $f_i(k)$.

3.5.2.A Desired Movement Law

With the addition of $u_i^f(k)$, the desired movement law is redefined as:

$$\hat{u}_i(k) = \theta \cdot u_i^s(k) + \beta \cdot u_i^c(k) + \gamma \cdot u_i^a(k) + \delta \cdot u_i^{attr}(k) + \epsilon \cdot u_i^u(k) + \zeta \cdot u_i^f(k) + \eta \cdot u_i^r(k) \tag{3.8}$$

where all weights $\theta, \beta, \gamma, \delta, \epsilon, \zeta, \eta \in \mathbb{R}^+$, with all component vectors normalized.

3.5.3 Formation Assembly

An agent formation acquires the form of a regular m-sided polygon, which is centered in the average position of the participating m agents.

3.5.3.A Formation Creation

Each agent creates a virtual structure representing the desired regular m-sided polygon based on its neighbors. Due to the absence of strips, neighbor agents do not necessarily have an identical neighbor set, which can cause the virtual formation structure of two neighbor agents to be different. This problem is mitigated by the *Cohesion* component that decreases the probability of a cluster of agents having differing neighbors sets by decreasing the overall distance between agents in the cluster. An internal formation structure representation is illustrated in Figure 3.4. Agent i has three neighbors, so its virtual structure is a (rotated) square.

3.5.3.B Slot Assignment

The formation slots correspond to the positions of the vertices of the m-sided polygon structuring the formation. The slots are assigned by each agent independently, by attributing each slot to the agent that is closest to it. This heuristic was chosen due to its simplicity and non-conflicting properties - it does not cause double-booking of slots. It is left for future work the implementation of a distributed heuristic that minimizes the global cost of the agents moving to their assigned slots, instead of the current greedy approach. Figure 3.4 shows the slots assigned to each agent (black arrows pointing to the vertices) based on this heuristic.

3.5.4 Final Control Law

The definition for the final control law remains unchanged:

$$u_i(k) = \frac{\hat{u}_i(k)}{\|\hat{u}_i(k)\|} \cdot \min(D, D_{env}) \tag{3.9}$$

with $\hat{u}_i(k)$, D, and D_{env} being defined in Equation (3.8) and Sections 2.5.4.B and 2.5.5.B, respectively.

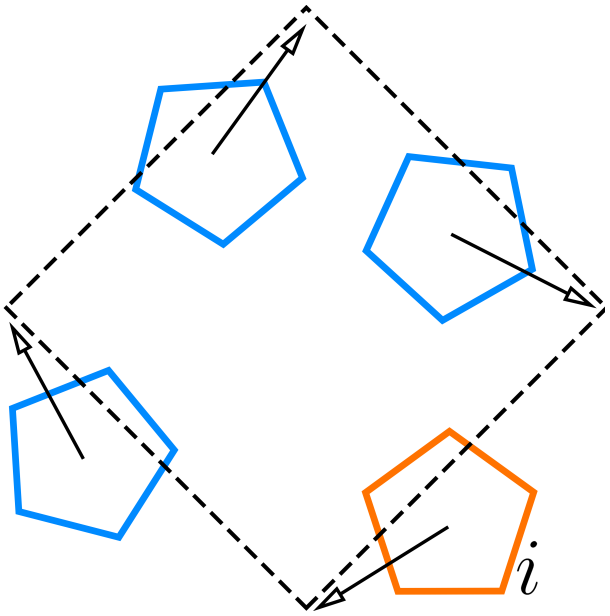


Figure 3.4: Formation structure, as internally represented by agent i. i has three neighbors, so its virtual formation structure representation is a (rotated) square. The black arrows represent the slot assigned for each agent. Due to the heuristic used, this assignment will be identical in the internal representations of all agents in the formation. Not at scale.

3.6 Convergence Analysis

In the previous section, we have constructed the actuation law as a sequence of terms that should guide each of the agents towards one of the objectives. These rules were inspired by the traditional mechanisms that are used to deterministically simulate flocks. In this section, we aim to provide a convergence analysis by viewing the overall system as a consensus algorithm driven by a signal that corresponds to a noisy gradient and a noisy desired velocity term to achieve the formation.

Theorem 2. *Consider a network of n agents running the algorithm given in 3.9 for sufficiently small $\theta, \beta, \delta, \eta \in \mathbb{R}^+$ and positive constants γ, ϵ, ζ. Then, for each agent i, it will be satisfied one of the following:*

i) $\lim_{k\to\infty} \|c_i(k) - f_i(k)\| \leq \varphi_1(\mathrm{p}, \xi)$*, where $\varphi_1(\mathrm{p}, \xi)$ is some constant dependent on the whole vector of parameters p and the noise ξ associated with the set-valued estimates;*

ii) $\lim_{k\to\infty} \|x_i(k) - x^\star\| \leq \varphi_2(\mathrm{p}, \xi)$*, where $\varphi_2(\mathrm{p}, \xi)$ is some constant dependent on the whole vector of parameters p and the noise ξ associated with the set-valued estimates, and $x^\star = _{x\in\mathcal{C}}\ h(x)$ for some closed set $\mathcal{C}$ such that $x^\star$ does not belong to the boundary of $\mathcal{C}$.*

Proof. Given that the statement of the theorem fixes a value for n, we can address the question with a single cluster of connected nodes and the same asymptotical results will hold when new nodes of a

different cluster come in contact with the current one. Without the collision avoidance mechanism, the velocity vector of each node in the cluster is subject to the following dynamics:

$$\begin{aligned} v_i(k+1) &= v_i(k) + \gamma \frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} v_j(k) - v_i(k) + \epsilon \nabla h(c_i(k)) + \zeta \left(f_i(k) - c_i(k)\right) + \sigma_i^{\mathrm{p}}(k) \\ &= (1-\gamma) v_i(k) + \gamma \frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} v_j(k) + \epsilon \nabla h(c_i(k)) + \zeta \left(f_i(k) - c_i(k)\right) + \sigma_i^{\mathrm{p}}(k) \end{aligned} \tag{3.10}$$

where $\sigma_i^{\mathrm{p}}(k)$ can be seen as a perturbation associated with the remaining components and such that $\|\sigma_i^{\mathrm{p}}(k)\|$ is sufficiently small and dependent on the vector of parameters p that contains θ, β, $\cdots$. We can further define $v_i^f(c_i(k)) := f_i(k) - c_i(k)$ since this is essentially a velocity vector driving the agent from $c_i(k)$ towards its intended position in the formation $f_i(k)$. Thus, we can further simplify the expression as:

$$v_i(k+1) = (1-\gamma) v_i(k) + \gamma \frac{1}{|\mathcal{N}_i(k)|} \sum_{j \in \mathcal{N}_i(k)} v_j(k) + \epsilon \nabla h(c_i(k)) + \zeta v_i^f(c_i(k)) + \sigma_i^{\mathrm{p}}(k) \tag{3.11}$$

Writing the above expression in vector form returns:

$$v(k+1) = W_\gamma v(k) + \epsilon \nabla \mathbf{h}(c(k)) + \zeta v^f(c(k)) + \sigma^{\mathrm{p}}(k) \tag{3.12}$$

where $\mathbf{h}$ is a vector-valued function receiving all polytope centers and returning the stack of all evaluations of h on each center. Since the support graph associated with W_γ is the complete graph and all entries are strictly positive, by the Perron-Frobenius theorem, the velocity vector with the gradient input, the formation velocity and the noise converges to consensus.

One can replace $c(k)$ by the true positions $q(k)$ by adding a second perturbation term whose norm is solely dependent on the maximum estimation error ξ and the Lipschitz constant of h in the following manner:

$$v(k+1) = W_\gamma v(k) + \epsilon \nabla \mathbf{h}(q(k)) + \zeta v^f(q(k)) + \sigma^{\mathrm{p}}(k) + \sigma^{\xi}(k). \tag{3.13}$$

The above consensus dynamics will converge to a cluster velocity vector driving the cluster positions to points that have zero input signals of the gradient and $v^f(q(k))$. Let us first consider the case where the cluster is in a position where the term $\nabla \mathbf{h}(q(k))$ is negligible since the function $\mathbf{h}$ is not assumed to be strictly convex. Then, the consensus dynamics will converge to a neighborhood of size dependent on the norm of the noise signals $\sigma^{\mathrm{p}}(k) + \sigma^{\xi}(k)$ around a point such that $v^f(q(k))$ is zero, and the conclusion i) follows.

