University of Modena and Reggio Emilia

Department of Sciences and Methods for Engineering

Doctoral school of Industrial Innovation Engineering

XXXV cycle

Control and Coordination of a Multi-Robot System: Bridging the Gap Between Theory and Real Scenarios

---

Author:
Federico Pratissoli

Supervisor:
Lorenzo Sabattini

PhD Coordinator:
Franco Zambonelli

October 2022

Copyright© 2022 by Federico Pratissoli

Reggio Emilia, Italy, October 2022

To my family

Acknowledgements

I would like to thank first of all my family, for always believing in me and unconditionally supporting me. Without them I would not have been able to complete this journey. I will be forever grateful for all the advice and lessons you gave me.
I would like to deeply thank my advisor, Prof. Lorenzo Sabattini, for the great help during these years, for the great support he always gave me and esplecially for his patience. I also thank all the people from the ARS Control lab, the lab mates and the professors, and the people I had the chance to work with. I am very grateful for this journey, that gave me the opportunity to learn, study, travel and meet amazing people around the world and finally that gave me the opportunity to improve the research by a very small, hopefully not irrelevant, fraction.

Abstract

Multi-robot systems (MRS) are rapidly improving and thanks to technological innovations they will become part of our everyday life. Multi-Robot Systems have emerged as a suitable alternative to single robots, since they can be more efficient, flexible and fault tolerant. However, the complexity of these systems poses some challenges at the time of their deployment, control and recovery. The near future needs robust control strategies and approaches to allow exploiting the potential of these systems for many useful automation applications in diverse domains. The aim is to enhance, for example, productivity, quality, accuracy, flexibility, cost-effectiveness, reliability and safety using a coordinated group of robots. This coordination depends on different features as sensing, actuation, communication, recognition, learning, perception and controlling capabilities.

In this thesis we proposed new control strategies to take multi-robot systems one step forward towards their applications in real world and their common usage. Firstly, we focused on the ability of a group of robots to achieve and perform required tasks in a distributed way. The computation and the control strategy is not centralised on a single computational unit, but every agent in the group is able to make its own decisions to improve the performance of the whole multi-robot system. In particular, the task the group of robots has to achieve is to maximise the coverage of an unknown environment for monitoring or exploration. This type of problem is well known and usually faced exploiting coverage based control strategies. These techniques are generally based on mild assumption, such as known environment and a global knowledge about the multi-robot system. Hence, we proposed a distributed coverage control law that considers the limited sensing range of robots, does not rely on a communication network and is able to deal with unknown and non-convex environments.

Moreover, coverage based control strategies are used only for monitoring purposes or to move the group of robots towards areas of higher interest. We studied the possibility for the multi-robot system to learn and estimate a feature or spatial process in an unknown environment. This spatial process can be the temperature, the pollution in a big city or the salinity in the ocean. We studied a methodology that allows the multi-robot system

to optimally learn and estimate an unknown and time-invariant process and uses this information to obtain a coverage of the environment depending on the distribution of this spatial process, which is modelled by a non-uniform density distribution. Finally, we managed to consider a scenario in which the spatial process changes over time. In fact, we get rid of the assumption that the spatial process is time invariant and, hence, we have a distribution that has features that are changing over time. Another important challenge is to control and coordinate a group of robots deployed in a real industrial scenario. We studied the traffic management and the coordination of a fleet of automated vehicles in an automated factory or warehouse. In particular, we proposed a hierarchical architecture that deals with the path planning and the coordination of the vehicles with the aim to optimise the efficiency, avoid congestions, limit the traffic, avoid collisions and prevent deadlocks. In particular the control approach we propose does not rely on ad-hoc assumptions or rules and is able to manage vehicles errors and delays.

Sommario

I sistemi multi-robot (MRS) stanno rapidamente migliorando e grazie alle innovazioni tecnologiche diventeranno parte della nostra vita quotidiana. I sistemi multi-robot sono emersi come un'alternativa adeguata ai singoli robot, poiché possono essere più efficienti, flessibili e tolleranti ai guasti. Tuttavia, la complessità di questi sistemi pone alcune sfide al momento della loro implementazione, del controllo e del ripristino da errori. Il prossimo futuro richiede strategie e approcci di controllo robusti che consentano di sfruttare il potenziale di questi sistemi per molte applicazioni di automazione utili in diversi ambiti. L'obiettivo è quello di migliorare, ad esempio, la produttività, la qualità, l'accuratezza, la flessibilità, l'economicità, l'affidabilità e la sicurezza utilizzando un gruppo coordinato di robot. Questa coordinazione dipende da diverse caratteristiche come le capacità di rilevamento, attuazione, comunicazione, riconoscimento, apprendimento, percezione e controllo.

In questa tesi abbiamo proposto nuove strategie di controllo per far fare ai sistemi multirobot un passo avanti verso le loro applicazioni nel mondo reale e il loro uso nella vita quotidiana. In primo luogo, ci siamo concentrati sulla capacità di un gruppo di robot di eseguire i compiti richiesti in modo distribuito. Il calcolo e la strategia di controllo non sono centralizzati su una singola unità di calcolo, ma ogni agente del gruppo è in grado di prendere le proprie decisioni per migliorare le prestazioni dell'intero sistema multirobot. In particolare, il compito che il gruppo di robot deve svolgere è quello di massimizzare la copertura di un ambiente sconosciuto per il monitoraggio o l'esplorazione. Questo tipo di problema è ben noto e di solito viene affrontato sfruttando strategie di controllo basate sul così detto "Coverage Control". Queste tecniche si basano generalmente su assunzioni poco veritiere, come un ambiente noto e una conoscenza globale del sistema multi-robot. Abbiamo quindi proposto una legge di controllo di tipo distribuito che tiene conto del limitato raggio di rilevamento dei robot, non si basa su una rete di comunicazione ed è in grado di gestire ambienti sconosciuti e non convessi.

Inoltre, le strategie di controllo basate sul "Coverage Control" sono utilizzate generalmente per scopi di monitoraggio o per spostare il gruppo di robot verso aree di maggiore

interesse. Abbiamo studiato la possibilità per il sistema multirobot di apprendere e stimare un aspetto o un fenomeno sparso nello spazio di un ambiente sconosciuto. Questo fenomeno o processo spaziale può essere ad esempio la temperatura, l'inquinamento di una grande città o la salinità dell'oceano. Abbiamo studiato una metodologia che consente al sistema multirobot di apprendere e stimare in modo ottimale un fenomeno sconosciuto e costante nel tempo e di utilizzare queste informazioni per ottenere una copertura dell'ambiente in funzione della distribuzione di questo fenomeno, modellato da una distribuzione di densità non uniforme. Infine, siamo riusciti a considerare uno scenario in cui tale fenomeno cambia nel tempo. Infatti, eliminiamo l'ipotesi che il processo sia invariante nel tempo e, quindi, abbiamo una distribuzione che ha caratteristiche che cambiano nel tempo. Un'altra sfida importante è quella di controllare e coordinare un gruppo di robot in uno scenario industriale reale. Abbiamo studiato la gestione del traffico e il coordinamento di una flotta di veicoli automatizzati in una fabbrica o in un magazzino automatizzato. In particolare, abbiamo proposto un'architettura gerarchica che si occupa della pianificazione del percorso e del coordinamento dei veicoli con l'obiettivo di ottimizzare l'efficienza, evitare le congestioni, limitare il traffico, evitare le collisioni e prevenire i così detti "stalli" (o "deadlocks"). In particolare, l'approccio di controllo che proponiamo non si basa su assunzioni o regole ad hoc ed è in grado di gestire gli errori e i ritardi dei veicoli.

Contents

Acknowledgements **v**

Abstract **vii**

Sommario **ix**

Table of Contents **xi**

List of Figures **xiii**

List of Publications **xxi**

1 Introduction **1**

1.1 Contribution and Thesis Outline 3

2 Coverage Based Control **7**

2.1 Notation and Definitions 8

2.2 Coverage Control - Background 10

2.2.1 Robot Model 10

2.2.2 Coverage Control 10

2.3 Coverage Control for Limited Range Multi-Robot Systems 12

2.3.1 Literature Review 13

2.3.2 Problem Statement 14

2.3.3 Partitioning of The Environment - Limited Voronoi Diagram . . . 15

2.3.4 Proposed Control Strategy 17

2.3.5 Numerical Issues - Variance Varying Control 20

2.3.6 Experimental Validations 22

2.3.7 Discussions 28

2.4 Coverage Control for Large-Scale Teams of Aerial Drones 29

2.4.1 Literature Review 29

2.4.2 Tests and Setups 30

2.4.3 Simulation Protocol 30

2.4.4 Field Test Protocol 31

2.4.5 Metrics 33

2.4.6 Results and Analysis 34

2.4.7 Discussions 38

2.5 Coverage Control for Exploration of Unknown Non-Convex Environments 39

2.5.1 Literature Review . . . 39
2.5.2 Problem Statement . . . 41
2.5.3 Proposed Strategy: Time-Varying density function . . . 42
2.5.4 Simulations and Experiments . . . 44
2.5.5 Discussions . . . 48
2.6 Coverage Control for Time-Varying Spatial Processes . . . 48
2.6.1 Literature Review . . . 50
2.6.2 Notation and Coverage Control . . . 51
2.6.3 Problem Statement . . . 52
2.6.4 Gaussian Process Regression . . . 52
2.6.5 Exploration - Exploitation Problem . . . 54
2.6.6 Data Sampling Strategy . . . 55
2.6.7 Time Varying Spatial Field . . . 56
2.6.8 Experimental Validations . . . 58
2.6.9 Performance Evaluation . . . 66
2.6.10 Discussions . . . 69

3 Traffic Management 71
3.1 Literature Review . . . 72
3.2 Algebraic Graph Theory . . . 75
3.3 System Overview . . . 76
3.4 Multi-Layer Architecture . . . 79
3.4.1 Roadmap Layer (Bottom Layer) . . . 79
3.4.2 Topological Layer (Top Layer) . . . 80
3.4.3 Middle Layer . . . 82
3.5 Hierarchical Path Planner . . . 83
3.5.1 Path Planning . . . 84
3.6 The Traffic Manager . . . 85
3.6.1 AGVs Coordination . . . 88
3.6.2 The Deadlock Prevention Policy . . . 90
3.7 Software Implementation . . . 92
3.7.1 Computational Complexity . . . 94
3.7.2 Simulations . . . 96
3.7.3 Experimental validation: Industrial Scenario . . . 98
3.8 Discussions . . . 101

4 Main Findings and Future Directions 103
4.1 Main Findings . . . 103
4.2 Future Works . . . 105

List of Figures

2.1 The Voronoi partitioning of a limited portion of environment around the robot i. The hatched area in the middle is the Voronoi cell V_i of the robot located in p_i. 8

2.2 The Voronoi partitioning and the R-limited Delaunay graph (dashed line) with seven seeds. The circles represents the balls with radius $R/2$ around every seed. 9

2.3 This figure is summarizing the coverage control methodology from [1]. The control strategy is applied to 32 agents deployed in a convex polygonal environment, with a Gaussian density distribution centered in the top right area of the environment. The left figure illustrates the initial robots locations and the respective Voronoi diagrams. The middle figure illustrates the robots trajectories. The right figure illustrates the finale robots locations and the respective Voronoi diagrams. 12

2.4 Comparison between *exact* and *limited* Voronoi. The environment Q is reported in *blue*, the robots are the *red* triangles, and the centroid of the single Voronoi cell are the *green* crosses, the single Voronoi cell are delimited with *solid black* lines. Fig. 2.4b also reports in *dashed black* lines the wrong partitioning built considering the whole sensing range R, and in *purple* the overlapping areas. 15

2.5 In the figures the robots are represented by *triangles*. The *blue* area represents the Voronoi partitioning in the sensing range area around the i-th robot with respect to which we calculate the local Voronoi. The *red* area represents the remaining partitioned area in the sensing range associated with a neighbour (another robot within the sensing range R). In particular, the images show the local Voronoi partition when a neighbor is removed from $\mathcal{N}_{\mathcal{G}_{disk}}(p_i, \mathcal{P})$ (in the left images the robot has one neighbor, while in the right ones the robot has no neighbours). Fig. 2.5a and Fig. 2.5b report the local Voronoi partitioning result when computed over the whole sensing range R. Fig. 2.5c and Fig. 2.5d report the local Voronoi partitioning when computed over half of the sensing range. Hence, the resulting cells are intersected with $\overline{B}(p_i, \frac{R}{2})$. The *dotted* line represents the sensing range. 18

2.6 One-dimensional representation of the varying-variance with different values of z corresponding to the principal numerical representation ($\sigma = 0.01$, $\mu = 0$). 22

2.7 Examples of two surfaces calculated with (2.23) ($z_{max} = 30$, $\mu = (0, 0)$). . 22
2.8 Optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$, as introduced in (2.5), achieved by a 4 robots controlled with the proposed control law (2.12), considering increasing sensing range, and compared with the standard method. The proposed method converges to the standard one for a sufficiently high sensing range. 23
2.9 Difference between Voronoi partitioning with the standard method (in *red*) and the proposed method with $R = 2$m (in *blue*). Initial and final positions are represented with *circles* and *triangles*, respectively. 24
2.10 Optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$, as introduced in (2.5), achieved by a different number of robots controlled with the proposed control law (2.12). The *solid* lines represent the standard method, instead the *dashed* lines represent our method. 25
2.11 The figure shows one of the simulations we did in non-convex environments to evaluate and stress the capabilities of the proposed distributed control strategy. The robots try to optimize the coverage of the environment considering their sensing range. The coloured dots indicate the robots locations that are changing over time. The coloured crosses indicate the computed centroids. The dashed coloured lines indicate the robots sensing ranges. 25
2.12 Experiments with uniform density function. In 2.12b and 2.12a every colored shape is the limited Voronoi cell of the respective robot. In 2.12b the *circles* are the starting positions, the *triangles* are the final positions. . 26
2.13 Coverage based control strategy with uniform probability density function. The group of robots starting from a concentrated configuration are tasked to optimally cover uniformly the environment. The robots sensing ranges are assumed to be large enough to include the whole domain ($R = 3$ meters). The coloured dots indicate the robots locations. The coloured crosses indicates the Voronoi cells centroids computation. The Voronoi diagrams are distinguished by the coloured polygons. 27
2.14 Experiments with not uniform density function $\phi(q)$. Initial and final positions with *circles* and *triangles*, respectively. The *black* cross is the center of $\phi(q)$ and the *black* concentric circles are different values of variance. 28
2.15 Coverage based control strategy with Gaussian probability density function. The group of robots starting from a randomly spread configuration are tasked to optimally cover the environment around the area of interest defined by the Gaussian function ($\mu = (0.8, 0.8); \sigma = 0.2$). 28
2.16 Figure of the MRS F450, the quad-rotor platform selected for the experiments 31
2.17 Vehicles trajectories of a real experiment with 7 drones 32
2.18 Average value of $\mathcal{A}(\mathcal{P})$ across all repetitions. The plots on the same line refer to the same total area to cover, while the plots on the same column refer to cases with same sensing range. The values for 8, 32 and 64 robots are displayed with blue, red and yellow lines respectively. 34

2.19 Average value of $\mathcal{A}(\mathcal{P})$ across all repetitions. The plots on the same line refer to the same total area to cover, while the plots on the same column refer to cases with same sensing range. The values for 8, 32 and 64 robots are displayed with blue, red and yellow lines respectively. 35

2.20 Average value of $\mathcal{E}(\mathcal{P})$ across all repetitions. The plots on the same line refer to the same total area to cover, while the plots on the same column refer to cases with same sensing range. The values for 8, 32 and 64 robots are displayed with blue, red and yellow lines respectively. 36

2.21 Average value of Equations (2.26), (2.27), (2.28) across all repetitions. The plots on the same line display the same metric, while the plots on the same column refer to cases with same sensing range. 37

2.22 On image (a) the initial configuration of the vehicles during a real experiment with sensitivity range $R = 50$; on image (b), the final configuration of the drones in the same experiment. On image (c) an aerial photo of the convergence process is shown, with a different orientation of the axes. . . . 38

2.23 Probability distribution of the environment 45

2.24 Two of the scenarios tested in simulation exploiting Gazebo environment. The two experiments differ in number of robots, environment dimension and complexity. The blue circles represent the sensing areas of the robots. 46

2.25 The experimental set-up of one of the experiments conducted with real robotic platforms. Three Turtlebot3 burger robots were tasked to explore an unknown non-convex area. 47