If on the other hand, the signal $v^f(q(k))$ is negligible in comparison with $\nabla \mathbf{h}(q(k))$, the consensus dynamics converges to a neighborhood around a local maximum as seen in [23], and ii) conclusion follows. □

We remark to the reader that the results in Theorem 2 seem to conflict with some of the evidence in the simulations section. The value of $\delta = 0.08$ seems to point towards the case examined in Theorem 2. However, given that the range of h typically will be $[0, 2000]$ for the simulated cases, even a $\delta = 0.08$ is not sufficiently small to make the term $\delta \cdot u_i^{attr}(k)$ serve as a perturbation. This was done by design since the intended behavior is to have nodes move in loosely coupled formation when exploring the mission plane and then converge to a rendezvous point of largest utility, i.e., a single location that is a local maximum of h. The term $u_i^{attr}(k)$ when added in the velocity consensus dynamics can be seen as a gradient-free optimization and similar results are applicable to those ii) in Theorem 2.

3.7 Simulation Results

To evaluate this algorithm, we executed multiple simulations across a variety of environments. Sections 3.7.1 and 3.7.2 present the common goals and environments over all the simulations, with 3.7.3 providing an analytical analysis of the results, and 3.7.4 comparing them to the desired outcome. The simulation figures and videos are available on GitHub [21].

3.7.1 Simulation Goals

Identically to the simulations in Chapter 2 (Section 2.7.1), the simulations are used to verify the following design objectives:

1. agents move towards and remain in the rendezvous areas as long as the areas correspond to the maxima of the utility function;
2. agents are repelled by the undesired areas;
3. agents do not enter unauthorized areas;
4. agents do not leave the mission plane;
5. agents do not collide;
6. indefinite movement deadlocks do not occur;
7. agents eventually leave low-utility areas.

Additionally, due to the new components of this algorithm, the simulations will be used to verify the following additional objectives:

8. all agents move in every discrete time instant;

9. agents enter and remain in formation;

10. agents do not compete for a formation slot.

3.7.2 Simulation Environment

Every simulation presented in Section 3.7.3 utilizes identical agents, including the weights in the control law in (3.9): $\theta = 0.07, \beta = .01, \gamma = 0.6, \delta = 0.08, \epsilon = 1.0, \zeta = 1.0, \eta = 0.01$. Similar to the environment used for *Algorithm 1* in Section 2.7.3, the simulations generally use the default 100×100 mission plane size and 10 agents. There are no communication towers and the sensing radii is set to 15 distance units. These parameters are modified in specific simulations when explicitly stated.

As mentioned in Section 2.7.2, the simulations have a pre-imposed hard time limit (generally equivalent to 500 discrete time steps) to be able to be recorded and analyzed. The time limit can result in the configuration of agents presented in the final figure of a simulation to imply that some agents are in a deadlocked state or in an irrelevant position to the mission objective. This misinterpretation is primarily manifested in simulations with dynamic-valued rendezvous areas: the agents reach an area, explore it, the area is removed from the mission plane, and the final configuration illustrates the agents in an area with no rendezvous area. To mitigate this occurrence, each simulation is presented with multiple snapshots of configurations across the simulation duration. Additionally, cases with intermediate time steps that do not present useful information for the system analysis (for instance, a single rendezvous area whose purpose is to test if the agents reach the desired area from an initially disconnected network topology) are only displayed with the initial configuration (at $k = 0$) and the configuration at a time step in which the agents are considered to have fulfilled their objective. The rule requires two objectives:

- All agents are in a rendezvous area;
- The sum of the distances moved by all agents between time steps $k-1$ and k is below a threshold for two consecutive time steps.

The first requirement is satisfied when each of the agents is at most at 10 distance units from its closest rendezvous area. This value corresponds to:

$$d_{rendezvous} = \left\lceil \left\lceil \frac{n}{4} \right\rceil \cdot 1.6 \cdot 2 \right\rceil \tag{3.14}$$

where n is the number of agents (generally 10) and 1.6 is the radius of each agent polytope. The value $d_{rendezvous}$ is an upper bound on the maximum distance at which an agent can be from a rendezvous point if all n agents are equally divided in each of the four cardinal directions around a single rendezvous area. In the simulated environments where this maximum distance requirement is increased, it will be explicitly stated. The second requirement is satisfied when the sum of the distances moved by all the

agents between a time step $k-1$ and k is less than 3 distance units, corresponding to 30% of the sum of the maximum movement for all 10 agents. Whenever the setup includes a dynamic utility function, such criteria is never satisfied since the mission objectives is always changing.

3.7.3 Analytical Analysis

The scenarios included are:

1. Single-maximum utility function - to account for rendezvous missions:
   (a) Static rendezvous area;
   (b) Moving rendezvous area;
2. Single-minimum utility function - to illustrate escape missions:
   (a) Static minimum area;
   (b) Moving minimum area;
3. Multiple maxima and minima - to depict the cases of conflicting objectives:
   (a) Static rendezvous areas;
   (b) Static rendezvous areas in a mission plane with unauthorized zones;
   (c) Static rendezvous areas with dynamic values;
   (d) Static rendezvous areas with dynamic values in a mission plane with unauthorized zones;
   (e) Static rendezvous areas with dynamic values and random maxima/minima creation and destruction;
   (f) Static rendezvous areas in a 200×200 mission plane with 30 agents.

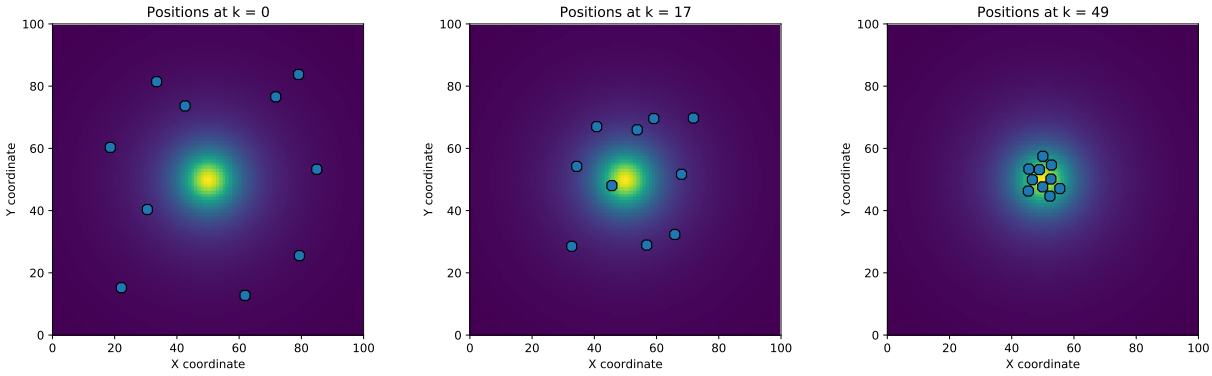
The *"static"* attribute is related to the time-invariant position; when the utility values change, it is explicitly stated with *"dynamic values"*.

In the cases of multiple maxima and minima - case 3 - the minimum areas are also static, and in 3e, they are also created and destroyed randomly. In order to represent the utility function, the simulation figures use a relative color scale: yellow represents positions with the highest utility value, while purple is assigned to the lowest values.

3.7.3.A Single Static Rendezvous Area

Figures 3.5(a) to 3.5(c) illustrate the simulation type 1a - single static rendezvous area. The rendezvous area is at the center of the mission plane (marked in yellow in the figures), and its position and utility

values are time-invariant. This first example demonstrates that convergence to a single cluster in the rendezvous area can be achieved from an initially disconnected network topology (objective 1). Formations are only briefly observed (objective 9) in this simulation type because the agents are promptly attracted to the rendezvous area, and their priority is to reach the high-utility area rather than create formations. This is demonstrated by the general absence of formations whenever the agents are at a rendezvous area (across all simulations). Pair-wise formations can be observed in Figure 3.5(b): the two pairs of agents at the top and the pair on the lower right. The simulation video shows that the agents in these pairs generally maintain their relative distances until they reach the rendezvous area. Objectives 4, 5, 6, and 8 are also realized and will generally not be explicitly stated in the remainder of this section because their fulfillment is constant across all presented simulations. Objectives 2 and 3 are invalid in this simulation type, and objectives 7 and 10 are not observable because the agents reach the objective too quickly.