2.26 Explored area in $400m^2$ environment 47

2.27 Comparison with other solutions 48

2.28 Explored area in field trials 49

2.29 Comparison between a time variant and time invariant approach with a 1-D signal estimation. We presume the 6 samples are tidily taken from left to right every 10 seconds. Hence, the older sample is in $x \sim 2$ and the latest is in $x \sim 10$. In the figure (a) with the time invariant approach all the samples are equally processed and the uncertainty of the estimation is zero in the sample locations. In the figure (b) we are using a time-varying learning approach, in which the samples are processed differently depending on the time they were taken. The older samples have greater uncertainties than the newer ones. Moreover the latest sample has a zero standard deviation since it was taken at the current time $t_6 = 60s$ and hence $\Delta t_6 = 0$. The parameters used in this example for the time varying approach are $\epsilon = 0.001$ and $\tau = 3000$. 59

2.30 The figure shows three different subfigures, one for each time step, of a simulation where the team of robots is tasked to estimate a constant spatial process and optimally cover it. Each subfigure shows four aspects of the simulation: the first image from the left shows the team of robots (blue dots), the Voronoi partitioning (black lines) and the computed centroid for every Voronoi cell (red dots). The second image from the left shows the uncertainty over the domain: yellow areas mean a high standard deviation and a high uncertainty, blue areas mean a low standard deviation and low uncertainty. The third image from the left shows the robots estimation up that instant of time. The fourth image from the left shows the unknown spatial process that the team of robots is tasked to estimate. 61
2.31 The figure shows four different subfigures, one for each time step, of a simulation where the team of robots is tasked to estimate a time-varying spatial process and optimally cover it. Each subfigure shows four aspects of the simulation: the first image from the left shows the team of robots (blue dots), the Voronoi partitioning (black lines) and the computed centroid for every Voronoi cell (red dots). The second image from the left shows the uncertainty over the domain: yellow areas mean a high standard deviation and a high uncertainty, blue areas mean a low standard deviation and low uncertainty. The third image from the left shows the robots estimation up that instant of time. The fourth image from the left shows the spatial process that changes over time and that the team of robots is tasked to estimate. 62
2.32 The figure shows three different subfigures, one for each time step, of a simulation where the team of robots is tasked to estimate a spatial process and optimally cover it. During the experiment we assume we have an unexpected event that is modelled as a step-like time decay function that can potentially change the spatial process. Each subfigure shows four aspects of the simulation: the first image from the left shows the team of robots (blue dots), the Voronoi partitioning (black lines) and the computed centroid for every Voronoi cell (red dots). The second image from the left shows the uncertainty over the domain: yellow areas mean a high standard deviation and a high uncertainty, blue areas mean a low standard deviation and low uncertainty. The third image from the left shows the robots estimation up that instant of time. The fourth image from the left shows the unknown spatial process that the team of robots is tasked to estimate. 64
2.33 The figure shows three different stages of a simulation where the team of drones is tasked to estimate a smoke distribution in a open environment in order to detect and monitor the respective fire. We assume each drone is able to detect the smoke particles concentration in the air and we assume the smoke has a Gaussian-like distribution in the environment. The team of drones initially explore the environment in order to learn the smoke distribution and finally uses this estimation to optimally cover the environment, monitoring more the area where the fire is concentrated. . . 65

2.34 The figure shows four different stages of a simulation where the team of drones is tasked to estimate a time-varying smoke distribution in a open environment in order to detect and monitor the respective fire. We assume each drone is able to detect the smoke particles concentration in the air and we assume the smoke has a Gaussian-like distribution in the environment. The team of drones initially explore the environment in order to learn the smoke distribution and to use this estimation to optimally cover the environment. After an initial period of time, a change in the location of the fire and in the corresponding smoke distribution occurred. The team of drones was able to adapt to the changes and to update the estimation of the process distribution. Finally, balancing the exploration-exploitation trade-off, the drones shifted again their focus towards the origin of the smoke. 66

2.35 The figure shows the performance comparison between the proposed coverage-based control strategy with three other approaches in a scenario where the robots are tasked to monitor a spatial process. The performance is evaluated by the optimization function of the coverage control that evaluate the robots location with respect to the spatial process location (eq. 2.35). The graphs reports the mean ad the standard deviation with 95% confidence interval. The green line represents the random exploration algorithm, the orange line represents the plain coverage-based control strategy where the robots lack any information on the spatial field. The red line represents the ideal coverage-based control in a scenario where the robots have a complete and accurate knowledge of the environment and the spatial process. The blue line represents the proposed coverage-based control strategy. . . 67

2.36 The figure shows the performance comparison between the proposed coverage-based control strategy with three other approaches in a scenario where the robots are tasked to monitor a time-varying spatial process. The performance is evaluated by the optimization function of the coverage control that evaluate the robots location with respect to the spatial process location (eq. 2.35). The graphs reports the mean ad the standard deviation with 95% confidence interval. The green line represents the random exploration algorithm, the orange line represents the plain coverage-based control strategy where the robots lack any information on the spatial field. The red line represents the ideal coverage-based control in a scenario where the robots have a complete and accurate knowledge of the environment and the spatial process. The blue line represents the proposed coverage-based control strategy. After 60s there is a sudden change of the spatial process. Both the ideal coverage control method and the proposed method exhibited a responsive behavior to this change. 68

3.1 The figure shows the factory structure (red, green, and black drawings) and the plant graph $\mathcal{G}$ of our use case, which is elicited by a real application. In particular, red nodes indicate load or unload locations, while red nodes indicate transit areas. 72

3.2 An example of directed time expanded graph used in the proposed coordination strategy. The graph considered in figure is composed by four nodes: A, B, C, D. The graph is considered over time and the edge set $\mathcal{E}(\mathcal{G}(i))$ at time $t = i$ can be different from the edge set $\mathcal{E}(\mathcal{G}(i+1))$ at time $t = i+1$. 77

3.3 Overview of the proposed multi-layer architecture. The figure show a portion of the three graphs, one for each layer, of the studied use case, from the bottom: the roadmap layer, the middle layer, the topological layer. On the roadmap layer the red nodes indicate the load or unload locations in the factory, while the white nodes indicate the transit locations. The blue dotted lines show the different plant areas, divided by topology, each one assigned to a sector S. 78

3.4 A full representation of the sector graph $\mathcal{V}$ used to model the traffic in our use case, which is elicited by a real industrial application. 81

3.5 An example of subsector division process. (a) connections among sectors A, B, C, D, E in $\mathcal{V}$; (b) nodes of $\mathcal{G}$ belonging to sector A, the subgraph defined by the set of nodes $\{1, .., 6\}$ is not connected; (c) the result of the subsector partitioning algorithm. 82

3.6 Visualization of coordination over time of 8 AGVs in the use case plant. The figure provides a visualization of the coordination of 8 AGVs evolved over time in the use case plant considered. On the plane XY the figure shows the node coordinates (the location) in the plant, while, on the Z axis the figure shows the time at which a node is crossed. Different colors indicates different vehicles, i.e. red is for AGV with number 1, blue is for AGV with number 6, etc. It is worth noting that the lines, as sequences of coloured nodes, never cross, since the simultaneous presence of two (or more) AGVs on the same node at the same time never takes place. This means that the coordination has been carried on properly and there are no collisions in the AGV trajectories. 86

3.7 A representation through the time expanded graphs of the precedences defined to coordinate three AGVs. Every layer corresponds to the graph representing the precedences, as result of the vehicles coordination, at a particular time step. For example, at time t_4 the coordination is managed by making the AGV 1 waiting for the AGV 2 and the AGV 2 waiting for the AGV 3. 92

3.8 A representation through the time expanded graphs of the precedences defined to coordinate three AGVs. Every layer corresponds to the graph representing the precedences, as result of the vehicles coordination, at a particular time step. The time expanded graph helped to avoiding a cyclic set of precedences and hence a deadlock at time t_4. The coordination is managed by making the AGV 1 waiting for the AGV 3 and not vice versa to avoid a deadlock scenario between the three vehicles. 93

3.9 The figure illustrates the computational benefits of a decoupled approach as compared to a standard coupled approach in terms of the path planning computation. The graph displays the computation time in seconds required to calculate the path for 8 AGVs in a roadmap with varying size between two of the most distant nodes in the graph. The discrepancy in computation time becomes more pronounced as the dimension and complexity of the graph increases. 95
3.10 The figure show an evaluation of the deadlock prevention policy algorithm computational cost. We compared the theoretical exponential computational cost with the one tested with several simulations. The orange and blue line show respectively the highest and the average number of iterations tested from simulations. The green line shows the theoretical expected number of iteration in the worst computational scenario. 96
3.11 The figure show an evaluation of the efficiency in the traffic management and vehicles coordination. The results were obtained by a serie of simulations conducted on the industrial use case scenario. The efficiency parameter aims to determine how much time is wasted due to the coordination. The blue line shows the average efficiency seen in several tests. The orange line shows the worsts values obtained from some tests. It is worth noting how the efficiency only slightly decreases, increasing the number of AGVs. 97
3.12 The figure shows the simulator provided by the company exploited to test the software built upon the proposed control architecture. We tested the software with several hours of simulations managing the coordination and the traffic of 8 automatic guided vehicles in the studied plant. Every group of coloured dots indicates an AGV location, the immediate future location and its occupied space due to its size. 97
3.13 A photo taken during the experiments conducted in the automated factory. The 4 AGVs are coordinating in a load and unload cargo area. 98
3.14 The figure shows the number of missions completed over time by the fleet of AGVs. The blue line indicates the company's software performance at full potential, while the orange one indicates the proposed software performance at full potential. The proposed software managed to coordinate 8 vehicles simultaneously. The company's software managed to coordinate 6 vehicles simultaneously. The green line shows the performance of the proposed software when constrained to move only 6 AGVs simultaneously. 99
3.15 The figure shows a heavy traffic scenario in which the control software has to deal with the coordination of 4 vehicles in a small area avoiding deadlocks. In fact, this area is modelled as a sector that can simultaneously contain a maximum of 4 vehicles. The time values indicated for each image aims to give an idea of the coordination timing. 100
3.16 Several experiments were conducted to stress the control strategy when dealing with static obstacles or vehicles errors. In this scenario a path or location is occupied for a long period by a vehicle. The coordination of the fleet of AGVs needs to consider this interruption in order to avoid congestions and adapt the traffic management consequently. 101

3.17 Several experiments were conducted to stress the control strategy when dealing with dynamic obstacles, communication errors or vehicles alarms. In this scenario a vehicle or a group of vehicles is temporarily interrupted in its motion delayed in fulfilling its task. The coordination strategy needs to deal with this delay since it affects the prevention of congestions and deadlocks. 101

List of Publications

Journal papers

- F. Pratissoli, R. Brugioni, N. Battilani, and L. Sabattini, "Hierarchical traffic management of multi-agv systems with deadlock prevention applied to industrial environments," *second revision round at IEEE Transactions on Automation Science and Engineering*, 2023

Conference papers

- F. Pratissoli, B. Capelli, and L. Sabattini, "On coverage control for limited range multi-robot systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022
- F. Bertoncelli, M. Belal, D. Albani, F. Pratissoli, and L. Sabattini, "On limited-range coverage control for large-scale teams of aerial drones: Deployment and study," in *The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2022
- M. Catellani, F. Pratissoli, F. Bertoncelli, and L. Sabattini, "Coverage control for exploration of unknown non-convex environments with limited range multi-robot systems," in *The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2022
- F. Pratissoli, N. Battilani, C. Fantuzzi, and L. Sabattini, "Hierarchical and flexible traffic management of multi-agv systems applied to industrial environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 009–10 015
- F. Pratissoli, L. Sabattini, and A. Prorok, "Distributed coverage control for time-varying spatial processes," in *submitted to Robotics: Science and Systems (RSS), 2023*, 2023

Chapter 1

Introduction

In 2011, a massive earthquake and tsunami caused a catastrophic failure at the Fukushima Daiichi Nuclear Power Plant, in Japan, resulting in the release of radioactive material: a multi-robot system was fundamental to deal with this nuclear disaster and to solve the cleanup of the radioactive area [8]. Cleaning up the disaster site was a massive undertaking that posed significant risks to human workers due to the high levels of radiation. A fleet of small robots was used to navigate through the cluttered spaces of the power plan, monitoring the air quality, detecting radiation levels and taking photos of the damages. Overall, the multi-robot system was able to significantly speed up the cleanup process and reduce the risks to human workers, making a significant contribution to the recovery efforts following the disaster.

A team of robots takes care of a crop field: [9, 10] the robots manage the watering to distribute water evenly across the field, ensuring that the crops are getting the optimal amount of water and minimizing the usage of pesticides and chemical products. Overall, the use of a multi-robot system in agriculture has the potential to revolutionize the way that farms are managed and operated, leading to increased efficiency and productivity.

In the hospital a group of robots can be in charge of disinfecting hospital rooms and surfaces, helping to reduce the risk of infection and to improve patient safety. Moreover, the robots can be used to transport medical supplies, such as medication and equipment, and to monitor the patients collecting data on their vital signs and alerting medical staff if needed [11]. Overall, the use of a multi-robot system in hospitals can help to improve the efficiency and safety of patient care, leading to better outcomes for patients.

These are some examples of real world applications in which the multi-robot system has been or can be significant in improving the work of people. Multi-robot systems have the potential to revolutionize a wide range of industries and applications, and they continue to be an active area of research and development in the field of robotics. This is confirmed by the recent reports [12, 13, 14, 15], where it is stated that the robotics industry will continue to experience significant growth in the coming decade. According to a report by PrecedenceResearch [16], robotics technology market size is expected to reach USD 225.6 billion by 2030 with a registered CAGR (compound annual growth rate) of 12.29% during the forecast period 2022 to 2030. This growth will concern especially mobile

robots, drones and automated solutions for logistics [17]. Moreover, the COVID-19 pandemic has accelerated the trend towards ecommerce undermining the logistics systems and, hence, highlighting the need of automation in warehouses, factories, distribution systems and logistics. Indeed, fleets of autonomous robots could be used to move boxes or packages from one location to another, improving the efficiency and accuracy of the logistics process.

In the last years, significant improvements have been made by the researchers into the field of multi-robot systems [18], with the idea that having more robots solving a complex task is more efficient and cheaper than having one large articulated robot [19]. In fact, these systems have gained relevance in several real world applications that require distributed capabilities provided by teams of cooperating robots [20]. There are several advantages and positive aspects in exploiting a multi-robot system [21], such as:

- Scalability: multi-robot systems can be easily scaled up or down depending on the task at hand. This allows for flexibility in the size and complexity of the task.
- Redundancy and robustness: with multiple robots working together, there is less reliance on any single robot. This means that if one robot fails, the others can still complete the task.
- Flexibility: a team of robots can be easily reconfigured to perform a wide range of tasks, making them highly flexible and adaptable.
- Performance: multi-robot systems can perform tasks that are beyond the capabilities of a single robot, such as tasks that are naturally distributed or require a large amount of force or require the operation of multiple agents. For example, simultaneous monitoring of a large environment is a task that require a distributed strategy that can be solved by a team of robots.

Together with the advantages, multi-robot systems bring several challenges in terms of methodologies to optimize the coordination and the cooperation between the agents. The group of robots, indeed, needs to coordinate to achieve a common goal, by only exploiting a communication network, local information sensed by the robots, information received by neighbors, dealing with faulty agents, communication errors, etc. Hence, one of the most challenging aspects of exploiting a multi-robot systems to solve a task is to define an optimal and efficient control strategy able to coordinate the team of robots facing the multitude of possible errors and difficulties that come from every element. Several control methodologies have been studied depending on the application, on the environment, whether in the real world or in an industrial scenario. However, lots of works found in literature are based on assumptions and idealties that make the deployment of the multi-robot system difficult in real scenarios.

Applications of multi-robot systems include precision agriculture [22, 23], emergency response and rescue [24, 25], environmental monitoring [26, 27], patrolling [28, 29], exploration [30, 31], search and rescue operations [32, 33], outdoor industrial operations such as fault diagnosis [18] and, one of the most relevant, logistics [34, 35]. Hence, it can be inferred the need of improving the current methodologies and strategies to meet the increasing demand of the multi-robot systems in several applications and, in particular, to make the multi-robot systems able to deal with real world scenarios and industrial scenarios. In the following section we introduce the studies described in this thesis together

with the aspects and the issues of the investigated methodologies.