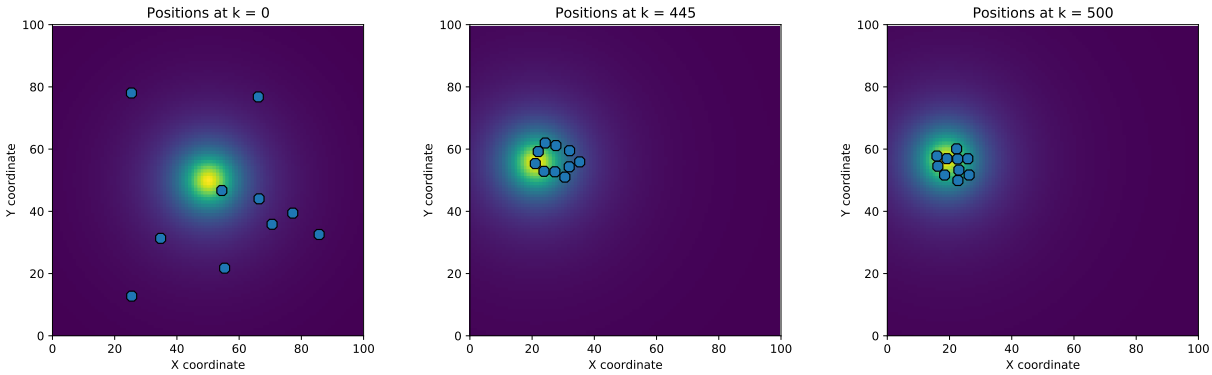


(a) Configuration at $k = 0$ **(b)** Configuration at $k = 17$ **(c)** Configuration at $k = 49$

Figure 3.5: Agents configurations for a simulation with a single static maximum at the center of the mission plane. Using *Algorithm 2*.

3.7.3.B Single Moving Rendezvous Area

Figures 3.6(a) to 3.6(c) illustrate type 1b - single moving rendezvous area (initially at the center of the mission plane). The agents track the moving desired area through the gradient of the utility function and rendezvous on the high-utility area (objective 1). Figure 3.6(b) shows the agents creating a formation (objectives 9 and 10).

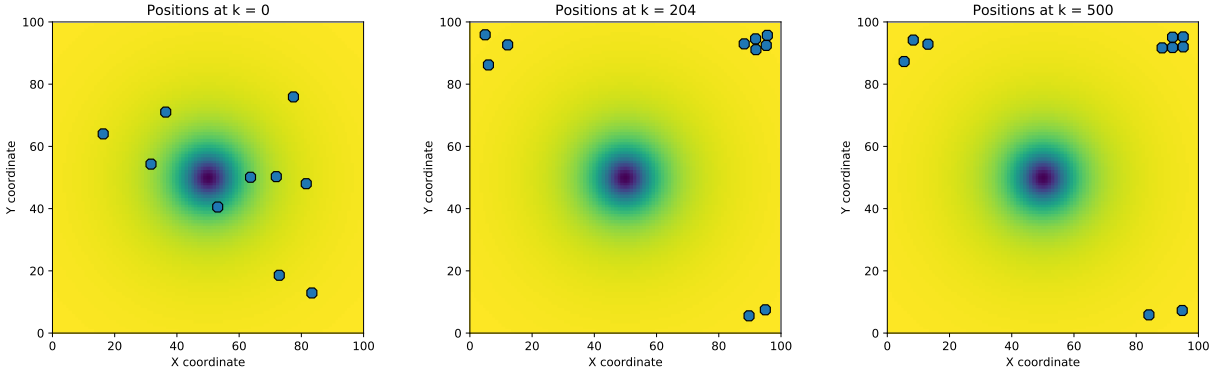


(a) Configuration at $k = 0$ **(b)** Configuration at $k = 445$ **(c)** Configuration at $k = 500$

Figure 3.6: Agents configurations for a simulation with a single moving maximum, initially at the center of the mission plane. Using *Algorithm 2*.

3.7.3.C Single Static Minimum Area

Simulation type 2a - single statically-positioned minimum (at the center of the mission plane) - is represented in Figures 3.7(a) to 3.7(c). This simulation type, and the simulation in Section 3.7.3.D, depict escape missions, which are considered accomplished, as the agents are repelled by the undesired area (objective 2). Due to the mission plane not having an explicitly defined rendezvous area, the agents do not form a single final cluster. Figure 3.7(b) shows the agents created formations: triangle on the top left and pair in the lower right (objectives 9 and 10).

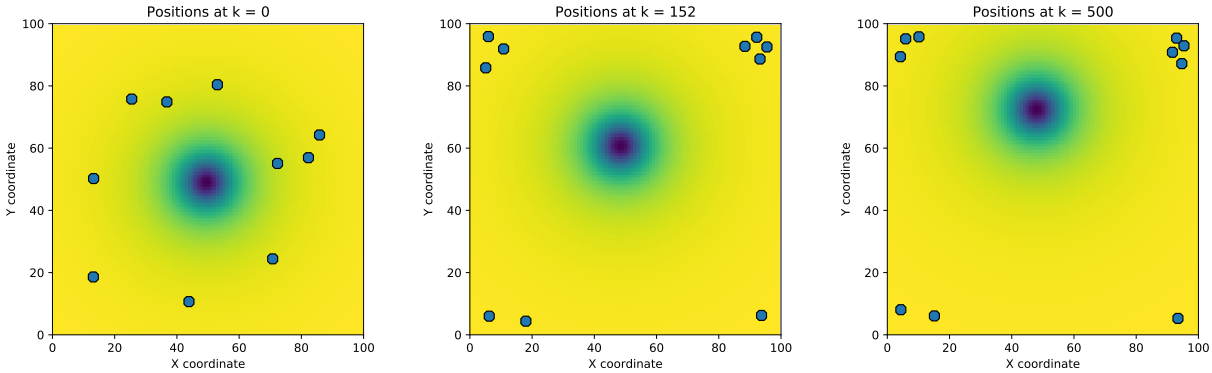


(a) Configuration at $k = 0$ **(b)** Configuration at $k = 204$ **(c)** Configuration at $k = 500$

Figure 3.7: Agents configurations for a simulation with a single static minimum at the center of the mission plane. Using *Algorithm 2*.

3.7.3.D Single Moving Minimum Area

Figures 3.8(a) to 3.8(c) illustrate simulation type 2b - single moving minimum area (initially at the center of the mission plane, moving randomly for the duration of the simulation). This simulation demonstrates that the agents avoid the undesirable moving area (objective 2) and create formations (objectives 9 and 10): triangle, square, and pair in Figure 3.8(b).



(a) Configuration at $k = 0$ **(b)** Configuration at $k = 152$ **(c)** Configuration at $k = 500$

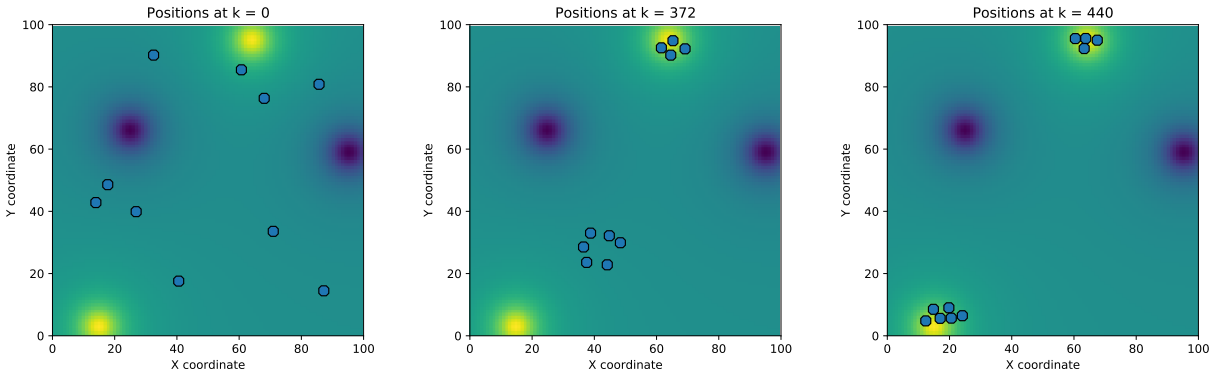
Figure 3.8: Agents configurations for a simulation with a single moving minimum, initially at the center of the mission plane. Using *Algorithm 2*.