1.1 Contribution and Thesis Outline

Multi-robot systems have been deeply studied in the last years to improve the hardware technology, the sensors, communication capabilities and, in particular, to improve the coordination methodologies to face various scenarios and environments. Moreover, recently, the main focus is to overcome the issues and limitations that are preventing the use of multi-robot systems in real world. Some of the challenges that arise when dealing with real life scenarios or industrial environments are [36, 37, 38, 39]:

- Limited sensing and perception: several methodologies found in the literature assume global knowledge of the environment. However, robots rely on sensors and on limited information that come from local perception of the surroundings.
- Environmental variability: real-world environments are usually highly variable, unknown and unpredictable. Initial assumptions of previous knowledge about the environment prevent the deployment of the team of robots in a real world scenario.
- Integration with existing systems: especially in industrial applications, the multi-robot system needs to be integrated in already existing technologies and architectures in order to function effectively.
- Hardware experimental validation: the studied methodologies needs to be rigorously tested, especially for industrial applications, on hardware platforms to overcome the issues that come from a real implementation.

These are some of the challenges and aspects related to multi-robot systems and investigated in this thesis. We studied and analyzed the methodologies proposed by the current state of the art and we worked to improve these methodologies to reduce the gap between simulations and real world implementations. In particular, we can distinguish between real life scenario applications, such as search and rescue or patrolling, and industrial applications, such as warehousing or transportation. In the first one, the aim is to have a robust control strategy able to deal with various scenarios, with various unpredictable events or environmental variability, with deployed robots characterised by limited capabilities. On the other hand, in industrial applications, the environment is generally known and constant over time, the robots can have global knowledge of the surroundings and, hence, the aim of the multi-robot system is to maximize the productivity, the throughput and to reduce the cost, minimizing the need for human labor in some tasks. In this thesis we investigated both of these aspects working on the challenges that arise when dealing with real implementations. In particular, we worked on control strategies able to use only local information from limited sensing capabilities, methodologies that consider the variability and unpredictability of the environment and control architectures capable of integrating with existing systems and dealing with unpredictable events. We finally managed to conduct several rigorous experiments in real environments and industrial scenarios with mobile robotic platforms, such as wheeled mobile robots [40] and unmanned aerial vehicles [41].

The dissertation is organized as follows.

In Chapter 2 we focuse on studying and developing control strategies for applications such as patrolling, exploration, search and rescue, where the aim is to have a multi-robot system capable of solving global tasks using local information and dealing with unknown and variable environments. The main representative classes of these strategies are discussed in [18]. Among them, a remarkable paradigm is represented by coverage control, which entails controlling a group of robots to be deployed in a given environment in order to cover it in an optimal manner. In Chapter 2 we propose novel approaches and variations of the coverage control problem.

In particular, in Section 2.3 we propose a distributed coverage control strategy that aims at deploying the robots in an open or closed environment, where different zones have different level of importance: the idea is that of having robots more densely deployed in higher importance areas. We present a novel coverage based control algorithm to coordinate a group of autonomous robots characterised by limited sensing capabilities and in absence of reliable communication. Most of the solutions presented in the literature rely on an *exact* Voronoi partitioning of the environment, whose computation requires complete knowledge of the environment and robots with unlimited sensing capabilities. To overcome these issues, we propose a definition and a characterization of the *limited* Voronoi partitioning, that allows robots to individually compute the partitioning of the environment based on information available within a limited sensing range. The limited Voronoi partitioning is then exploited to develop the novel control strategy that allows us to perform coverage (1) of an unknown environment, (2) in a limited sensing scenario, and (3) without requiring communication among the robots. The performance of the proposed method is experimentally evaluated on a large team of drones for outdoor operations (Section 2.4). We perform a deep and extensive analysis of both realistic simulations and real field tests with a fleet of quad-rotor drones in an unstructured environment as the Abu Dhabi desert. We note that this represents the first large-scale deployment of a team of drones performing fully decentralized and distributed coverage control in an outdoor environment. The study of this topic has produced the following publications:

- **On Coverage Control for Limited Range Multi-Robot Systems**
  Federico Pratissoli, Beatrice Capelli, Lorenzo Sabattini
  IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022.
- **On Limited-Range Coverage Control for Large-Scale Teams of Aerial Drones: Deployment and Study**
  Filippo Bertoncelli, Mehdi Belal, Dario Albani, Federico Pratissoli, Lorenzo Sabattini
  The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS), 2022

In section 2.5 we focus on the potentiality of the coverage based control strategy to explore an unknown environment. We present a novel methodology that allows the robots to explore previously unexplored areas exploiting only locally available information. The proposed methodology is based on the modelling of the already explored areas by time-varying probability density functions, which are used in the coverage control algorithm.

Exploiting this strategy, which allows to obtain an optimal configuration of the agents in the environment in order to maximize the covered area, we provide a solution to dynamically explore an unknown environment while continuously passing through it, in order to keep it constantly monitored. We, hence, combine the coverage based control methodology with a mixture of Gaussian distributions to model the areas of the environments already explored. The proposed approach performs the exploration and monitoring without a discretization of the environment (grid map or graph) and without any knowledge of the boundaries of the environment. We extensively tested the proposed control in both simulated and real environment experiments. We compared the proposed approach with other typical ones that do not require discretization or partial knowledge of the environment, such as random walk and potential field approaches. The study of this topic has produced the following publication:

- **Coverage Control for Exploration of Unknown Non-Convex Environments with Limited Range Multi-Robot Systems**
  Mattia Catellani, Federico Pratissoli, Filippo Bertoncelli, and Lorenzo Sabattini
  The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS), 2022

In Section 2.6 we address the problem of deploying a multi-robot team to optimally cover a domain where the density distribution, typically describing the interesting areas in the coverage problem, is initially unknown. We propose a coverage based control algorithm to coordinate a group of autonomous robots with the aim to simultaneously learn and estimate a spatial field over a domain. Hence, as in the previous case, the team of robots still need to explore the environment, but with the aim to estimate a spatial process or feature spread into the environment (such as pollution in the air or salinity in the ocean). In particular, we propose a novel control strategy that enables the team of robots to optimally learn and estimate a potentially time-varying spatial process and optimally cover it. We propose a novel approach to efficiently deal with the trade-off between the *exploration* phase to learn the spatial process and the *exploitation* phase to maximize the coverage of the estimated spatial process. The trade-off in question is time independent and determined by the degree of uncertainty and the accuracy of the spatial field estimation. Moreover, the trade-off is generally balanced on the exploration phase during the initial transient where the spatial field is completely unknown. The team of robots will move towards the spatial field if the estimation is sufficiently precise, and they will prioritize exploring the environment if the estimation is not of sufficient quality. Moreover, this control strategy allows the multi-robot system to deal with time varying spatial processes. We propose a novel methodology that continuously adapts the exploration-exploitation trade-off and the robots movements to the spatial process that changes over time. The control strategy exploits Gaussian Process Regression (GPR) to build a model of the monitored spatial process in the environment. Finally, we propose an approach that finds a solution to the computational issues of Gaussian Processes by continuously checking the validity of the training set: only the sampled data that provide sufficient information and effectively contribute to the process estimation are added to the training set. The performance of the proposed algorithm is evaluated in several rigorous simulations. The study of this topic has produced the following newly submitted paper:

- **Distributed Coverage Control for Time-Varying Spatial Processes**
  Federico Pratissoli, Lorenzo Sabattini, Amanda Prorok
  be submitted to Robotics: Science and Systems (RSS), 2023

In Chapter 3 we focus on the capacity of a multi-robot system to improve the efficiency and the productivity of an industrial scenario. In particular we studied how fleet of autonomous vehicles are exploited in automated factories and warehouses to increase the efficiency, flexibility and reduce costs by helping to automate the facility. Thus, the work discussed in Chapter 3 concerns the coordination and the traffic management of a group of Automated Guided Vehicles (AGVs) moving in a real industrial scenario. In particular, the current state-of-the-art does not consider the typical issues and non-idealities introduced by a real world application. In most of modern automated factories, the AGVs are guided through manually defined routes with ad-hoc strategies for coordination. This increases installation costs and reduces the flexibility when a change in the environment is required. The presented control architecture addresses the typical practical issues such as graph irregualrities (overlapping nodes, missing edges, etc.), a predefined roadmap (constrained by the plant layout), vehicle errors, dynamical obstacles, etc. The coordination strategy does not rely on ad-hoc assumptions or rules and exploits a novel deadlock avoidance policy to avoid collisions and congestions. We developed a novel hierarchical methodology based on a three-layer control architecture, which is described as follows: 1) the Top Layer (or Topological Layer) allows to model the traffic of vehicles among the different areas of the environment; 2) the Middle Layer allows the path planner to compute a traffic sensitive path for each vehicle; 3) the Bottom Layer (or Roadmap Layer) defines the final routes to be followed by each vehicle and coordinates the AGVs over time. The developed control algorithms were implemented and tested on a real use case in collaboration with a local company. The results show how the developed software was able to efficiently manage the studied automated factory. The study of this topic has produced the following works:

- **Hierarchical and Flexible Traffic Management of Multi-AGV Systems Applied to Industrial Environments**
  Federico Pratissoli, Nicola Battilani, Cesare Fantuzzi, Lorenzo Sabattini
  IEEE International Conference on Robotics and Automation (ICRA), 2021
- **Hierarchical Traffic Management of Multi-AGV Systems with Deadlock Prevention Applied to Industrial Environments**
  Federico Pratissoli, Riccardo Brugioni, Nicola Battilani, Lorenzo Sabattini
  second revision round at IEEE Transactions on Automation Science and Engineering (T-ASE)

Chapter 2

Coverage Based Control

In this chapter we describe one of the most important methodologies to coordinate a group of autonomous robots: the coverage based control algorithm. In the near future large groups of robots, such as autonomous vehicles, unmanned aerial vehicles, underwater robots, etc., will be used in real life applications to perform challenging tasks [42]. The many applications of a multi-robot system include precision agriculture, emergency response and rescue, environmental monitoring, exploration, surveillance, search and rescue operations and outdoor industrial operations such as fault diagnosis [18]. Among the several control strategies studied, a remarkable paradigm is represented by coverage control, which entails controlling a group of robots to be deployed in a given environment in order to cover it in an optimal manner. This control approach is related to the classic Lloyd algorithm [43] taking in consideration the capabilities and limitations of a mobile sensing network. The coverage control strategy is based on partitioning the environment in cells, one for each robot, and consists in the optimization of a cost function that represents the ability of the group of robots to sense a portion of the environment. The result of this optimization problem brings the robots to a configuration that maximizes the coverage of the environment [1]. Moreover, the solution of the optimization problem can be weighted by a probability density function. The role of this function is to model as a distribution the most interesting areas in the environment. The coverage control strategy is sensible to this probability density function and the configuration to which the robots converge aims to optimize the coverage of the environment depending on the locations of the interesting areas. This feature makes the coverage control methodology optimal for real world applications such as surveillance or search and rescue scenarios, where the robot team needs to be focused on particular areas of the environment.

In this chapter we present different control algorithms that evolve from the coverage based control theory and that are suitable for real world scenarios. We propose and describe an alternative partitioning algorithm that allows robots to individually compute the partitioning of the environment based on information available within a limited sensing range. We proposed a novel coverage control strategy, based on this partitioning, that allows the multi-robots system to perform coverage of an unknown environment, in a limited sensing scenario, and without requiring communication among the robots. To validate our approach, we perform an ablation study over a large set of simulations that

we evaluate with a set of proposed metrics. At the same time, the algorithm has been tested on the field, with real quad-copters, and with a set of proof of concept experiments deploying up to seven aerial units. Moreover, we present a novel methodology, built upon the coverage control, for the exploration and monitoring of unknown environments performed by a group of coordinated mobile robots. Finally, we propose an extension of the coverage control strategy for monitoring and accurate reconstruction of an unknown physical process in the environment. Therefore, the proposed methodology allows the multi-robot system to optimally learn and estimate the unknown process and uses this information to obtain a coverage of the environment influenced by a non-uniform density distribution.

2.1 Notation and Definitions

In this section we introduce the main definitions, notations and math background we use in this chapter. We denote by $\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$ the set of natural, real, real non-negative, and real positive numbers. For $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. Given $x \in \mathbb{R}^n$, let $\|x\|$ be the Euclidean norm.

Let $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ be a graph characterized by a set $\mathcal{U}$ of vertices and a set $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{U}$ of edges. Given an edge $(i, j) \in \mathcal{E}$, then the vertex i is a neighbor of the vertex j. Let $\mathcal{N}_{\mathcal{G}}(i)$ be the set of neighbors of the vertex i in $\mathcal{G}$. A graph $\mathcal{G}$ is said to be *undirected* if $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. Let $\mathbb{F}(\mathbb{R}^2)$ be the collection of finite point sets in $\mathbb{R}^2$. We can denote an element of $\mathbb{F}(\mathbb{R}^2)$ as $\mathcal{P} = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, where $\{p_1, \ldots, p_n\}$ are points in $\mathbb{R}^2$.

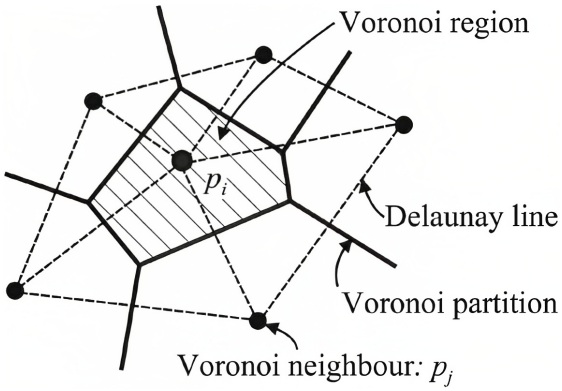


Figure 2.1: The Voronoi partitioning of a limited portion of environment around the robot i. The hatched area in the middle is the Voronoi cell V_i of the robot located in p_i.

The *Voronoi partitioning* can be defined on a polygonal environment in $\mathbb{R}^2$, Figure 2.1. In the rest of the chapter, we will use $Q \subset \mathbb{R}^2$ to denote such convex polygonal environment: it will be used, in particular, to denote the environment to be covered by the robots. An arbitrary point in Q is denoted by $q \in Q$.

Let then $\mathcal{P}$ be a set of n points $\{p_1, \ldots, p_n\}$ in Q. The *exact* Voronoi partitioning generated by $\mathcal{P}$ consists in the set $\mathcal{V}(\mathcal{P}) = \{V_1(\mathcal{P}), \ldots, V_n(\mathcal{P})\}$, where:

$$V_i(\mathcal{P}) = \{q \in Q : \|q - p_i\| \leq \|q - p_j\|, \forall p_j \in \mathcal{P}\}. \tag{2.1}$$

In the following, for the sake of brevity, we will use the notation V_i to refer to $V_i(\mathcal{P})$. Two agents are said to be *Voronoi neighbors* if $V_i \cap V_j \neq \emptyset$. We refer to [44] for an extensive discussion on the Voronoi diagrams.

Let us define the *proximity graph* as a graph $\mathcal{G}$ in which the edge set $\mathcal{E}_\mathcal{G}$ depends on the location of the vertices. In this work we consider graphs defined by points $\{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$. Hence, we can define a proximity graph function $\mathcal{G} : \mathbb{F}(\mathbb{R}^2) \to \mathbb{G}(\mathbb{R}^2)$ that associates to $\mathcal{P} \in \mathbb{F}(\mathbb{R}^2)$ an undirected graph with $\mathcal{P}$ being the set of vertices and $\mathcal{E}_\mathcal{G}(\mathcal{P})$ the set of edges, where $\mathcal{E}_\mathcal{G} : \mathbb{F}(\mathbb{R}^2) \to \mathbb{F}(\mathbb{R}^2 \times \mathbb{R}^2)$ has the property that $\mathcal{E}_\mathcal{G}(\mathcal{P}) \subseteq \mathcal{P} \times \mathcal{P} \setminus \text{diag}(\mathcal{P} \times \mathcal{P})$. Given $p \in \mathbb{R}^2$ and $R \in \mathbb{R}_{>0}$, we denote the closed and open ball in $\mathbb{R}^2$ centered at p with radius R with $\overline{B}(p, R) = \left\{q \in \mathbb{R}^2 | \|q - p\| \leq R\right\}$ and $B(p, R) = \left\{q \in \mathbb{R}^2 | \|q - p\| < R\right\}$, respectively. Throughout the chapter, we will use 1_S to represent the indicator function, which is defined as $1_S(q) = 1$, if $q \in S$, and $1_S(q) = 0$, if $q \notin S$.

The following proximity graphs are relevant to our discussion:

1. the *R-disk graph* $\mathcal{P} \mapsto \mathcal{G}_{disk}(\mathcal{P}, R) = (\mathcal{P}, \mathcal{E}_{\mathcal{G}_{disk}}(\mathcal{P}, R))$, where the edges are defined as:
$$\mathcal{E}_{\mathcal{G}_{disk}}(\mathcal{P}, R) = \quad \{(p_i, p_j) \in \mathcal{P} \times \mathcal{P} \setminus \text{diag}(\mathcal{P} \times \mathcal{P}) \mid \|p_i - p_j\| \leq R\};$$

2. the *Delaunay graph* $\mathcal{P} \mapsto \mathcal{G}_D(\mathcal{P}) = (\mathcal{P}, \mathcal{E}_{\mathcal{G}_D}(\mathcal{P}))$, where the edges are defined as:
$$\mathcal{E}_{\mathcal{G}_D}(\mathcal{P}) = \quad \{(p_i, p_j) \in \mathcal{P} \times \mathcal{P} \setminus \text{diag}(\mathcal{P} \times \mathcal{P}) \mid V_i(\mathcal{P}) \cap V_j(\mathcal{P}) \neq \emptyset\};$$

3. the *R-limited Delaunay graph* (see Fig. 2.2) $\mathcal{P} \mapsto \mathcal{G}_{LD}(\mathcal{P}, R) = (\mathcal{P}, \mathcal{E}_{\mathcal{G}_{LD}}(\mathcal{P}, R))$, where edges $(p_i, p_j) \in \mathcal{P} \times \mathcal{P} \setminus \text{diag}(\mathcal{P} \times \mathcal{P})$ are defined as:
$$(V_i(\mathcal{P}) \cap \overline{B}(p_i, R/2)) \cap (V_j(\mathcal{P}) \cap \overline{B}(p_j, R/2)) \neq \emptyset.$$

For each proximity graph $\mathcal{G} = \{\mathcal{G}_{disk}, \mathcal{G}_D, \mathcal{G}_{LD}\}$ we can define the respective set of neighbors of the point p as $\mathcal{N}_\mathcal{G}(p, \mathcal{P}) = \{q \in \mathcal{P} | (p, q) \in \mathcal{E}_\mathcal{G}(\mathcal{P} \cup \{p\})\}$.

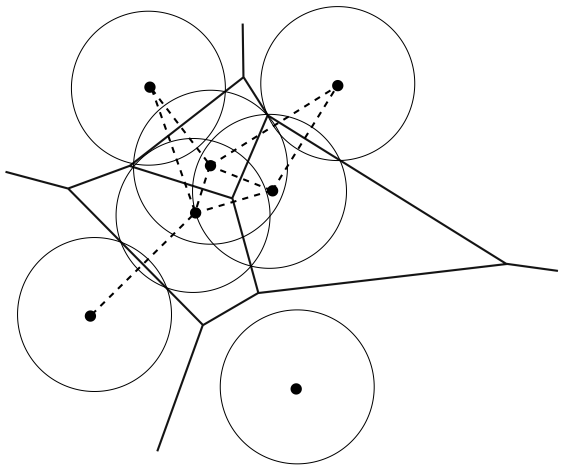


Figure 2.2: The Voronoi partitioning and the R-limited Delaunay graph (dashed line) with seven seeds. The circles represents the balls with radius $R/2$ around every seed.

2.2 Coverage Control - Background

As previously mentioned, the control strategies we propose in this chapter are built upon the theory behind the coverage based control. In this section we briefly summarize the math and the solution to the coverage problem as presented in [1], and we will refer to this approach as the *standard* coverage control. We remark that most of the works found in literature, which belong to the general category of coverage problem, are based on the standard coverage control strategy for a multi-robot system.

2.2.1 Robot Model

In this Chapter we present the control strategies studied and developed for generally a multi-robot system constituted by n robots that move in a 2-dimensional space. We assume each robot to be modeled as a single integrator system, whose position $p_i \in \mathbb{R}^2$ evolves according to

$$\dot{p}_i = u_i, \tag{2.2}$$

where $u_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \ldots, n$. The set of robots is represented by $\mathcal{P} = \{p_1, \ldots, p_n\}$. We would like to remark that, even though the single integrator is a very simplified model, it can still effectively be exploited to control real mobile robots: using a sufficiently good trajectory tracking controller, the single integrator model can be used to generate velocity references for widely used mobile robotic platforms, such as wheeled mobile robots [45], and unmanned aerial vehicles [46].

2.2.2 Coverage Control

In the coverage control problem a performance function is defined to be maximized in order to obtain the optimal coverage of the group of robots over Q. The performance function is chosen to model how reliable is the measurement, at point $q \in Q$, performed by robot i whose position is p_i, as a function of the distance $\|q - p_i\|$. Therefore, the performance function can be defined as a non-increasing differentiable function $f(\|q - p_i\|) : \mathbb{R}_{\geq 0} \to \mathbb{R}$. Moreover, an integrable probability density function $\phi : Q \to \mathbb{R}_{\geq 0}$ can be defined in order to characterize the portions in Q where an event of interest occurs. Namely, ϕ is used to capture the relative importance of determined areas in the environment. For example, $\phi(q_1) > \phi(q_2)$ implies that q_1 is more important than q_2, or that the area around q_1 has a higher probability of being an area of interest. Such probability density function may be known a priori to all the robots, or may be estimated, at run-time, based on local measurements, as discussed in [47]: hence, without loss of generality, we will hereafter assume the probability density function to be known to all the robots.

The following *optimization* function $H : Q^n \to \mathbb{R}$ can then be formulated:

$$\mathcal{H}(\mathcal{P}) = \int_Q \max_{i \in \{1,\ldots,n\}} f(\|q - p_i\|)\phi(q)dq, \tag{2.3}$$

with a better coverage corresponding to a higher value of the function. In order to solve the maximization problem, the environment is assumed to be partitioned into regions, where each node is in charge of covering its own region: the notion of Voronoi tessellation, introduced in (2.1) in section 2.1, becomes then fundamental. Thus, given an *exact* Voronoi partitioning $\mathcal{V}(P)$ of the environment Q in, so called, n Voronoi cells $\{V_1, \ldots, V_n\}$ (see Figures 2.1 and 2.3), the *optimization* function $H : Q \to \mathbb{R}$ can be formulated as follows:

$$\mathcal{H}(\mathcal{P}, \mathcal{V}) = \sum_{i=1}^{n} \int_{V_i} f(\|q - p_i\|)\phi(q)dq, \tag{2.4}$$

where each node is in charge of covering its own cell, and with a better coverage corresponding to a higher value of the function. In the literature, the performance function is usually chosen as: $f(x) = -x^2$. Therefore, the optimization function becomes:

$$\mathcal{H}_{\mathcal{V}}(\mathcal{P}) = -\sum_{i=1}^{n} \int_{V_i} \|q - p_i\|^2 \phi(q)dq = -\sum_{i=1}^{n} J_{V_i,p_i}, \tag{2.5}$$

where $\mathcal{H}_{\mathcal{V}}(\mathcal{P}) = \mathcal{H}(\mathcal{P}, \mathcal{V})$ and J_{V_i,p_i} indicates the polar moment of inertia of the region V_i about the point p_i. In order to solve the optimization problem, the gradient of the optimization function can be computed, obtaining:

$$\frac{\partial \mathcal{H}_{\mathcal{V}}}{\partial p_i}(\mathcal{P}) = 2M_{V_i}(C_{V_i} - p_i), \tag{2.6}$$

where M_{V_i} and C_{V_i} denote respectively the mass and the center of mass with respect to the density function ϕ of the Voronoi cell $V_i \subset Q$ of the robot i in position p_i. Therefore, the mass and centroid can be computed as follows:

$$M_{V_i} = \int_{V_i} \phi(q)dq, \; C_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} q\phi(q)dq. \tag{2.7}$$

For more details we refer the reader to [1, 47].

According to (2.6), the configuration of points $\mathcal{P}$ which maximizes the optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$ coincides with the *centroids* of the respective Voronoi cells. In other words, the solution to the maximization problem is achieved when each agent is located at the centroid of its Voronoi region, such that $p_i = C_{V_i}, \; \forall i$. Such configurations are called *centroidal Voronoi configurations*, see [48]. $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$ can be used as a Lyapunov-like function and we can design a control law that drives the group of robots to a centroidal Voronoi configuration, locally maximizing the coverage in the environment. In particular, the control input for each robot can be computed according to the Lloyd algorithm [1], as follows:

$$u_i = k(C_{V_i} - p_i), \tag{2.8}$$

where $k \in \mathbb{R}_{>0}$ is a proportional gain (see Fig.2.3).

The exact Voronoi partitioning $\mathcal{V}(\mathcal{P})$ of the region Q is assumed to be continuously updated. Thus, it is assumed that the region Q is known or that each robot is able to measure it. The network of the group of robots is defined by the Delaunay proximity

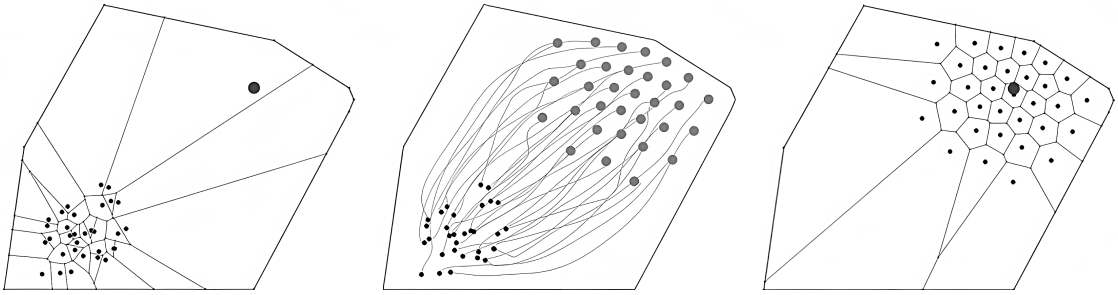


Figure 2.3: This figure is summarizing the coverage control methodology from [1]. The control strategy is applied to 32 agents deployed in a convex polygonal environment, with a Gaussian density distribution centered in the top right area of the environment. The left figure illustrates the initial robots locations and the respective Voronoi diagrams. The middle figure illustrates the robots trajectories. The right figure illustrates the finale robots locations and the respective Voronoi diagrams.

graph $\mathcal{G}_D(\mathcal{P})$. To compute the i^{th} component of the $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$ function, it is necessary for the robot i to know its own position and the positions of its neighbors over the graph $\mathcal{G}_D(\mathcal{P})$. Usually the positions of all the robots are assumed to be known or measurable. A modified optimization function is then introduced in [49], where coverage is achieved considering the R-limited Delaunay graph $\mathcal{G}_{LD}(\mathcal{P}, R)$, where two robots are neighbors if their respective Voronoi cells are neighboring, *and* if their distance is smaller than the communication range R. It is worth noting, however, that this graph is built upon the exact Voronoi partitioning $\mathcal{V}(\mathcal{P})$, whose computation requires, as already discussed, full knowledge of the region Q and at least of the Delaunay neighbors. In the next section we will propose an alternative partitioning approach, that overcomes these issues, requiring only local knowledge of the portion of the region Q and neighbors that can be directly observed by each robot.

2.3 Coverage Control for Limited Range Multi-Robot Systems

In this section we propose a distributed coverage control strategy that aims to perform coverage of an unknown environment, in a limited sensing scenario without requiring communication among the robots. The proposed methodology is based on an alternative Voronoi partitioning of the environment and allows to get rid of the assumptions, generally found in literature, that prevent us to use this control in real world scenarios. In fact, most of the solutions presented in the literature rely on an *exact* Voronoi partitioning, whose computation requires complete knowledge of the environment to be covered. This can be achieved only by robots with unlimited sensing capabilities, or through a reliable communication network among robots in a limited sensing scenario. To overcome these limitations or assumptions, we present a distributed control strategy to cover an unknown environment with a group of robots with limited sensing capabilities and in the absence of reliable communication. The control law is based on a *limited* Voronoi partitioning of the sensing area. We propose a definition and a characterization of the limited Voronoi partitioning and we demonstrate that the group of robots can optimally cover the environment using only information that is locally detected (without communi-

cation). Moreover, we introduce a novel methodology to deal with numerical issues that typically come from large scale experiments on hardware platform when using Gaussian-type probability density distribution in the coverage based control algorithm. Finally, the proposed method is validated by means of simulations and experiments carried out on a group of mobile robots.

2.3.1 Literature Review

Various strategies have been introduced for implementing coverage control with networked mobile robots. In particular, most relevant to this study are the results reported in [1, 49]. In these works, the authors presented decentralized coordination algorithms for groups of mobile agents based on Voronoi diagrams [48] and proximity graphs [50]. This approach gives a simple solution that guarantees the convergence of the networked robots to a configuration that maximizes the coverage of the environment. This strategy is based on a Voronoi partitioning of the whole environment, which is assumed to be known [51] or measurable by the robots [47]. For a successful implementation of this control law each robot should be able to compute the respective Voronoi diagram autonomously. In most of the works found in literature about the Coverage problem the Voronoi diagram is assumed to be given and is considered over the entire environment. In [47] the authors re-interpreted the problem in a non-probabilistic framework in which the robots can measure the value and the gradient of a sensory function. However, the authors assumed that a Voronoi tessellation of the environment is provided to the robots. There exist several algorithms to construct an *exact* Voronoi diagram by assuming that each robot can obtain the position of all the others [52] or by means of communication among robots and broadcasting of messages [53, 54, 55]. Moreover, these algorithms generally assume each robot knows the boundary of the environment.

An alternative tool that can be exploited in the coverage based control problems is the Power diagram [56]. Power diagrams can be used in a Voronoi based control for identifying collision free multi-robot configurations [57] or to generate a distributed control scheme for heterogeneous robots tasked to track an unknown number of agents [58]. In [59] the authors introduced a novel partitioning of the environment based on Power diagrams. The control strategy is able to deal with non-convex domains of interest and heterogeneity of the network (in terms of nodes' sensing range). However, a reliable communication network between nodes has to be ensured.

However, in most practical scenarios, robots do not necessarily have information about the environment and the location of all other robots and may not have the possibility to rely on a communication network: hence, the Voronoi partitioning needs to be studied in a sensor range constrained scenario. Thus, instead of an exact Voronoi cell we consider a *limited* Voronoi cell, which is a partitioning of the area within the sensing range of the robot. A distributed methodology was presented in [1, 49] to compute the exact Voronoi diagram for a limited sensing network, which is based on the algorithm presented in [60]. The exact Voronoi cell of each robot can be gradually refined by computing the partitioning within an adjustable sensing range and incrementally increasing the

sensing range. The Adjust-Sensing-Radius algorithm has become the standard one in the field of distributed coverage control to solve the problem of the Voronoi partitioning. However, the objective of this algorithm is to determine the smallest distance R_i for agent i which provides enough information to compute the exact Voronoi cell V_i [1]. The algorithm requires sensors with controllable sensing range in order to determine the relative location of each Voronoi neighbor (more details were given in Section 2.2). In practical applications, the limited sensing capability of the robot is a constraint and rarely the sensing radius can be adjusted to detect sufficient information to compute the exact Voronoi diagram. An algorithm was proposed in [52] to efficiently compute the Voronoi cell. However, it is based on a reliable communication among the robots, which in real scenarios can not always be guaranteed (such as in underwater environments [61]). We consider a control strategy constrained by the information detected by the sensor (a communication network is not needed) and, thus, based on the Voronoi partitioning of the area within the sensing range (limited Voronoi diagram). An algorithm studied to perform efficiently this computation is described in [62].

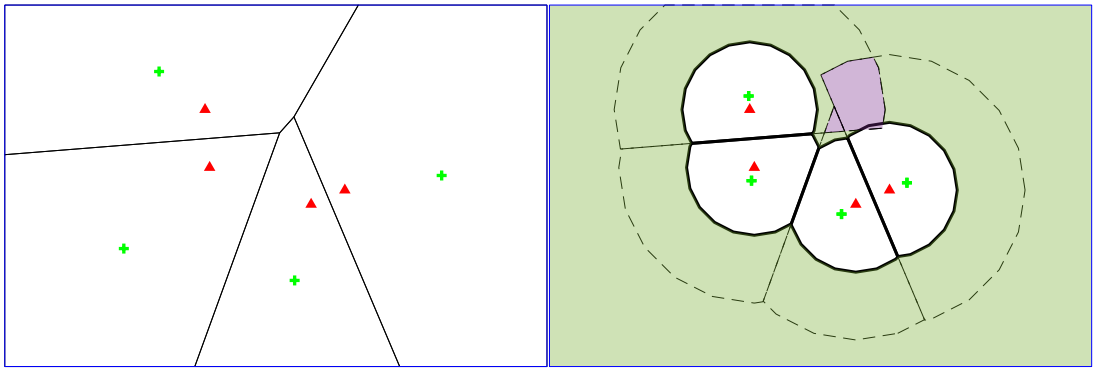
The effect of the characteristics of robot sensing capabilities on the performance of coverage control algorithms have been recently addressed in the literature. Various studies focused on robots sensing capabilities have been proposed. In [63, 64, 65], the authors introduced agents heterogeneity into the standard coverage problem in order to consider the different sensing modalities. Differently, in [66] the authors focused on the anisotropic aspect of the limited sensing range. They proposed an alternative coverage problem based on a definition of a new proximity graph.