3.7.3.E Multiple Static Rendezvous Areas

A mission plane with multiple static rendezvous areas - type 3a - is shown in Figures 3.9(a) to 3.9(c). The rendezvous areas and utility values do not change over time. The agents converge to the multiple rendezvous areas and avoid the low-utility areas (objectives 1 and 2). In Figure 3.9(b), the agents are finalizing a formation (objectives 9 and 10) while moving towards a rendezvous area.

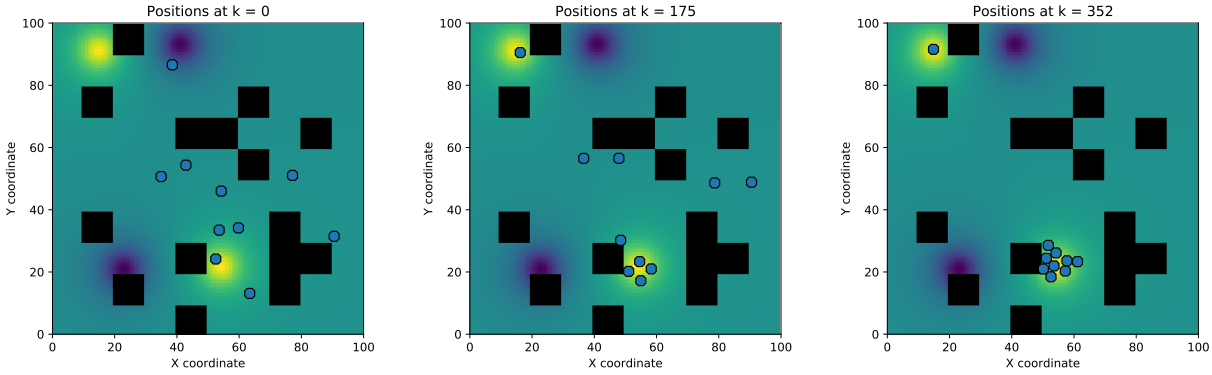
3.7.3.F Multiple Static Rendezvous Areas in a Mission Plane with Unauthorized Areas

Figures 3.10(a) to 3.10(c) illustrate type 3b - multiple static rendezvous areas with unauthorized areas. Similarly to 3a, the agents converge to the rendezvous areas while evading the low-utility areas (objectives 1 and 2). Additionally, the agents avoid the unauthorized areas represented by the black squares (objective 3). Figure 3.10(b) shows the agents in formation: 4 agents at the bottom forming a square, and two pairs of agents in the middle, each forming a line. It is not shown in the figures, but the paired agents maintain the relative position seen in Figure 3.10(b), which appears in the simulation video.



(a) Configuration at $k = 0$ **(b)** Configuration at $k = 372$ **(c)** Configuration at $k = 440$

Figure 3.9: Agents configurations for a simulation with a mission plane with multiple static maxima and minima. Using *Algorithm 2*.



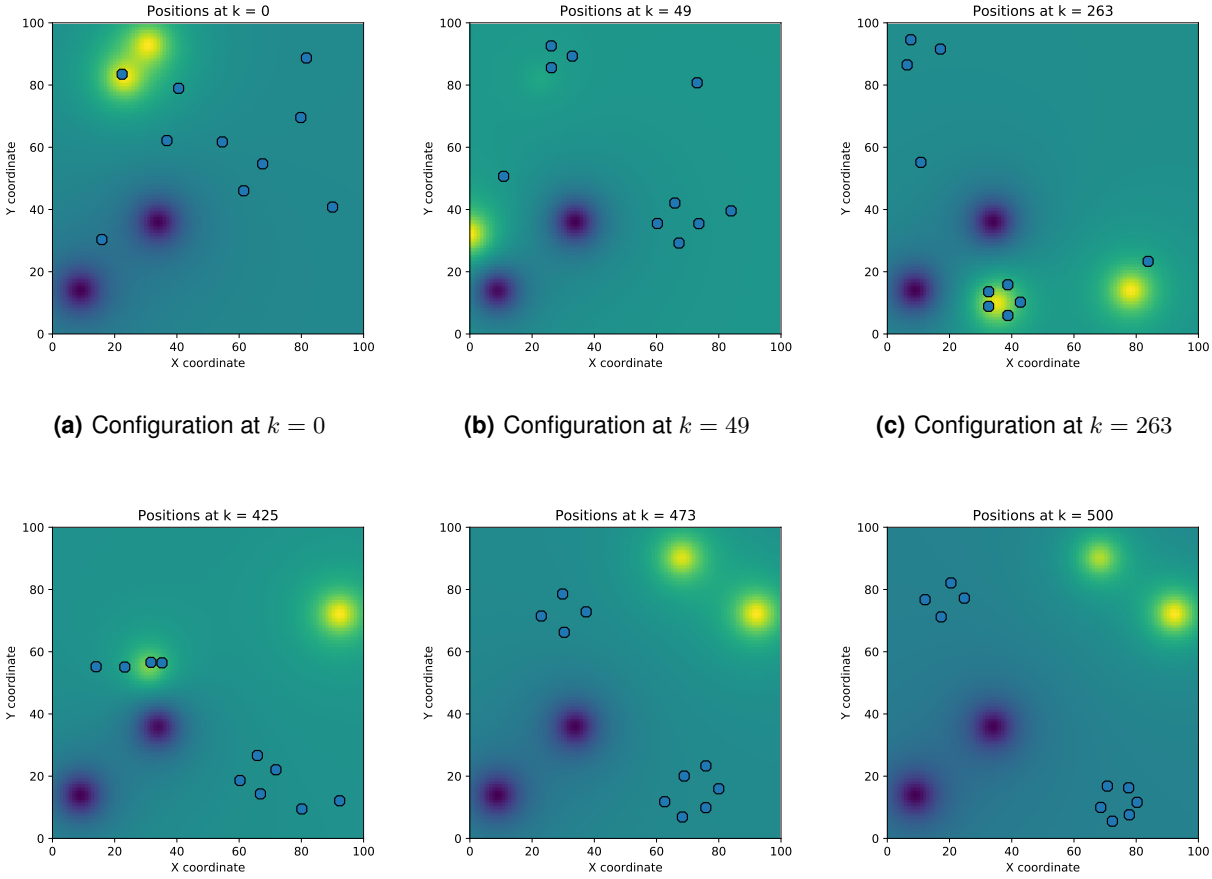
(a) Configuration at $k = 0$ **(b)** Configuration at $k = 175$ **(c)** Configuration at $k = 352$

Figure 3.10: Agents configurations for a simulation with a simulation with a mission plane with multiple static maxima, minima, and unauthorized areas (black squares). Using *Algorithm 2*.

3.7.3.G Multiple Static Rendezvous Areas with Dynamic Values

Type 3c - multiple static rendezvous areas with dynamic values - is illustrated in Figures 3.11(a) to 3.11(f). The utility values of the rendezvous areas change based on agent exploration, described in Section 2.7.3.G. The agents rendezvous to the multiple desired areas and, when they detect that the areas have been removed, they continue exploring the mission plane for new high-utility locations (objective 1). The cluster of five agents at the lower left in Figure 3.11(c) detects the rendezvous area has been removed (between 3.11(c) and 3.11(d)) and explores for a new one (3.11(d), 3.11(e), and 3.11(f)). Additionally, the agents avoid the low-utility areas (objective 2) and create formations (objectives 9 and 10), such as the square and the triangle in Figure 3.11(b), the pentagon and the same triangle in 3.11(c), and the

hexagon in 3.11(f) that started to form in 3.11(d).



(a) Configuration at $k = 0$ **(b)** Configuration at $k = 49$ **(c)** Configuration at $k = 263$

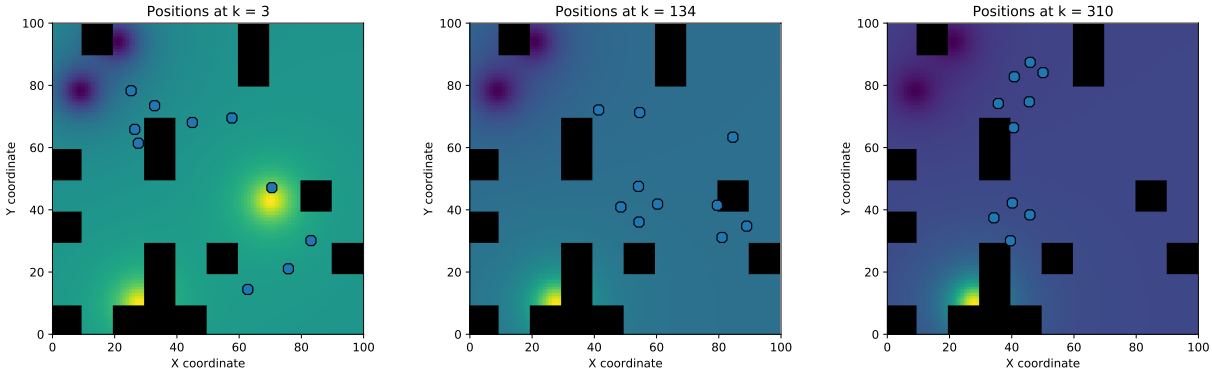
(d) Configuration at $k = 425$ **(e)** Configuration at $k = 473$ **(f)** Configuration at $k = 500$

Figure 3.11: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values and multiple static minima. Using *Algorithm 2*.

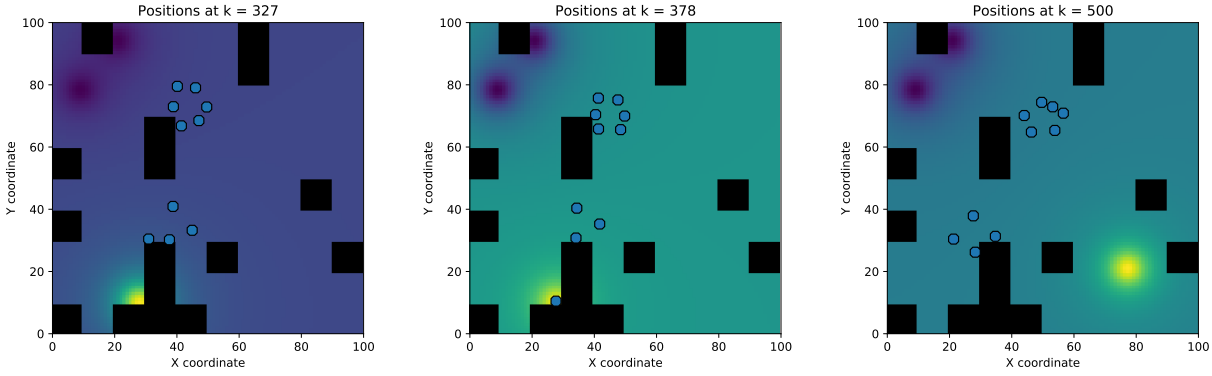
3.7.3.H Multiple Static Rendezvous Areas with Dynamic Values in a Mission Plane with Unauthorized Areas

Figures 3.12(a) to 3.12(f) illustrate type 3d. The results from the simulation of type 3c persist, and additionally, the agents do not enter the unauthorized areas (totaling to objectives 1 to 6, and 8 to 10). Objective 7 is not considered to be achieved because the formation of six agents observed at the top of Figure 3.12(c) does not leave the area, as evidenced by 3.12(d), 3.12(e), and 3.12(f). This is attributed to all the agents in the cluster having the same utility values and being in a zero-gradient area. This results in their final control law being governed by the *Formation* component - non-zero component with the highest weight (*Utility* and *Attraction* are estimated to be zero-vectors here). From the path of the

lone agent at the lower rendezvous area in Figure 3.12(e), from its position in 3.12(c) to 3.12(e), this simulation shows the agents reach a desired area in the presence of static environmental obstacles in their path.



(a) Configuration at $k = 3$ **(b)** Configuration at $k = 134$ **(c)** Configuration at $k = 310$

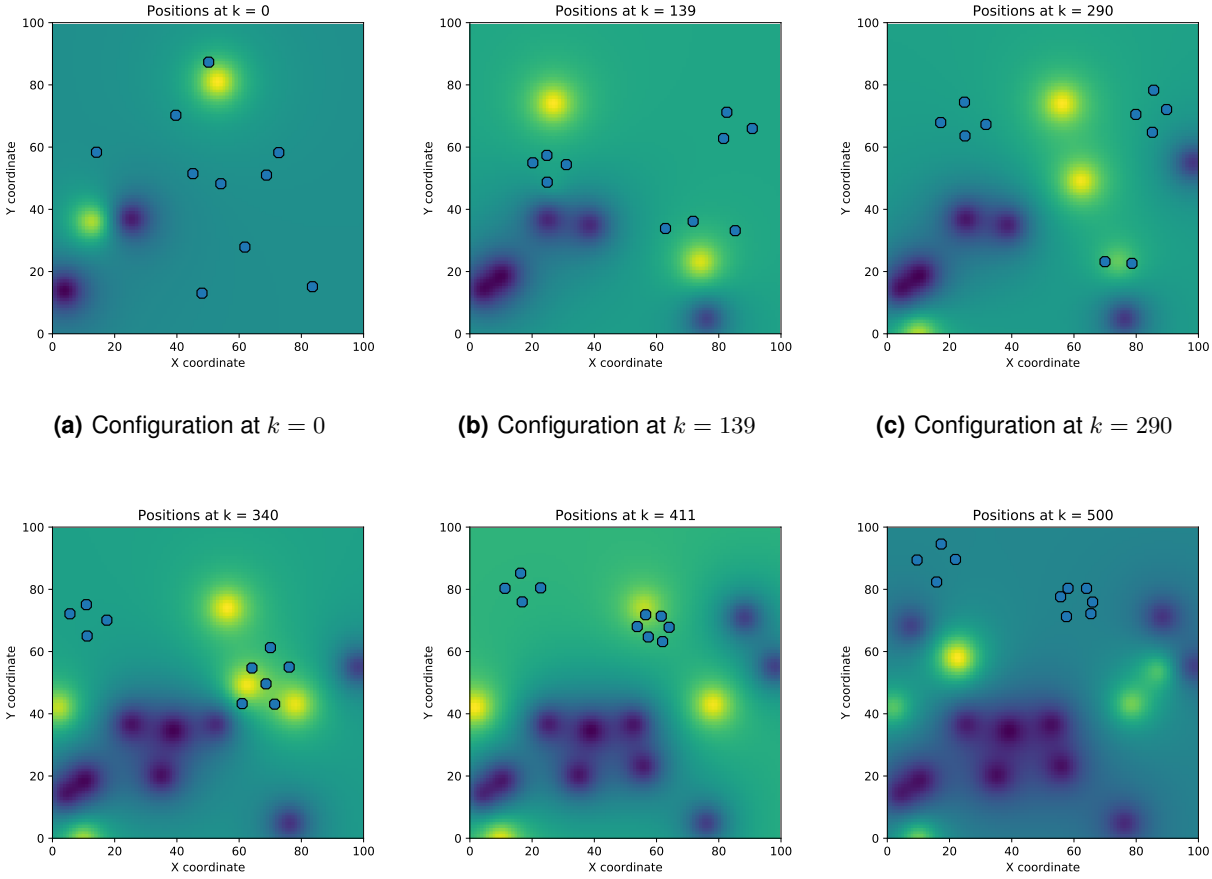


(d) Configuration at $k = 327$ **(e)** Configuration at $k = 378$ **(f)** Configuration at $k = 500$

Figure 3.12: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values, multiple static minima, and unauthorized areas (black squares). Using *Algorithm 2*.