It is worth noting that all the aforementioned works are based on an exact Voronoi partitioning of the environment: hence, the control algorithm requires the location of the Voronoi neighbors and the knowledge of the boundary of the environment to compute each robot's Voronoi cell. The control strategy we propose can be applied to a group of robots deployed in an unknown environment, since the only information required by the control law can be locally acquired by on board sensors. Moreover, we would like to remark that only a few works can be found in literature in which coverage control is implemented and tested on real robots: an example is provided by [47], where the authors tested the proposed strategy on a swarm of mobile robots. Similarly, in this work we validate the proposed methodology in real experiments on ground robots.

2.3.2 Problem Statement

As discussed in Section 2.2.1, we consider a multi-robot system constituted by n holnomic robots that move in a 2-dimensional space. It is worth noting that this study can be theoretically extended to a 3-dimensional scenario, with a 3-dimensional version of the Voronoi partitioning, but at the cost of increasing considerably the computational complexity. We assume each robot to be modeled as a single integrator system (according to (2.2)), whose position $p_i \in \mathbb{R}^2$ evolves according to $\dot{p}_i = u_i$, where $u_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \ldots, n$. The set of robots is represented by $\mathcal{P} = \{p_1, \ldots, p_n\}$.

We consider the following assumptions:



(a) *Exact* Voronoi $\mathcal{V}$. **(b)** *Limited* Voronoi $\mathcal{V}^r$.

Figure 2.4: Comparison between *exact* and *limited* Voronoi. The environment Q is reported in *blue*, the robots are the *red* triangles, and the centroid of the single Voronoi cell are the *green* crosses, the single Voronoi cell are delimited with *solid black* lines. Fig. 2.4b also reports in *dashed black* lines the wrong partitioning built considering the whole sensing range R, and in *purple* the overlapping areas.

1. *Convex unknown environment*: the environment to be covered by the multi-robot system is supposed to be represented by a convex polytope Q, which is not known a priori by the robots.

2. *Limited sensing capabilities*: each robot is able to measure the position of neighboring robots and objects, and to detect the boundaries of the environment Q, within its limited sensing range (defined by a ball with radius $R \in \mathbb{R}_{>0}$);

3. *No communication capabilities*: robots do not exchange information among each other.

Based on these assumptions, each robot computes its control inputs based only on directly measurable information, within the sensing range. Hence, the information exchange infrastructure can be modeled as the R-disk graph $\mathcal{G}_{disk}(\mathcal{P}, R))$.

The problem addressed in this study is then formalized as follows:

Problem. *Define a distributed control strategy that allows a multi-robot system with limited sensing capabilities to perform coverage of an unknown convex environment Q without reliable communication.*

2.3.3 Partitioning of The Environment - Limited Voronoi Diagram

In this section, we introduce a novel methodology to obtain a partitioning of the environment that, differently from [1, 49], considers the limited sensing range of the agents as a constraint, and thus exploits only the information directly detectable by the robots: we will hereafter refer to this concept as the *limited* Voronoi partitioning.

To achieve this, for each robot with sensing range R, and position p_i we consider:

1. the set of neighbors $\mathcal{N}_{\mathcal{G}_{disk}(R)}(p_i, \mathcal{P})$ on the R-disk graph,

2. the set of points $P_i = \{p_i \cup \mathcal{N}_{\mathcal{G}_{disk}(R)}(p_i, \mathcal{P})\}$,

3. a ball with radius equal to half of the sensing range, namely

$$\overline{B}(p_i, r), \quad \text{with} \quad r = \frac{R}{2}. \tag{2.9}$$

Every robots is equipped with a sensing device such as infrared or laser that can detect objects, neighbors and the boundaries of the environment Q, within sensing range R. The limited sensed area can be represented by the closed ball $\overline{B}(p_i, R)$. The *limited* Voronoi partitioning generated by $\mathcal{P}$ is then defined as $\mathcal{V}^r(\mathcal{P}) = \{V_1^r(P_1), \ldots, V_n^r(P_n)\}$, where:

$$V_i^r(P_i) = \{q \in \overline{B}_{\cap Q}(p_i, r) \mid \|q - p_i\| \leq \|q - p_j\|, \forall p_j \in \mathcal{N}_{\mathcal{G}_{disk}(R)}(p_i, \mathcal{P})\}, \tag{2.10}$$

where $\overline{B}_{\cap Q}(p_i, r) = \{Q \cap \overline{B}(p_i, r)\}$ is the intersection between the environment and the ball of radius r for robot i, as introduced in (2.9). Fig. 2.4 reports the difference between the exact Voronoi partitioning of Q, performed with the knowledge of the agents' positions, and the limited Voronoi partitioning introduces in this work, which is computed with the information directly sensed by the robot. We can then introduce the *r-limited Voronoi graph* $\mathcal{P} \mapsto \mathcal{G}_{LV}(\mathcal{P}, r) = (\mathcal{P}, \mathcal{E}_{\mathcal{G}_{LV}}(\mathcal{P}, r))$, where edges are defined as:

$$\mathcal{E}_{\mathcal{G}_{LV}}(\mathcal{P}, r) = \quad \{(p_i, p_j) \in \mathcal{P} \times \mathcal{P} \backslash \text{diag}(\mathcal{P} \times \mathcal{P}) \mid V_i^r(P_i) \cap V_j^r(P_j) \neq \emptyset\}.$$

We will hereafter show that the introduction of the balls of radius r in (2.9) is instrumental for the definition of a valid partitioning of the environment, that can be computed based only on locally available information, i.e., collecting information over the graph $\mathcal{G}_{disk}(R)$.

Lemma 1. *The limited Voronoi partitioning defined in* (2.10) *is equivalent to the exact Voronoi partitioning limited to the portion of the environment Q that is covered by the union of the areas within the robots' balls of radius r introduced in* (2.9)*, i.e.,* $\cup_{i=1}^n \overline{B}_{\cap Q}(p_i, r)$

Proof. The R-limited Delaunay graph $\mathcal{G}_{LD}(R)$ is the proximity graph corresponding to the exact Voronoi partitioning limited to the robots' balls of radius r, namely

$$V_i(P) \cap \overline{B}(p_i, r)$$

According to [50, Prop. 2.9], $\mathcal{G}_{LD}(R)$ is spatially distributed over the graph $\mathcal{G}_{disk}(R)$. This implies that the R-disk graph encodes sufficient information to compute the R-limited Delaunay graph $\mathcal{G}_{LD}(R)$. Hence, it is possible to conclude that

$$V_i^r(P_i) = V_i(P) \cap \overline{B}(p_i, r). \tag{2.11}$$

which proves the statement. □

Remark 1. *Given Lemma 1, $\mathcal{G}_{LV}(r)$ is spatially distributed over $\mathcal{G}_{disk}(R)$.*

It is possible to show that the proposed definition of limited Voronoi diagram is well posed: namely, for a sufficiently large sensing range, or for a sufficiently large number of robots, the limited Voronoi diagram corresponds to the exact Voronoi diagram.

Lemma 2. *Consider the definition of limited Voronoi diagram given in* (2.10)*. For any n, a value $\overline{r}$ exists such that, for $r \geq \overline{r}$, then $\mathcal{V}^r(P) = \mathcal{V}(P)$.*

Proof. This Lemma can be proven showing that it is always possible to find $\overline{r}$ such that $Q \subseteq \cup_{i=1}^{n} \overline{B}(p_i, r)$: in this case $\mathcal{V}^r(P) = \mathcal{V}(P)$.

Choosing $\overline{r}$ as the maximum distance between two points in Q, then $Q \subseteq \overline{B}(p_i, r)$, which implies $Q \subseteq \cup_{i=1}^{n} \overline{B}(p_i, r)$. □

Lemma 3. *Consider the definition of limited Voronoi diagram given in* (2.10)*. For any r, a value $\overline{n}$ exists such that, for $n \geq \overline{n}$, then $\mathcal{V}^r(P) = \mathcal{V}(P)$.*

Proof. As for Lemma 2, it is always possible to find $\overline{n}$ such that $Q \subseteq \cup_{i=1}^{n} \overline{B}(p_i, r)$: in this case $\mathcal{V}^r(P) = \mathcal{V}(P)$. For a given value r, $\overline{n}$ can be found considering the minimum number of balls whose union covers the entire area Q. □

Remark 2. *It is worth noting that considering radius r is fundamental for achieving a valid environment partitioning. In fact, if we considered* (2.10) *with R instead of r, such that $q \in \overline{B}_{\cap Q}(p_i, R)$, the Voronoi partitioning generated would not be a valid partitioning. This is due to the fact that $\mathcal{G}_D \cap \mathcal{G}_{disk}(R)$ is not spatially distributed over $\mathcal{G}_{disk}(R)$ [50, Sec. 2.2.1], as opposed to $\mathcal{G}_{LV}(r)$ (see Remark 1). An example is shown in Fig. 2.4b, where dashed lines show the invalid partitioning generated considering R, and overlapping areas are shown in purple. In fact, a valid partitioning of the environment is a collection of subsets of $\mathbb{R}^2$ that have disjoint interiors.*

In addition to what stated in Remark 2, another reason for using r instead of the whole sensing radius R is to avoid instability when the robot set of neighbours changes: in fact the area of the partition changes in an abrupt manner when a neighbor is added or removed from $\mathcal{N}_{\mathcal{G}_{disk}}(p_i, \mathcal{P})$. This issue is illustrated in Fig. 2.5, where the change in the neighbours set from Fig. 2.5a to Fig. 2.5b is relevant if compared with the one from Fig. 2.5c to Fig. 2.5d. This sudden change would cause a chattering movement of the centroid of this partition, which would generate undesirable oscillations in the control law (2.8). Instead, if we consider half of the sensing radius r as in the proposed limited Voronoi diagram $\mathcal{V}^r$ (eq. (2.10)) we obtain a progressive change of the area, and consequently of the position of the centroid.

Remark 3. *The proposed limited Voronoi partitioning allows us to take full advantage of the limited sensing capabilities of the robots, i.e., using all the information over $\mathcal{G}_{disk}(R)$, and to obtain a valid partitioning of the environment computing it over $\overline{B}(p_i, r)$ (see Remark 2).*

2.3.4 Proposed Control Strategy

In this section, building upon the concept of limited Voronoi partitioning introduced in Section 2.3.3, we propose a control strategy to solve Problem 2.3.2.

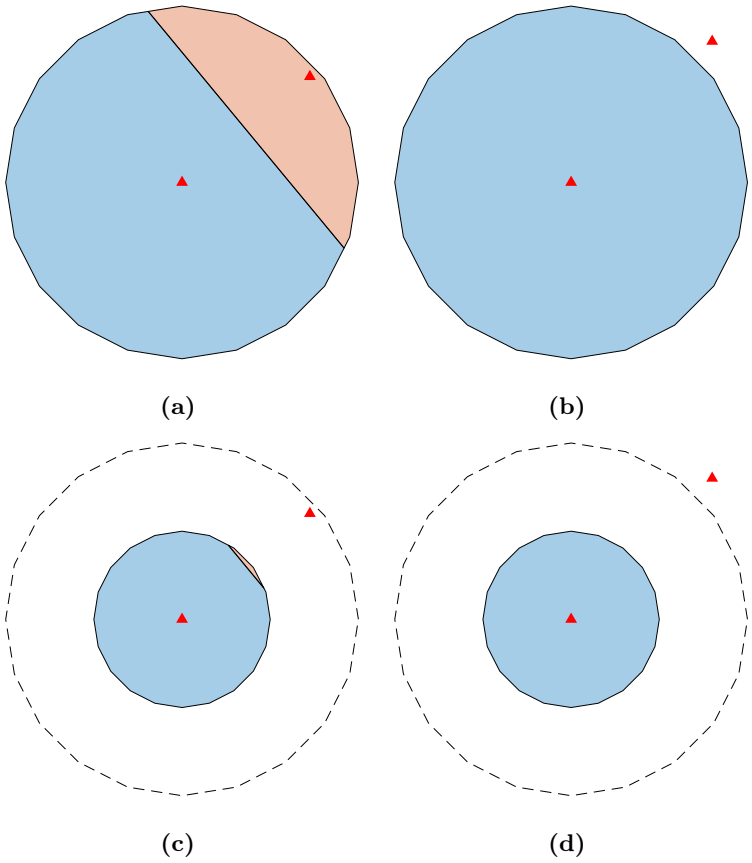


Figure 2.5: In the figures the robots are represented by *triangles*. The *blue* area represents the Voronoi partitioning in the sensing range area around the i-th robot with respect to which we calculate the local Voronoi. The *red* area represents the remaining partitioned area in the sensing range associated with a neighbour (another robot within the sensing range R). In particular, the images show the local Voronoi partition when a neighbor is removed from $\mathcal{N}_{\mathcal{G}_{disk}}(p_i, \mathcal{P})$ (in the left images the robot has one neighbor, while in the right ones the robot has no neighbours). Fig. 2.5a and Fig. 2.5b report the local Voronoi partitioning result when computed over the whole sensing range R. Fig. 2.5c and Fig. 2.5d report the local Voronoi partitioning when computed over half of the sensing range. Hence, the resulting cells are intersected with $\overline{B}(p_i, \frac{R}{2})$. The *dotted* line represents the sensing range.

In particular, we propose the following control law:

$$u_i = k(C_{V_i^r} - p_i), \tag{2.12}$$

where $k \in \mathbb{R}_{>0}$ is a proportional gain and $C_{V_i^r}$ is the centroid of the cell V_i^r defined by the r-limited Voronoi diagram of robot i. We will hereafter show that this control strategy leads to optimally deploying the limited sensing range robots within the unknown environment Q. For this purpose, we consider the environment Q to be partitioned according to an exact Voronoi partitioning $\mathcal{V}(\mathcal{P})$, as defined in (2.1). Hence, inspired by [49], we consider the following performance function:

$$f^r(x) = -x^2 1_{[0,r]}(x) - r^2 1_{(r,\infty)}(x), \tag{2.13}$$

which, as we will show below, leads to optimally deploying robots in their corresponding Voronoi cell considering the limited sensing capabilities. The performance function is intuitively chosen to be a non-increasing, piece-wise differentiable and continuous function

which provides an indication of the robot sensor performance [1], i.e. the further the point of measure is from the robot and the worse the measure will be (due to noise, etc.). Hence, we consider the standard approach to the problem as $f(x) = -x^2$ for values within the range r, i.e., inside the ball $\overline{B}(p_i, r)$ (first term of (2.13)). Also, we consider a constant function otherwise, as $f(x) = -r^2$ for values outside the range r, i.e., outside the ball $\overline{B}(p_i, r)$, to have a continuous function (second term of (2.13)).

Following the definition in (2.4), the optimization function can then be derived as

$$\mathcal{H}^r_{\mathcal{V}}(\mathcal{P}) = -\sum_{i=1}^{n} \int_{V_i(\mathcal{P})} \|q - p_i\|^2 1_{[0,r]} \phi(q) dq - r^2 \sum_{i=1}^{n} \int_{V_i(\mathcal{P})} 1_{(r,\infty)} \phi(q) dq. \tag{2.14}$$

Theorem 1. *Consider a multi-robot system composed of n robots with limited sensing range whose dynamics evolves according to* (2.2)*. Then, the control law* (2.12) *leads to maximize the optimization function* (2.14)*.*

Proof. We will show that control law (2.12) implements a gradient ascent of the optimization function (2.14). Namely, we will show that the time derivative of the optimization function, that is

$$\dot{\mathcal{H}}^r_{\mathcal{V}}(\mathcal{P}) = \sum_{i=1}^{n} \frac{\partial \mathcal{H}^r_{\mathcal{V}}(\mathcal{P})}{\partial p_i}^T \dot{p}_i, \tag{2.15}$$

is non-negative and assumes a value of zero exclusively when the robots are positioned at the centroids of their respective Voronoi cells. This specific configuration corresponds to the local maxima of the optimization function.