3.7.3.I Multiple Static Rendezvous Areas with Dynamic Values and Random Creation/Destruction of Rendezvous Areas

Figures 3.13(a) to 3.13(f) illustrate type 3e - multiple static rendezvous areas with dynamic values in a mission plane with random creation and destruction of maximum and minimum areas. The random creation/destruction process is explained in Section 2.7.3.I. This simulation demonstrates the clusters converge to multiple rendezvous areas in the presence of a highly-dynamic utility function. All objectives - 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 - are achieved.



(a) Configuration at $k = 0$ **(b)** Configuration at $k = 139$ **(c)** Configuration at $k = 290$

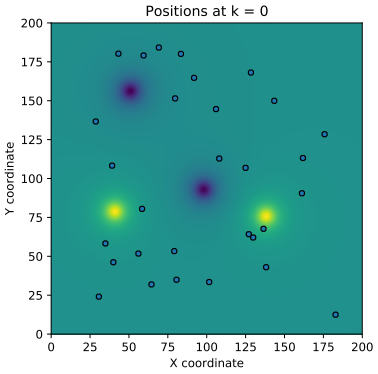
(d) Configuration at $k = 340$ **(e)** Configuration at $k = 411$ **(f)** Configuration at $k = 500$

Figure 3.13: Agents configurations for a simulation with a mission plane with multiple static maxima with dynamic values and multiple static minima. During the simulation, multiple rendezvous and undesired areas are randomly added or removed. Using *Algorithm 2*.

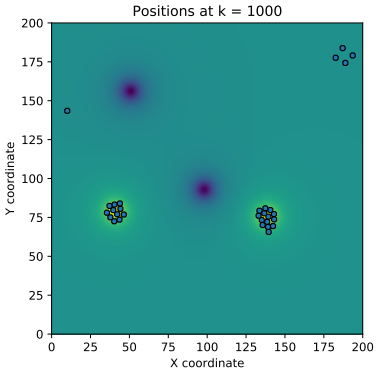
3.7.3.J Multiple Static Rendezvous Areas in a 200×200 Mission Plane with 30 agents

This section presents multiple simulations of type 3f - multiple static rendezvous areas in a 200×200 mission plane with 30 agents. The first simulation considers agents with a sensing radius, SR, of 15 distance units, depicted in Figures 3.14(a) and 3.14(b). It is shown 25 of the 30 agents converge to the rendezvous areas from the initially disconnected network topology. The limited rendezvous was initially attributed to the small sensing radii compared to the size of the mission plane. To verify this hypothesis, further simulations were executed from the initial configuration shown in Figure 3.14(a), with increasing values of sensing radii. From Figures 3.14(c) to 3.14(e), it can be seen that hypothesizing that the imperfect rendezvous is only caused by the value of the sensing radius (compared to the size of the mission plane) is inaccurate. Even with $SR = 45$, full convergence is not achieved in this mission plane.

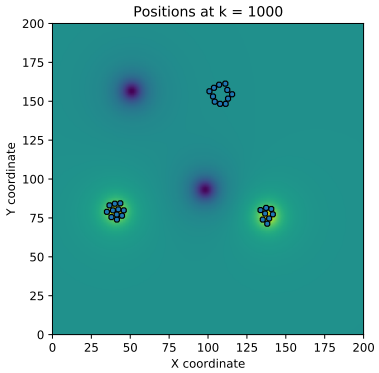
With $SR = 60$, almost all agents reach a rendezvous area. With smaller SR values (15, 35, and 45), the partial rendezvous is caused by the small sensing radii relative to the mission plane's size, as it had been previously proposed, and to the agents' prioritization of creating and maintaining formations over exploring the mission plane when they are in areas of uniform and negligible utility values.



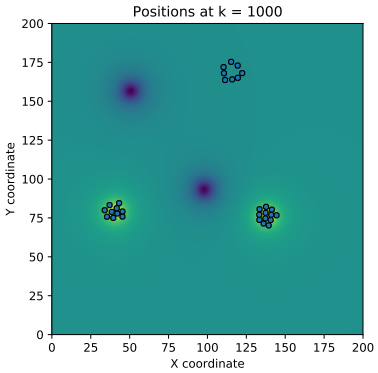
(a) Configuration at $k = 0$



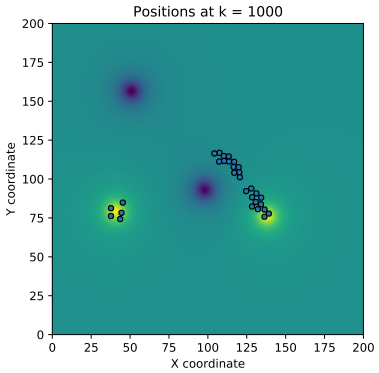
(b) Configuration at $k = 1000$ with $SR = 15$



(c) Configuration at $k = 1000$ with $SR = 35$



(d) Configuration at $k = 1000$ with $SR = 45$



(e) Configuration at $k = 1000$ with $SR = 60$

Figure 3.14: Agents configurations for a simulation with a 200×200 mission plane with multiple static maxima and minima and 30 agents with the sensing radius indicated in the captions (15, 35, 45, 60). Using *Algorithm 2*.

3.7.4 Results Discussion

As presented in Section 3.7.1, the design objectives to be tested by the simulations were the following:

1. agents move towards and remain in the rendezvous areas while these are in the utility function;
2. agents are repelled by the undesired areas;

3. agents do not enter unauthorized areas;
4. agents do not leave the mission plane;
5. agents do not collide;
6. indefinite movement deadlocks do not occur;
7. agents eventually leave low-utility areas;
8. all agents move in every discrete time instant;
9. agents enter and remain in formation;
10. agents do not compete for a formation slot.

Objectives 4, 5, 6, and 8 were verified across all simulation scenarios presented. Concerning Objective 6, as explained in Section 3.7.2, the final figures presented for each simulation might erroneously imply the agents are in a deadlocked state or in irrelevant positions for the mission objective, which is contradicted by the simulation videos corresponding to the evolution for all time instants.

The simulations presented in Section 3.7.3.C and 3.7.3.D serve to test the repelling property of the algorithm from undesired areas, may they be static or moving. From the simulation figures, it can be observed that this objective is fully accomplished.

Analogously, the primary objective of the simulations presented in 3.7.3.F and 3.7.3.H is to verify the effectiveness of the environmental obstacle collision avoidance method. As described in Section 2.5.5.B and further explained in Section 2.7.4, the implemented method does not provide exact collision detection and can result in an agent polytope partially intersecting an unauthorized area. This can be observed in Figure 3.12(b), where an agent of the triangular formation on the lower right is intersecting the square-shaped unauthorized area. Consequently, objective 3 is not fully accomplished.

Regarding one of the fundamental differences from *Algorithm 1*, agent formations can be observed across all simulations. Moreover, the structures change as the simulation progresses depending on the number of agents participating: Figure 3.13(c) illustrates two formations on the right side, a pair and a square; these two formations are then merged (evidenced by Figure 3.13(d)) to form a hexagon in Figure 3.13(e). There is no observable evidence of the agents competing for a formation slot. Therefore, objectives 9 and 10 are considered fully accomplished.

The simulation presented in Section 3.7.3.H illustrates agents that remain in the same low-utility area - area with a low value for the mission objective - for approximately 40% of the simulation. This is attributed to the component weights in the control law: the *Formation* and *Utility* components both have the highest weight of all the components. Due to agents being in a zero-gradient area (evidenced by

the uniform color of the mission plane in their positions), they prioritize creating a formation over following the behavior of the other components, including the *Randomness* component which motivates the exploration of new areas. Thus, the resulting control input in this situation will include an acceleration that could possibly move the agents out of the zero-gradient utility (from the *Randomness* component) but will be dominantly guided by the acceleration calculated by the *Formation* component. The presence of statically-positioned environmental obstacles in proximity with the cluster can contribute to the absence of exploration. However, this phenomenon is additionally observed in the simulation type 3e (Section 3.7.3.I), which does not contain unauthorized areas: the square-shaped cluster remains in the same area from Figure 3.13(c) to 3.13(e), moving before 3.13(f), presumably due to the creation of the undesired area in its vicinity. Consequently, objective 7 is considered to only be partially accomplished - the agents generally eventually leave low-utility areas.

The primary objective of achieving rendezvous to desirable areas was a focus on all simulations, with the agents always reaching rendezvous areas. Whenever there was a single location (Sections 3.7.3.A and 3.7.3.B), all the agents achieved consensus on that position. This was accomplished even in the cases with undesirable areas (3.7.3.C and 3.7.3.D). On the other hand, when there are multiple possibilities (3.7.3.E to 3.7.3.J), the agents reached one of the rendezvous areas while avoiding environmental obstacles even when there were random changes in the utility function. Consequently, objective 1 is considered fully accomplished.

The simulations presented did not test how the system reacts to agents arbitrarily leaving and entering the network because it is hypothesized the agent behavior would be identical to when agents discover new neighbors or lose connection to existing ones. This hypothesis results from the implemented local-agent approach: an agent dynamically adapts to the number of neighbors. The execution of simulations that test this behavior is left for future work.

3.8 Conclusions

This Chapter addressed the problem of having a group of mobile agents rendezvous to multiple dynamic desired areas in formation with a fully-decentralized approach. No global knowledge of the utility function is assumed and the algorithm can cope with time-varying functions. The decentralization means that agents do not have communication capabilities but are equipped with localization sensors and can measure the position, velocity, and utility values within a sensing radius. Such an algorithm is particularly suited for missions in communication-denied environments. Thus, the solution is also resilient to communication failures and nodes joining and leaving the network as the control law is computed with local information.

The proposed solution avoids the need for an initially connected topology, synchronous communi-

cation, and leader selection protocols by defining the movement based on improved flocking rules that dynamically merge previously unknown agents in a single cluster while driving them to higher utility zones in the mission plane. If the gradient is available, this direction can also be used to further guide the formation. Due to the use of set-membership filters to obtain the estimates for the positions, the algorithm can leverage the knowledge that the true state is contained within the generated polytope to enforce no agent-agent collisions. The localized movement rule also results in a scalable solution given that the computation only uses nearby node information with a complexity that is dependent solely on the number of neighbors. The efficacy of the algorithm is illustrated using various simulated scenarios with sparse agent occupation. It is shown that agents generally achieve rendezvous while in formations and avoiding undesired and unauthorized areas. Unlike the literature examples presented in Sections 2.2 and 3.2, *Algorithm 2* rendezvous the mobile agents to multiple rendezvous areas, with unknown positions and movement dynamics, while creating and maintaining dynamic leaderless formations whose structure is not predetermined and can adapt to agents joining and leaving the neighbor set.

3.8.1 System Limitations

As discussed in Section 3.7.4, the agents can partially intersect the unauthorized areas due to the simplified method to check collisions with the obstacles. This limitation appearing in the simulations can be solved with the solution presented in 2.8.2 that is based on ray-tracing and applied to detecting future collisions between agents.

Objective 7 is considered partially accomplished, given the weight selection to prioritize formations when the *Utility* and *Attraction* components' values are close to zero. This excessive priority results in the system limitation of agents not leaving low-utility areas for the mission objective. Section 3.8.2 presents a possible solution to mitigate this limitation.

3.8.2 Future Work

Similar to *Algorithm 1*, improvements in algorithmic efficiency are also left for future work, such as the vertex culling method in the ray tracing process, based on *back-face culling* in [20].

Another direction is the further development of techniques to fully address objective 3 by using the collision detection method used for the agents to prevent running into unauthorized zones. This can imply a higher computational cost depending on the number of obstacles. In addition, methods based on obstacle circumvention can also be applied to better guide the agents.

To improve the partial accomplishment of objective 7, it is left for future work the development of dynamic weights for the movement components of the control law (3.8). For instance, if the agent has been in a zone with a small utility value compared with the sensed vicinity, the weight of the *Randomness*

component can be increased to promote exploration. Comparatively, whenever nodes are close to a rendezvous area (detected by the maximum utility within the sensing radius), the *Utility* weight can be decreased to favor the maintenance of the formation.

In the current proposal, agents decide in a greedy fashion which positions to occupy in the formation. Even though no conflicts were detected in the simulations, these selfish choices also cause a higher social cost for the entire formation. A future development could include either algorithms or better heuristics that can reduce the average movement of the agents.

4

Conclusions

Contents

4.1 Conclusions

This dissertation addressed the problem of having a group of mobile agents rendezvous to multiple dynamic desired areas without assuming the connectedness of the network topology while providing guarantees for an environment without agent-agent collisions in the presence of noisy measurements.

Two algorithms were proposed: *Algorithm 1* - a partially-decentralized approach - considers agents without localization capabilities and only one-way tower-to-agent communication (with no agent-to-agent communication), and *Algorithm 2* - a fully-decentralized approach - considers agents with localization capabilities (which consequently avoids the need for measurement/communication towers) and no communication. *Algorithm 2* has the additional objective of creating and maintaining formations (within each cluster). *Algorithm 1* requires measurement/communication towers that transmit positioning data to the agents in directional broadcasts.

Algorithm 1 and *Algorithm 2* improve on the examples present in the literature with the following aspects:

- There is no apriori information regarding the number, positions, and movement dynamics of the rendezvous areas. Additionally, this number is not fixed: new rendezvous areas are created arbitrarily and areas are removed from the mission plane by being explored by the agents. By having a flocking-based movement algorithm, the agents explore the mission plane and find these desirable areas autonomously.
- The network topology does not need to be initially connected. Examples were shown of agents in an initially disconnected topology reaching a single cluster. The flocking-based movement used dynamically merges two previously independent clusters and mitigates agent separation.
- Due to the position and velocity measurements being corrupted by noise, they are estimated as worst-case sets in order to achieve effective collision avoidance.
- The collision avoidance method implemented provides guarantees of an environment without agent-agent collisions with each agent predicting the worst-case set of all possible positions for its neighboring agents considering their past known positions by extending the worst-case sets estimated during the measurement process.
- The agents have no apriori knowledge on the total number of agents in the system or any agent's initial position, unlike literature examples based on consensus, which require the agents to have apriori knowledge of least a subset of the agents in the network.
- All agents are equal, following the same control law, requiring no leader selection protocols or specialized message transmissions. Due to the agents following a flocking-based movement algorithm, they are autonomously organized. In the case of formations, where most literature examples

require a leader, our proposed agents do not require leaders or consensus contracts to achieve and maintain stable formation structures due to having an adaptable slot assignment protocol.

- There is no communication between agents unless the utility function cannot be measured at all points inside the sensing radius in *Algorithm 2*.

Additionally, the proposed algorithms focus on the following objectives:

- Deploying to networks of large size. There is no computational limit on the number of allowed agents in the network or a limit to the mission plane size - although the convergence efficiency will generally decrease if the size of the mission plane is larger than the range of the communication towers. This contribution is computationally infeasible with pure-consensus approaches.
- No single point-of-failure. The system will function while there are at least one agent and at least one communication tower (in *Algorithm 1*), or just at least one agent (in *Algorithm 2*).

Due to requiring external measuring equipment - measuring/communication towers - and the localization equipment and processes being offloaded to the towers, *Algorithm 1* is better suited for urban environments, in which the towers - whose implementation can be seen as fixed towers in the mission plane's perimeter or more advanced mobile structures - can be easily deployed (when fixed) or can dock to extend their reduced battery autonomy (when mobile), compared to the mobile agents. Because the positioning data is measured by the communication towers, *Algorithm 1* is ideal for operations in GPS-denied environments, such as open-space indoor navigation.

Algorithm 2, by not requiring communication, is ideally suited for missions in communication-denied environments such as extremely rural areas. Additionally, *Algorithm 2* maximizes the system availability by being fully-decentralized: only reaching an irreversible failed state when all agents experience failure and leave the network.