For this purpose, consider the definition of the optimization function in (2.14). Following the same computation as in (2.5), the first term can be rewritten by means of the polar moment of inertia. Furthermore, the second term represents the area, weighted with function ϕ, of the portion of Q that is not covered by the union of the balls $\overline{B}(p_i, r)$ (in *green* in Fig. 2.4b). Hence, we can rewrite (2.14) as:

$$\mathcal{H}^r_{\mathcal{V}}(\mathcal{P}) = -\sum_{i=1}^{n} J_{V_i(P)\cap\overline{B}(p_i,r),p_i} - r^2 Area_\phi(Q \setminus \cup_{i=1}^{n} B(p_i, r)), \tag{2.16}$$

with $Area_\phi(S) = \int_S \phi(q)dq$, $S \subset Q$.

Given [49, Thm. 2.2], considering the choice of the performance function (2.13), we can compute the partial derivatives of the optimization function $\mathcal{H}^r_{\mathcal{V}}(\mathcal{P})$ as follows:

$$\begin{aligned} \frac{\partial \mathcal{H}^r_{\mathcal{V}}(\mathcal{P})}{\partial p_i} &= 2 \int_{V_i(P)\cap\overline{B}(p_i,r)} (q - p_i)\phi(q) dq \\ &= 2M_{V_i\cap\overline{B}(p_i,r)}(C_{V_i\cap\overline{B}(p_i,r)} - p_i), \end{aligned} \tag{2.17}$$

where $M_{V_i\cap\overline{B}(p_i,r)}$ and $C_{V_i\cap\overline{B}(p_i,r)}$ are respectively the mass and the centroid of the cell given by the intersection of $\overline{B}(p_i, r)$ with the exact Voronoi partitioning of the environment.

Considering the control law (2.12) and the dynamics (2.2), the time derivative of the

optimization function (2.15) can be written as follows:

$$\dot{\mathcal{H}}^r_{\mathcal{V}}(\mathcal{P}) = \sum_{i=1}^{n} \left[2M_{V_i \cap \overline{B}(p_i,r)} (C_{V_i \cap \overline{B}(p_i,r)} - p_i) \right]^T k (C_{V_i^r} - p_i). \tag{2.18}$$

Defining $\kappa_i = k\, 2M_{V_i \cap \overline{B}(p_i,r)} > 0$, we can rewrite (2.18) as follows:

$$\dot{\mathcal{H}}^r_{\mathcal{V}}(\mathcal{P}) = \sum_{i=1}^{n} \kappa_i (C_{V_i \cap \overline{B}(p_i,r)} - p_i)^T (C_{V_i^r} - p_i). \tag{2.19}$$

Consider now the results of Lemma 1, we obtain the following:

$$C_{V_i^r} = C_{V_i \cap \overline{B}(p_i,r)}, \quad \forall i = 1, \ldots, n. \tag{2.20}$$

Therefore, from (2.19) and (2.20), we can state that

$$\dot{\mathcal{H}}^r_{\mathcal{V}}(\mathcal{P}) = \sum_{i=1}^{n} \kappa_i \left\| C_{V_i \cap \overline{B}(p_i,r)} - p_i \right\|^2 \geq 0, \tag{2.21}$$

which proves the statement. □

According to Theorem 1, we can conclude that the proposed control law (2.12), which can be computed by each robot based only on directly sensed information, leads to optimally covering the environment.

During the execution of the control algorithm, every robot continuously updates the limited Voronoi diagram and moves towards the centroid of the respective Voronoi cell. The centroid computation and, hence, the control law (2.12), depend on the neighbors detected and the environment sensed. In other words, given a constant $\phi(q)$, every robot will tend to move far from other robots and from the environment boundary, while maximizing the coverage of the area.

These results are supported by several simulations we conducted in MATLAB, in Gazebo and several experiments conducted on ground robots described in Section 2.3.6.

2.3.5 Numerical Issues - Variance Varying Control

In the proposed control strategy we consider a probability density function $\phi : Q \to \mathbb{R}_{>0}$ in order to measure the position in Q where an event of interest occurs. Following the control law presented in (2.12), each robot has to compute the mass and the centroid of a region (defined by the Voronoi diagram) weighted by the density function. We assume ϕ has the form of a Gaussian distribution[1]. Gaussian functions are, indeed, characterized by various useful properties making easier the computation of the weighted mass and centroid. In [67], for example, the authors propose a line integration method to compute mass and centroid of a Voronoi cell, which alleviates the computational issues compared with the spatial discretization method.

[1]It is worth noting that the proposed concepts can be easily extended to general probability density functions, approximated as combination of Gaussian distributions.

In order to keep the notation simple, we will first consider a one-dimensional density function $\phi : \mathbb{R} \to \mathbb{R}_{\geq 0}$ with the form:

$$\phi(x) = exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.22}$$

where $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}_{>0}$ are the mean and the variance of the density function. The one-dimensional case can be easily extended to the two-dimensional one, as will be shown at the end of the section. The Gaussian function is characterized by a high gradient, such that for points far from the mean value the Gaussian function assumes almost endlessly small values. This feature typically causes numerical issues for experiments or simulations in large environments. For example, we consider a standard Gaussian distribution $\phi : \mathbb{R} \to \mathbb{R}_{>0}$ ($\mu = 0$ and $\sigma = 1$). We have that $\phi(x = 38) \approx 2.7e^{-314}$, while the smallest positive double precision floating-point number in IEEE standard is $\approx 2.2e^{308}$: this makes the portion of the area for $x \geq 38$ unusable. In a coverage problem, the σ parameter determines how scattered the group of robots should cover the area around the μ value. However, if the robot is located too far from the mean value μ, it faces numerical issues and it might not be able to compute the control input.

We propose a robot control characterised by a self adapting variance whose value depends on the position of the robot relatively to μ. We define as $\phi_i(x)$ the density function processed by the robot i, with σ_i and μ_i the variance and the mean, respectively. The robot's controller artificially increases the variance of the processed density distribution ϕ_i depending on the robot position and on the computational capacity of the robot's CPU (e.g., float or double variables). The variance value converges to the desired one as the robot moves close to the mean value. Thus, we have that $\mu_i = \mu$ since the controller acts only on the variance.

Given a generic Gaussian distribution $\phi(x)$ with variance σ and mean μ, we can define the corresponding standard Gaussian distribution $\phi(z)$ where $z = \left|\frac{x-\mu}{\sigma}\right|$. In this way we can define a single constant parameter z_{max} above which the Gaussian distribution gives numerical issues. This parameter depends on the computational capacity of the hardware platform, e.g., $z_{max} = 38$ for double variables or $z_{max} = 13$ for float variables. Thus, we can easily determine if a value x_i brings to numerical issues and we can change the variance value of the processed Gaussian function $\phi_i(x_i)$ consequently. The processed variance σ_i becomes a function of x_i and assumes the form:

$$\sigma_i(x_i) = \begin{cases} \sigma & \text{if } z_i \leq z_{max} \\ \left|\frac{x_i-\mu}{z_{max}}\right| & \text{otherwise} \end{cases} \tag{2.23}$$

where $z_{max} \in \mathbb{R}_{>0}$ is a parameter that depends on the CPU capacity. Fig. 2.6 graphically represents (2.23), which describes the proportional relation between robot position and variance. As the robot approaches the mean value μ, the variance σ_i decreases until it reaches the value of σ, i.e., the variance of density function ϕ over the environment.

The solution presented in (2.23) applies to one-dimensional density function. However,

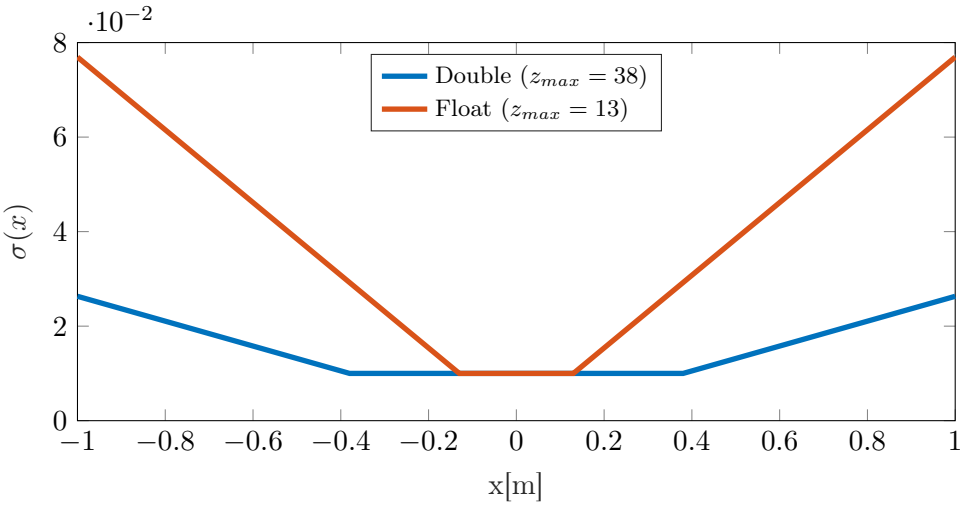
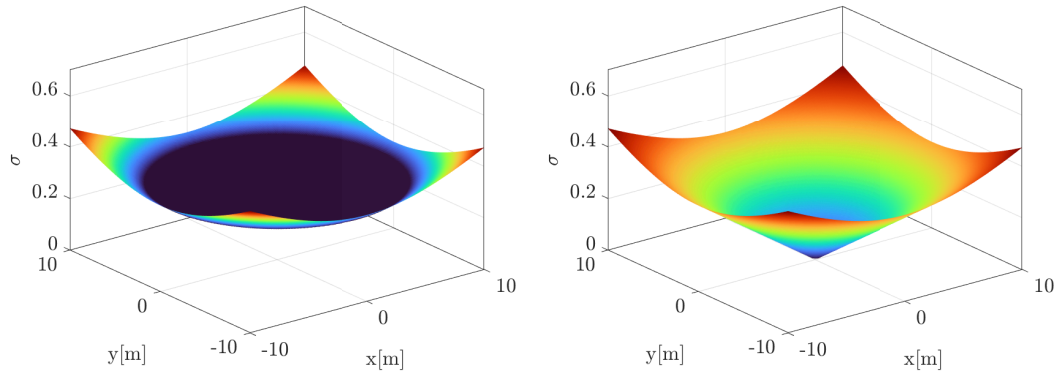


Figure 2.6: One-dimensional representation of the varying-variance with different values of z corresponding to the principal numerical representation ($\sigma = 0.01$, $\mu = 0$).



(a) $\sigma = 0.3$ **(b)** $\sigma = 0.01$

Figure 2.7: Examples of two surfaces calculated with (2.23) ($z_{max} = 30$, $\mu = (0, 0)$).

the same idea can be easily extended to a two-dimensional density function, as we have in the coverage problem. Given the density function $\phi : Q \to \mathbb{R}_{>0}$, where $Q \subset \mathbb{R}^2$, we can define:

$$\sigma_i(p_i) = \begin{cases} \sigma & \text{if } z_{i,Q} \leq z_{max} \\ \dfrac{\|p_i - \mu\|}{z_{max}} & \text{otherwise} \end{cases} \tag{2.24}$$

where:

$$z_{i,Q} = \frac{\|p_i - \mu\|}{\sigma} \tag{2.25}$$

Fig. 2.7 reports two examples of the surfaces that map the position p_i to the corresponding $\sigma_i(p_i)$.

2.3.6 Experimental Validations

In this section we report the simulations and the real experiments we carried out to verify the proposed algorithm.

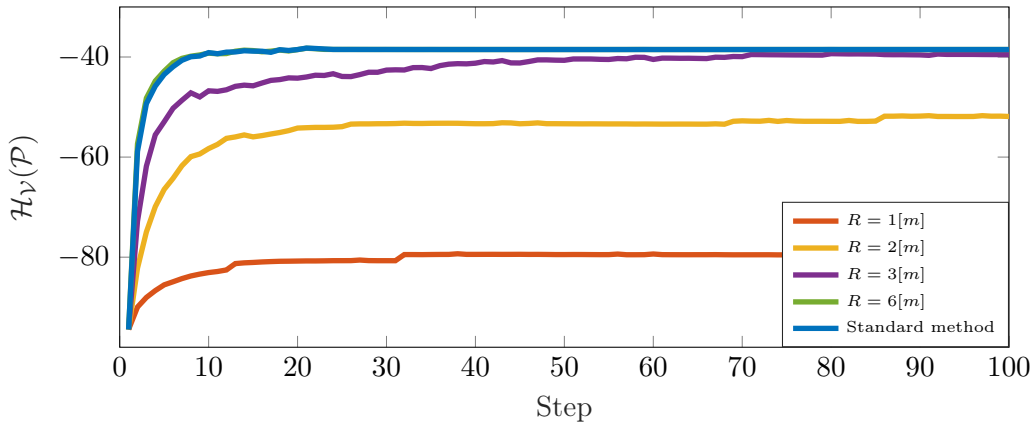


Figure 2.8: Optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$, as introduced in (2.5), achieved by a 4 robots controlled with the proposed control law (2.12), considering increasing sensing range, and compared with the standard method. The proposed method converges to the standard one for a sufficiently high sensing range.

Simulations

The simulations were carried out in MATLAB® and aimed at investigating the effectiveness of the proposed method, also in comparison to the standard method introduced in Section 2.2. When the density function ϕ is constant, the objective of the robots is to spread uniformly in the environment. To understand the difference between the proposed method and the standard one, we tested both in the same environment (Q is a square 6m $\times$ 6m).

In order to evaluate the performance of the system with the proposed controller from a global point of view, Fig. 2.8 reports the value of the standard optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$, as introduced in (2.5), considering four robots controlled with (2.12), varying the sensing range, and with the standard method (2.8). We use the standard optimization function (2.5) as a benchmark for both our method and the standard one to obtain comparable measures. We take as reference the performance of the standard method (in *blue* in Fig. 2.8), which is not affected by the sensing range of the robots. Intuitively, robots with a small sensing range are not able to optimally cover a large environment, when compared to an ideal condition of unlimited sensing. Thus, with the proposed control strategy, the robots converge to a sub-optimal solution, which represents a tradeoff achieved considering limited sensing capabilities. Thus, the coverage performance with small sensing range robots (e.g., $R = 1$m) controlled by our methodology is worse with respect to the standard one. However, when the sensing range is sufficiently big (e.g., $R = 3$m) the performance is comparable with the one obtained with the standard method and, with $R = 6$m, the performance obtained is the same as with the standard method. To visually show the difference, at steady-state, when the sensing range is small, we report, in Fig. 2.9, the initial (circles) and final positions of the robots (triangles), and the Voronoi partitions for both our case (in *blue*) and the standard method (in *red*). With our method it is clear how, with a sufficiently large environment, the robots converge to positions in which the neighbors $\mathcal{N}_{\mathcal{G}_{disk}}(p_i, \mathcal{P})$ are on the boundary of $\overline{B}(p_i, R)$, and the balls $\overline{B}(p_i, r)$ are tangent to each others, $\forall i$ (see Fig. 2.9).

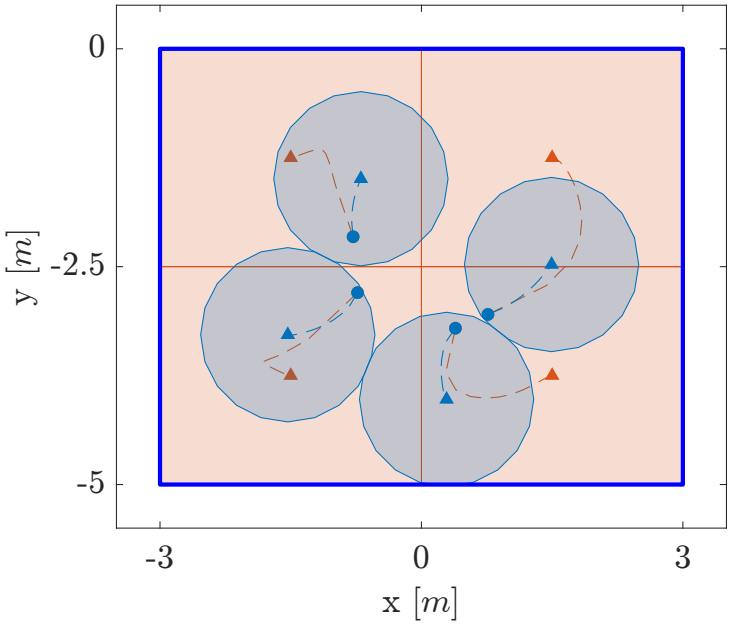


Figure 2.9: Difference between Voronoi partitioning with the standard method (in *red*) and the proposed method with $R = 2$m (in *blue*). Initial and final positions are represented with *circles* and *triangles*, respectively.

Fig. 2.10 reports, again, the optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$ defined in (2.5) as a benchmark for different numbers of robots controlled with the standard method (*solid* lines) and with our method with sensing range $R = 2$m (*dashed* lines). If a group of robots with small sensing range is not able to optimally cover a large environment, increasing the number of robots, intuitively, will improve the optimization function consequently. Thus, in the proposed control strategy the low number of robots converge to a sub-optimal solution of the coverage problem when compared to the standard one, due to the limited sensing capabilities. In fact, with 5 robots (in *blue* in Fig. 2.10) our method does not perform as the standard one. Instead, increasing the number of robots the performance is comparable. The only difference is the transient, where the convergence is reached slowly with our method as the centroids are closer to the robots' position. In our method the centroid is in $\overline{B}(p_i, r)$, while in the standard method it is inside the Voronoi cell: hence, if the cell is big, the centroid can be further away from the position of the robot. It is worth noting that the steady-state value is higher with a higher number of robots given the chosen optimization function. In fact, if the number of robots is larger, then the area is better allocated to the robots, namely each point $q \in Q$ is closer to a robot than in the case with fewer robots. It is worth noting that these results corroborate what was demonstrated in Lemma 2 and Lemma 3, respectively. In fact, increasing the sensing range or the number of robots we obtain the same value of the optimization function because the underlying limited Voronoi diagram is equal to the exact one.

Moreover, we tested the proposed approach in a more complex environment, as shown in Figure 2.11. The simulation, implemented in Matlab® , shows the motion of 6 robots deployed in a non-convex environment. Each robot (indicated with a colored dot) is directed towards the centroid (colored cross) of the respective limited Voronoi cell within the sensing range R (dotted colored circle). We remind that the standard coverage control is not suitable for non-convex environments because of the Voronoi partitioning

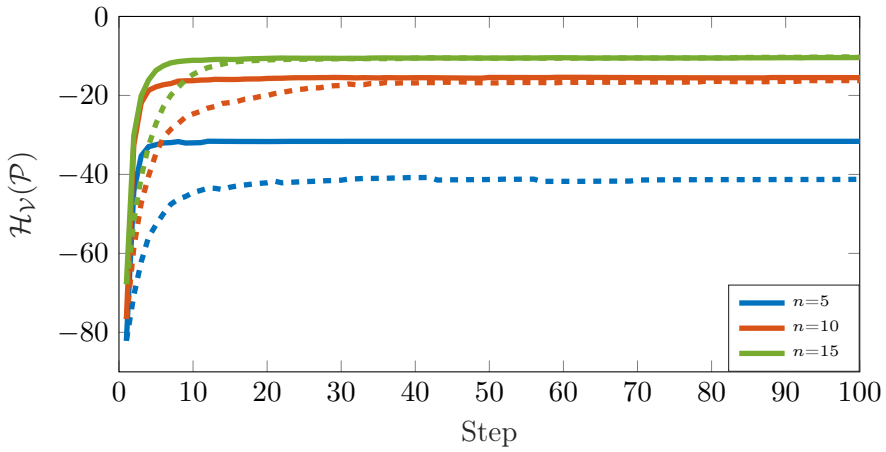
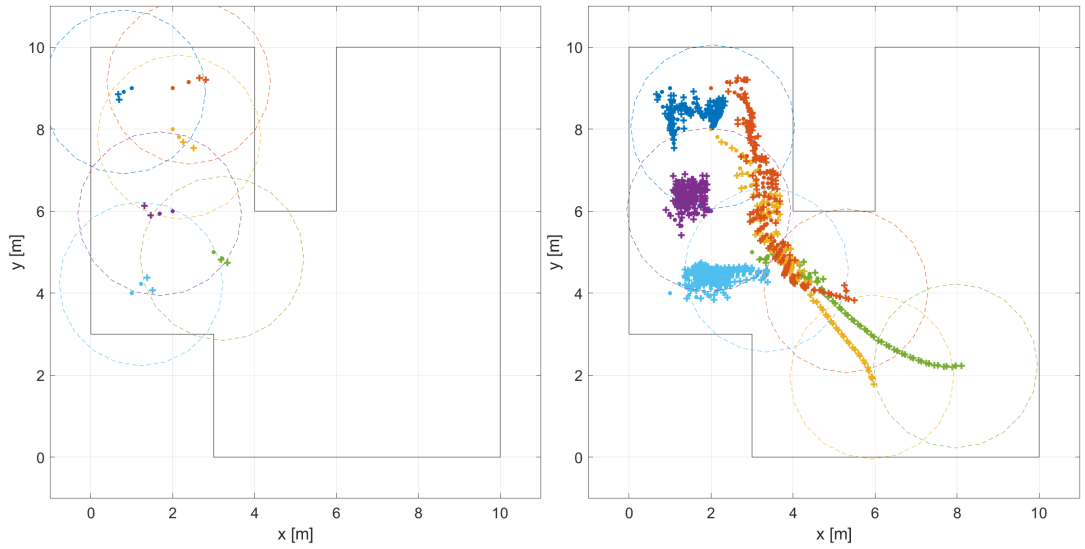


Figure 2.10: Optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$, as introduced in (2.5), achieved by a different number of robots controlled with the proposed control law (2.12). The *solid* lines represent the standard method, instead the *dashed* lines represent our method.



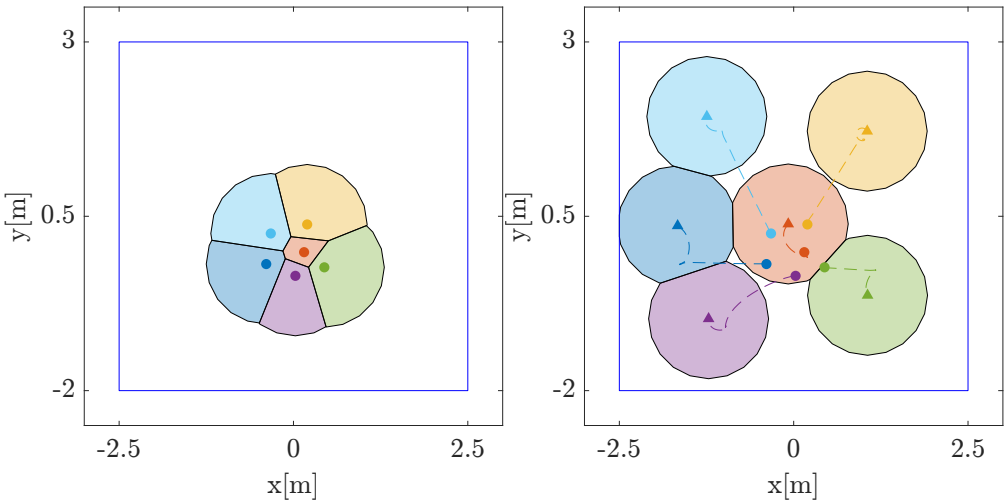
(a) $t = 2seconds$ **(b)** $t = 25seconds$

Figure 2.11: The figure shows one of the simulations we did in non-convex environments to evaluate and stress the capabilities of the proposed distributed control strategy. The robots try to optimize the coverage of the environment considering their sensing range. The coloured dots indicate the robots locations that are changing over time. The coloured crosses indicate the computed centroids. The dashed coloured lines indicate the robots sensing ranges.

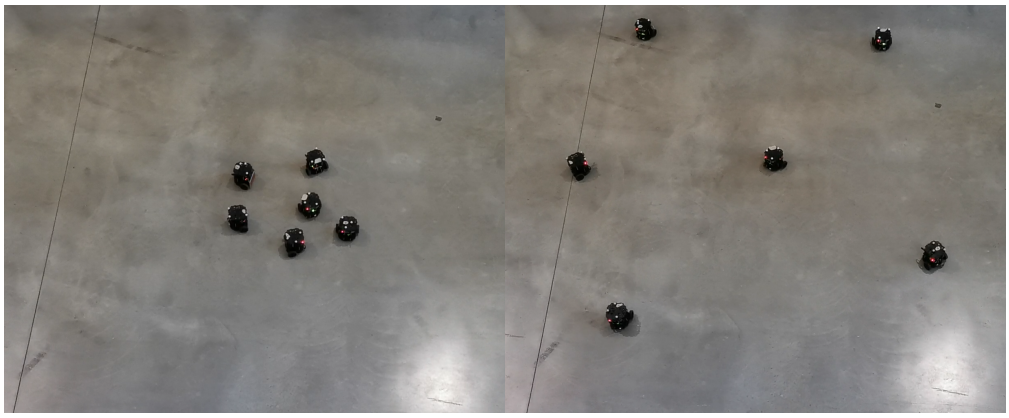
and it needs to be combined with further control strategies [68, 69]. We remark that the simulation has the aim to show an experiment in a more complex situation and that the control strategy proposed seems to be able to deal with non-convex environments. However, further studies need to be conducted on this field.

Implementation on the Robots

Some preliminary tests were carried out on real robotic platforms. The hardware set-up was made up of a group of TurtleBot 3 Burger robots, equipped with a Raspberry Pi 4 mounting Ubuntu 20.04, and of an OptiTrack tracking system. The proposed algorithm was implemented in C++ and, for the calculation of the limited Voronoi, we adapted



(a) Initial positions and limited Voronoi cells. **(b)** Final positions and limited Voronoi cells.



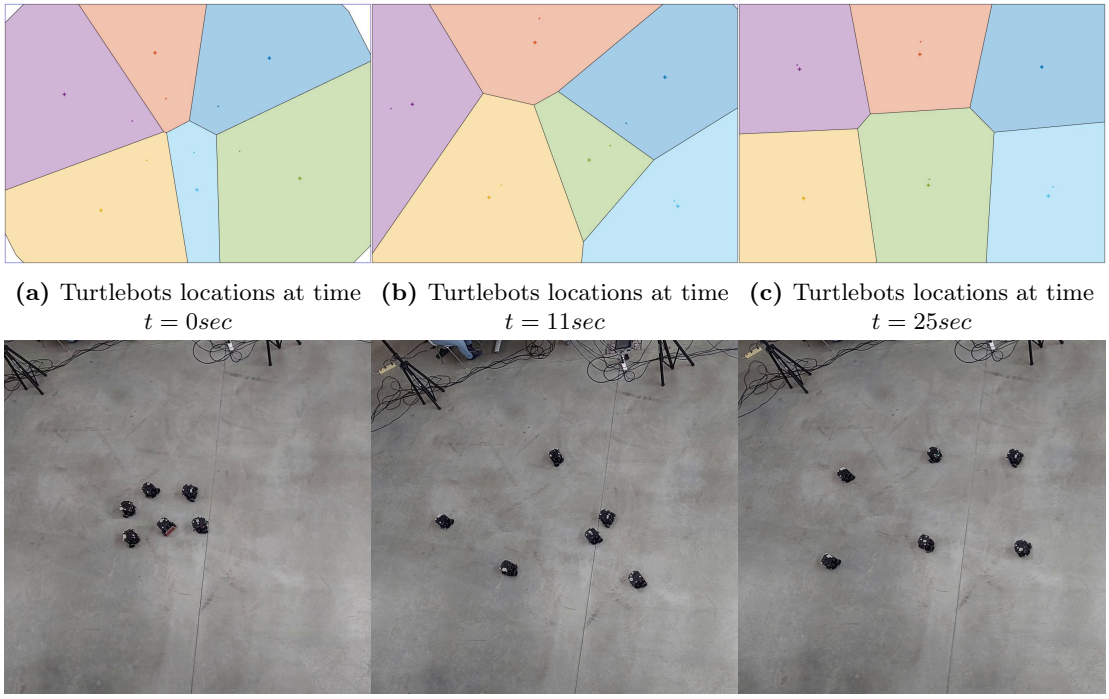
(c) Initial position of the real robots. **(d)** Final position of the real robots.

Figure 2.12: Experiments with uniform density function. In 2.12b and 2.12a every colored shape is the limited Voronoi cell of the respective robot. In 2.12b the *circles* are the starting positions, the *triangles* are the final positions.

a library[2] developed with the Fortune Algorithm [70]. The interface with the robots was implemented in ROS2, which managed also the communication with the OptiTrack system. It is worth noting that the proposed method does not need any communication between the robots and that the external tracking system can be replaced by on board sensors. In fact, the OptiTrack system was used to emulate a limited range sensor that is able only to measure neighbors' position. This information was used to create the limited Voronoi diagram with the C++ library, which calculated also the centroid of the cell based on the density function. Hence, each robot was able to calculate on-board the limited Voronoi cell V_i^r, and the centroid $C_{V_i^r}$. To obtain a faster implementation we used the line integration method proposed in [67]. The input calculated in terms of $\dot{p}_i$ from (2.12) was modified according to the input-output state feedback linearization [71] to be correctly applied to the robot. In the first experiment, the density function $\phi(q)$ was constant, hence, the aim of the robots was to spread into the environment. Fig. 2.12 reports the successful result of the experiment: initial and final positions of the robots are reported together with their respective limited Voronoi cell.

We did several experiments with a uniform probability density function varying the

[2]https://github.com/pvigier/FortuneAlgorithm



(a) Turtlebots locations at time $t = 0sec$ **(b)** Turtlebots locations at time $t = 11sec$ **(c)** Turtlebots locations at time $t = 25sec$

(d) Turtlebots locations at time $t = 0sec$ **(e)** Turtlebots locations at time $t = 11sec$ **(f)** Turtlebots locations at time $t = 25sec$

Figure 2.13: Coverage based control strategy with uniform probability density function. The group of robots starting from a concentrated configuration are tasked to optimally cover uniformly the environment. The robots sensing ranges are assumed to be large enough to include the whole domain ($R = 3$ meters). The coloured dots indicate the robots locations. The coloured crosses indicates the Voronoi cells centroids computation. The Voronoi diagrams are distinguished by the coloured polygons.

robots sensing range and analyzing the multi-robot system performance. As shown in Section 2.3.6, increasing the robots sensing range the coverage result of the proposed distributed approach is similar to the standard approach, and the limited Voronoi diagram is similar to the exact Voronoi diagram (see Figure 2.13). In Figure 2.13 the robot sensing range is sufficiently large that the union of all the sensing areas includes the domain of the environment. It is worth noting that, even the limited Voronoi diagram looks similar to the exact one, the team of robots are using only local available information: each robot does not know globally the location of the other robots in the domain. In this scenario we can see how the performance of the proposed control strategy is near to the standard one.

The second experiment aimed at testing the algorithm with a non uniform density function. We assumed $\phi(q)$ had the form of a bivariate unimodal Gaussian distribution with $\mu = \begin{bmatrix} 0 & 0 \end{bmatrix}$ and $\Sigma \in \mathbb{R}^{2\times 2}$, $\Sigma = \begin{bmatrix} \sigma & 0 \\ 0 & \sigma \end{bmatrix}$, where $\sigma = 0.3$. μ and Σ are respectively the mean and the covariance matrix of the Gaussian distribution. Fig. 2.14 reports the trajectories of the robots from the initial positions (circles) to the final positions (triangles). As expected, the robots converged to the center and they rounded up proportionally to the variance. The Gaussian distribution is represented with the *black* cross, which is the mean value, and the *black* concentric circles, which represent equidistant levels of the variance.