The efficacy of the algorithms is illustrated using various simulated scenarios with sparse agent occupation. It is shown that agents generally achieve rendezvous while avoiding undesired and unauthorized (environmental obstacles) areas.

4.2 System Analysis

4.2.1 Goals

The objectives for *Algorithm 1* and *Algorithm 2* were the following:

1. agents move towards and remain in the rendezvous areas while these are in the utility function;
2. agents are repelled by the undesired areas;

3. agents do not enter unauthorized areas;

4. agents do not leave the mission plane;

5. agents do not collide;

6. indefinite movement deadlocks do not occur;

7. agents eventually leave low-utility areas.

As described in Sections 2.7.4 and 3.7.4, objectives 1, 2, 4, 5, and 6 were fully accomplished by both algorithms. Objective 3 is considered to be partially accomplished (in both algorithms) because the environmental obstacle collision detection method does not provide exact results: an agent's polytope can partially intersect the unauthorized areas (that simulate the statically-positioned environmental obstacles). Objective 7 is considered fully accomplished in *Algorithm 1* but only partially accomplished in *Algorithm 2*. The partial completion in the second algorithm is attributed to the agents' prioritization of creating and maintaining formations over exploring when in low-utility areas for the mission objective.

Algorithm 2 had the following additional goals:

8. all agents move in every discrete time instant;

9. agents enter and remain in formation;

10. agents do not compete for a formation slot.

As described in Section 3.7.4, objectives 8, 9, and 10 are considered fully accomplished.

4.2.2 System Limitations

In *Algorithm 1*, due to agents receiving positioning data from the communication towers through the broadcasts to all the agents in the strip, the positions have to be interpreted identically by all receivers, which requires the agents' frames of reference to be identical. Due to this limitation, before an agent enters the network, its coordinate system must be aligned with the common coordinate system accepted by all agents in the system.

In *Algorithm 1* and *Algorithm 2*, the agents can partially intersect the unauthorized areas that simulate the environmental obstacles.

An additional limitation of *Algorithm 2* results from the agents' prioritizing creating and maintaining formations over leaving low-utility areas to the mission objective.

4.3 Future Work

Three different avenues can be explored in future work:

- The improvement of the position prediction method such that each agent calculates the probable positions in which its neighbors will be multiple time steps in the future. This prediction module would be used for the agents to better choose their desired movement acceleration, which would generally increase the convergence efficiency.
- The addition of a computer vision module to effectively replace the localization sensors in *Algorithm 2* for less expensive and more power-efficient cameras.
- The addition of a reinforcement learning method that modulates the weights of the movement components in the control law based on the agent's current situation.

Bibliography

[1] H. Qiu and H. Duan, “Multiple uav distributed close formation control based on in-flight leadership hierarchies of pigeon flocks,” *Aerospace Science and Technology*, vol. 70, 08 2017. [Online]. Available: https://doi.org/10.1016/j.ast.2017.08.030

[2] X. Fu, J. Pan, H. Wang, and X. Gao, “A formation maintenance and reconstruction method of uav swarm based on distributed control,” *Aerospace Science and Technology*, p. 105981, 06 2020. [Online]. Available: https://doi.org/10.1016/j.ast.2020.105981

[3] R. Parasuraman, J. Kim, S. Luo, and B. Min, “Multipoint rendezvous in multirobot systems,” *IEEE Transactions on Cybernetics*, vol. 50, no. 1, pp. 310–323, Jan 2020. [Online]. Available: https://doi.org/10.1109/TCYB.2018.2868870

[4] A. Turgeman, A. Datar, and H. Werner, “Gradient free source-seeking using flocking behavior,” in *2019 American Control Conference (ACC)*, July 2019, pp. 4647–4652. [Online]. Available: https://doi.org/10.23919/ACC.2019.8815372

[5] C. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” vol. 21, 07 1987, pp. 25–34. [Online]. Available: https://dl.acm.org/doi/10.1145/280811.281008

[6] F. Faure and B. Raffin, “Ray-traced collision detection for deformable bodies,” *3rd International Conference on Computer Graphics Theory and Applications, GRAPP 2008*, 01 2008. [Online]. Available: https://doi.org/10.5220/0001097902930299

[7] S. Moe, K. Pettersen, and J. Gravdahl, “Set-based collision avoidance applications to robotic systems,” *Mechatronics*, vol. 69, p. 102399, 08 2020. [Online]. Available: https://doi.org/10.1016/j.mechatronics.2020.102399

[8] A. Singha, A. Ray, and A. Samaddar, “Trajectory tracking in the desired formation around a target by multiple uav systems,” *Procedia Computer Science*, vol. 133, pp. 924–931, 01 2018. [Online]. Available: https://doi.org/10.1016/j.procs.2018.07.092

[9] R. Olfati-Saber and R. Murray, “Distributed cooperative control of multiple vehicle formations using structural potential functions,” *IFAC Proceedings Volumes*, vol. 35, 07 2002. [Online]. Available: https://doi.org/10.3182/20020721-6-ES-1901.00244

[10] F. Mehdifar, C. Bechlioulis, F. Hashemzadeh, and M. Baradarannia, “Prescribed performance distance-based formation control of multi-agent systems,” *Automatica*, vol. 119, p. 109086, 09 2020. [Online]. Available: https://doi.org/10.1016/j.automatica.2020.109086

[11] R. Ribeiro, D. Silvestre, and C. Silvestre, *Decentralized Control for Multi-agent Missions Based on Flocking Rules*, 09 2020, pp. 445–454. [Online]. Available: https://doi.org/10.1007/978-3-030-58653-9_43

[12] R. Ribeiro, D. Silvestre, and C. Silvestre, “Partially decentralized formation control based on flocking rules,” *Systems & Control Letters*, under review.

[13] R. Olfati-Saber and R. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520 – 1533, Sep. 2004. [Online]. Available: https://doi.org/10.1109/TAC.2004.834113

[14] J. Cortes, S. Martinez, and F. Bullo, “Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions,” *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, Aug 2006. [Online]. Available: https://doi.org/10.1109/TAC.2006.878713

[15] K. N. Krishnanand and D. Ghose, “Detection of multiple source locations using a glowworm metaphor with applications to collective robotics,” in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, June 2005, pp. 84–91. [Online]. Available: https://doi.org/10.1109/SIS.2005.1501606

[16] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: algorithms and theory,” *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, March 2006. [Online]. Available: https://doi.org/10.1109/TAC.2005.864190

[17] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, “Set-consensus using set-valued observers,” in *American Control Conference (ACC), 2015, Chicago, Illinois, USA.*, July 2015.

[18] K. Chung and W. Wang, “Quick collision detection of polytopes in virtual environments,” in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, ser. VRST ’96. New York, NY, USA: ACM, 1996, pp. 125–132. [Online]. Available: http://doi.acm.org/10.1145/3304181.3304206

[19] G. Zachmann, “Rapid collision detection by dynamically aligned dop-trees,” in *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No.98CB36180)*, March 1998, pp. 90–97. [Online]. Available: https://doi.org/10.1109/VRAIS.1998.658428

[20] P. Jimenez, F. Thomas, and C. Torras, “3d collision detection: A survey,” *Computers & Graphics*, vol. 25, pp. 269–285, 04 2001. [Online]. Available: https://doi.org/10.1016/S0097-8493(00)00130-8

[21] R. Ribeiro, D. Silvestre, and C. Silvestre. Master dissertation: Simulation figures and videos. [Online]. Available: https://github.com/RafaelMenesesRibeiro/MasterDissertation

[22] R. Ribeiro, D. Silvestre, and C. Silvestre, “Decentralized formation control for multi-agent systems based on flocking rules,” *IEEE Transaction on Control of Network Systems*, under review.

[23] H. Mohammadi, M. Razaviyayn, and M. R. Jovanovic, “Robustness of accelerated first-order algorithms for strongly convex optimization problems,” *IEEE Transactions on Automatic Control*, pp. 1–1, 2020. [Online]. Available: https://doi.org/10.1109/TAC.2020.3008297