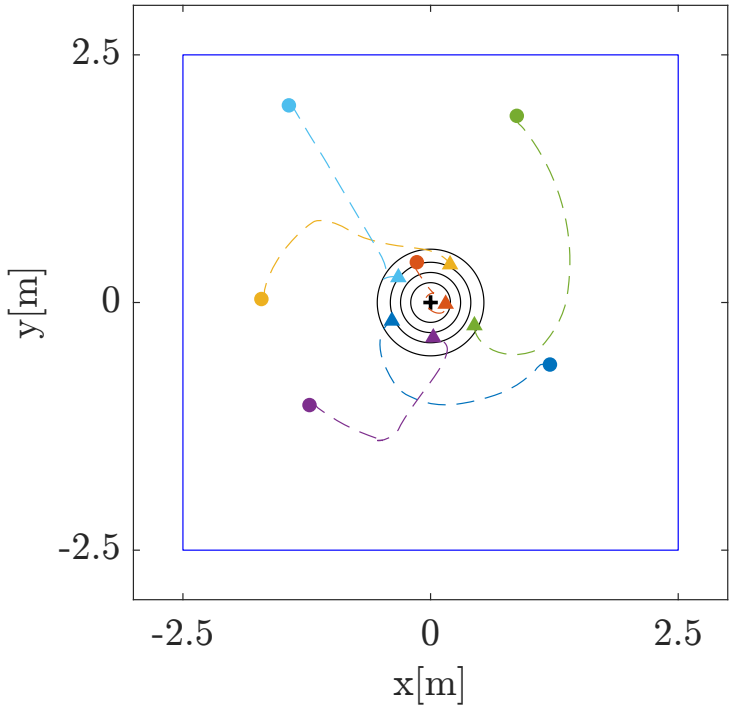
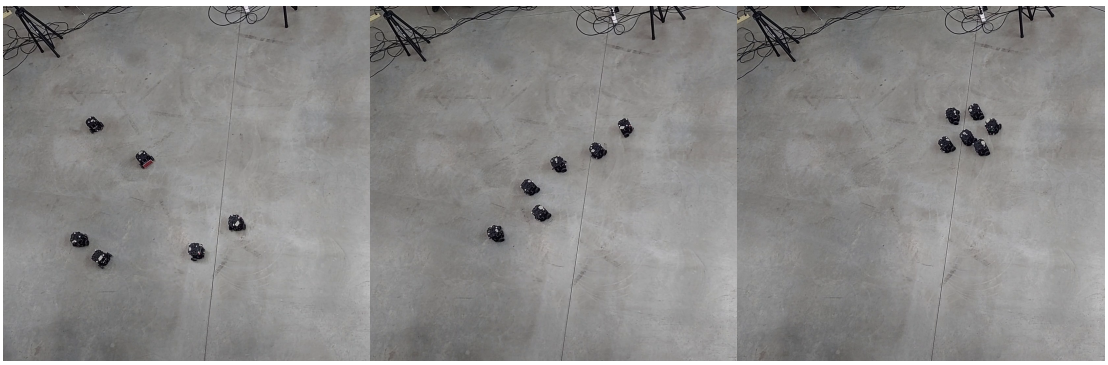


Figure 2.14: Experiments with not uniform density function $\phi(q)$. Initial and final positions with *circles* and *triangles*, respectively. The *black* cross is the center of $\phi(q)$ and the *black* concentric circles are different values of variance.



(a) Turtlebots locations at time $t = 1sec$ **(b)** Turtlebots locations at time $t = 9sec$ **(c)** Turtlebots locations at time $t = 17sec$

Figure 2.15: Coverage based control strategy with Gaussian probability density function. The group of robots starting from a randomly spread configuration are tasked to optimally cover the environment around the area of interest defined by the Gaussian function ($\mu = (0.8, 0.8); \sigma = 0.2$).

In Figure 2.15 we show one of the experiments we did with a fleet of Turtlebots testing the proposed strategy with a Gaussian distribution. In these experiments the robots are randomly deployed in the environment and the Gaussian function is randomly chosen. The function in Figure 2.15 is defined by $\mu = (0.8, 0.8)$ and $\sigma = 0.2$.

2.3.7 Discussions

In this study, we presented a distributed coverage based control to coordinate a group of autonomous robots with limited sensing capabilities in an unknown environment without a reliable communication network. The proposed novel control strategy, based on a limited Voronoi partitioning, uses only the information directly detected by the robot. We

formally show how the control law drives the group of robots to the respective centroid of the limited Voronoi cell. In addiction, we presented a methodology to deal with numerical issues that come from the Gaussian-type probability density function typically used in the coverage base control algorithms. The variance-varying methodology connects the control law with the computational capacity of the robot. A series of experiments and simulations were performed on a group of mobile robots to validate the performance of the proposed control method.

2.4 Coverage Control for Large-Scale Teams of Aerial Drones

When considering a multi-robot aerial system, coverage control can be seen as a basic behavior serving as a building block for complex mission. However, when the number of units in play becomes too high, centralized and global optimal solutions might not scale well, causing a reduction in efficiency and reactiveness. To this end, in this study we analyze the control methodology proposed in section 2.3 for area coverage that is fully distributed and is able to mitigate the above factors while still providing high performances. The approach we analyze in this study relies on local Voronoi partitioning constrained by limited robot sensing and communication capabilities. To validate our approach, we perform an ablation study over a large set of simulations that we evaluate with a set of proposed metrics. At the same time, the algorithm has been tested on the field, with real quad-copters, and with a set of proof of concept experiments deploying up to seven aerial units.

2.4.1 Literature Review

The literature is flourishing with strategies proposing multiple implementation of coverage control performed by a group of mobile robots—either ground, aerial, marine or hybrid. Amongst these, [1] proposes decentralized coordination algorithms for groups of mobile agents based on Voronoi diagrams [48], [52], [62] and proximity graphs [50], [66]. This control strategy provides a methodology to guarantee convergence of the positions of the robots in the team to a configuration that maximizes the area covered in the environment. The main drawback is related to the fact that the computation of each robot's control input requires the computation of the Voronoi partitioning of the entire environment, which requires global knowledge and/or global communication and information exchange [51], [47]. However, in many practical application scenarios, robots may not have access to global information or to a reliable all-to-all communication network: therefore, methods were proposed to define the Voronoi partitioning in a range-constrained scenario. To overcome these issues, the methodology proposed in section 2.3 aims to implement coverage control in a totally distributed manner, letting each robot compute its own control input base on locally available information only. In particular, the control strategy allows a team of mobile robots, moving in a two-dimensional environment, to optimally cover a bounded region with unknown feature. Robots are considered with lim-

ited sensing capabilities, and without access to reliable high-bandwidth communication infrastructures.

The aim of this work is to experimentally evaluate the performance of the coverage control architecture proposed in 2.3 deployed on a large team of drones for outdoor operations. In this section, we perform a deep and extensive analysis of both realistic simulations and real field tests with a fleet of quad-rotor drones in an unstructured environment as the Abu Dhabi desert. Thanks to these, we assess the scalability of the control methodology and its performance in the presence of real-world constraints and limitations such as localization error coming from noisy GPS readings. In order to guarantee the statistical relevance of the simulated results, we perform an ablation study of the main parameters with randomized initialization conditions repeated over multiple runs. We note that this represents the first large-scale deployment of a team of drones performing fully decentralized and distributed coverage control in an outdoor environment.

2.4.2 Tests and Setups

The aim of the experiments is to evaluate the performance of a team of drones controlled by a decentralized, limited-range, coverage control strategy (section 2.3), deployed in a large outdoor environment. For this purpose, field trials and high-fidelity dynamic simulations were set up, to achieve a statistically relevant characterization. Details of the methods are provided in the following paragraphs.

2.4.3 Simulation Protocol

Simulated experiments have been performed using the Flightmare Quadrotor Simulator [72], a photo-realistic simulator, with integrated physics, developed by the Robotics and Perception group at UZH. The control algorithm described in Section 2.3.4 has been implemented using the Robot Operating System—ROS—on separate threads, one for each drone. Each robot thread also handles the low-level controller for the quad-rotor fly, the initial take-off, and the sharing of information. A python script acting as a parallel daemon manages the execution of the simulation batches and collects the data needed to assess performances. The simulations have been performed on an Intel Xeon Platinum 8280M CPU@2.70 GHz with 4 processors, for a total of 112 cores and 512 gigabytes of RAM. Each simulation collects the global position for each drone over a time interval of 120 seconds—empirically set at the end of the simulation. To guarantee the Independence of the proposed methodology from the initial conditions, drones are spawned inside the bounded environment in randomized positions: for every simulation run, the vehicles are spawned within the simulation environment in a neighborhood of the center of the coverage area with a random displacement. The final position is bounded to be at least 0.5m from any of the already positioned vehicles. After arming and take-off of all the vehicles at the same altitude, the coverage control algorithm takes over and steers the drones toward their wanted position thanks to the velocities by Equation (2.12). The simulator makes use of the RPG Quadrotor control [73], which allows for generating desired orientation, collective thrust command, body rates, angular accelerations, and

angular jerk to generate the control values of the four motors from the requested velocity command.

The measurements of the positions of the other vehicles, used to feed the control function, are added to a noise value generated by a normal distribution $\mathcal{N}(0, 1)$ to simulate realistic measurement error. Ten simulations have been performed by combining different parameters expressed by a tuple (n, a, r) respectively representing: the number of drones $n \in N = \{8, 32, 64\}$; the size of the operational area $a \in A = \{50m, 100m, 200m\}$; and the sensitivity range $r \in R = \{10m, 50m, 100m, 500m\}$.

2.4.4 Field Test Protocol

Real-world experiments have been performed in a remote location in the Al Khaznah region of the Abu Dhabi emirate in the UAE. The platform selected as the flying unit is depicted in Figure 2.16. It is a custom-built quad-rotor developed by the company Fly4Future, consisting of a standard DJI frame of 450mm, a Pixhawk4 autopilot with custom firmware, and an Intel NUC with i7 processor and 16 gigabytes of RAM as an on-board computer. The latter is used to run the high-level software control stack that, in this particular case, runs on the MRS UAV system openly available for research purposes and released by the Multi-robot Systems Group at the Czech Technical University in Prague [74].



Figure 2.16: Figure of the MRS F450, the quad-rotor platform selected for the experiments

The robotic unit is equipped with several sensors among which, for the specific use case we analyze here, we highlight a Global Positioning System module for self-localization—as the one coming with the Pixhawk autopilot bundle—and a XBee devices for communication among the units and information sharing. The latter, makes use of an ad-hoc peer-to-peer proprietary mesh-network called DiGiMesh that we limit to be single-hop

to force the range communication constraint. We employed omni-directional low-power antennas on board all drones, each of which runs on 900Mhz empirically estimated to guarantee a good connectivity among all the units in the group. The vehicle controller [75] allows to generate the desired orientation, collective thrust command, body rates, angular accelerations, and angular jerk based on the coverage control. The experiments were carried out using 7 vehicles, in an area of 50×50 m; as previously described, the communication devices used offer full connectivity but, to evaluate the behavior of the control strategy we manually limit the effect of the sensing range r with the same values used for the simulations— i.e., $R \in \{10m, 50m, 100m\}$. Three experiment runs have been performed for each value of the sensing range. As in the simulated scenario, the control algorithm proposed in Section 2.3.4 runs in a distributed way on board of each drone. The position information of neighboring units needed by the coverage control are solely provided through radio communication and only by the aerial units—i.e., no external or central computer is used. Figure 2.17 shows the trajectories of the UAVs in one of the experiments carried out; following the takeoff procedure to bring the vehicles to the same altitude, the coverage control function is performed to determine the new positions of the vehicles.

The speed control of Equation 2.2 is limited to $2m/s$ for safety reasons, taking into account the number of vehicles involved in the experiment and the degree of freedom that the drones have to move in various directions.

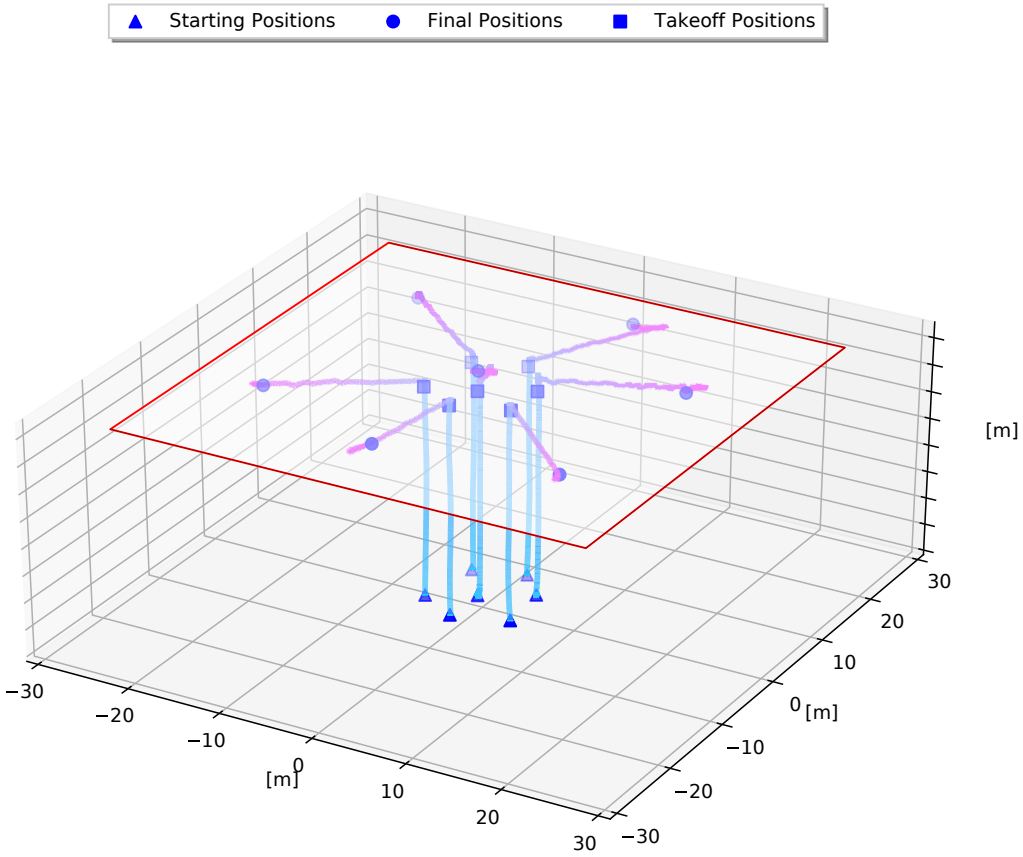


Figure 2.17: Vehicles trajectories of a real experiment with 7 drones

2.4.5 Metrics

In order to evaluate the overall performance of the system, the same metrics were collected both in simulations and in field tests. In particular, we consider the following three main aspects: *(i)* optimality of the final configuration, *(ii)* efficiency in the coverage of the environment, and *(iii)* effectiveness in the use of the robots.

Optimality of The Configuration

As is typically done in the literature, the performance of the coverage control algorithm is evaluated computing the value of an optimization function $\mathcal{H}_{\mathcal{V}}(\mathcal{P})$ as introduce in (2.5), whose value indicates how close the displacement of the robots is to the optimal configuration, that is the centroidal Voronoi configuration. We report the optimization function in (2.26), referring to it as $\mathcal{H}(\mathcal{P})$ simplifying the notation.

Considering the positions $\mathcal{P}$ of the robots, it is possible to partition the environment Q into n *range-unlimited* Voronoi cells $\{\bar{V}_1, \ldots, \bar{V}_n\}$, where the i-th cell is computed using position p_i as a seed. The optimization function is then defined as follows:

$$\mathcal{H}(\mathcal{P}) = -\sum_{i=1}^{n} \int_{\bar{V}_i} \|q - p_i\|^2 \phi(q) dq \tag{2.26}$$

As discussed in [1, 47], such definition allows to quantify how close the configuration of the robots is to the centroidal Voronoi configuration, defined considering, for each point $q \in Q$, the value of the pdf $\phi(q)$.

Coverage Effectiveness

Considering the limited-range sensing capabilities of the robots, in order to assess how well the environment Q is covered by the set of individual robot non-overlapping limited-range cells $\{V_1, \ldots, V_n\}$, we propose to evaluate the ratio $\mathcal{A}(\mathcal{P})$ of the integral of the pdf $\phi(\cdot)$, computed over the summation of the cells V_i, with respect to the same integral computed over Q, namely

$$\mathcal{A}(\mathcal{P}) = \frac{\sum_{i=1}^{n} \int_{V_i} \phi(q) dq}{\int_{Q} \phi(q) dq} \tag{2.27}$$

Overall Efficiency

Considering the fact that each robot is able to individually cover a circular area with radius r, we measure the effectiveness in the use of resources computing the ratio between the actual area covered by the team of drones, and the upper-bound on the area that

could be theoretically covered, namely:

$$\mathcal{E}(\mathcal{P}) = \frac{\sum_{i=1}^{n} \int_{V_i} dq}{n\,\pi\,r^2} \tag{2.28}$$

2.4.6 Results and Analysis

In this section we report the results for both simulations and field tests performed according to the presented methodology. Quantitative results are related to the value of the optimization function defined in Equation (2.26), with larger values corresponding to a configuration that is closer to the optimal one; to the integral of the PDF defined in Equation (2.27), where again larger values reflect better coverage efficiency; and last to the measurement of the overall covered area as defined in (2.28), where greater values reporting a better effectiveness in the usage of the robots.

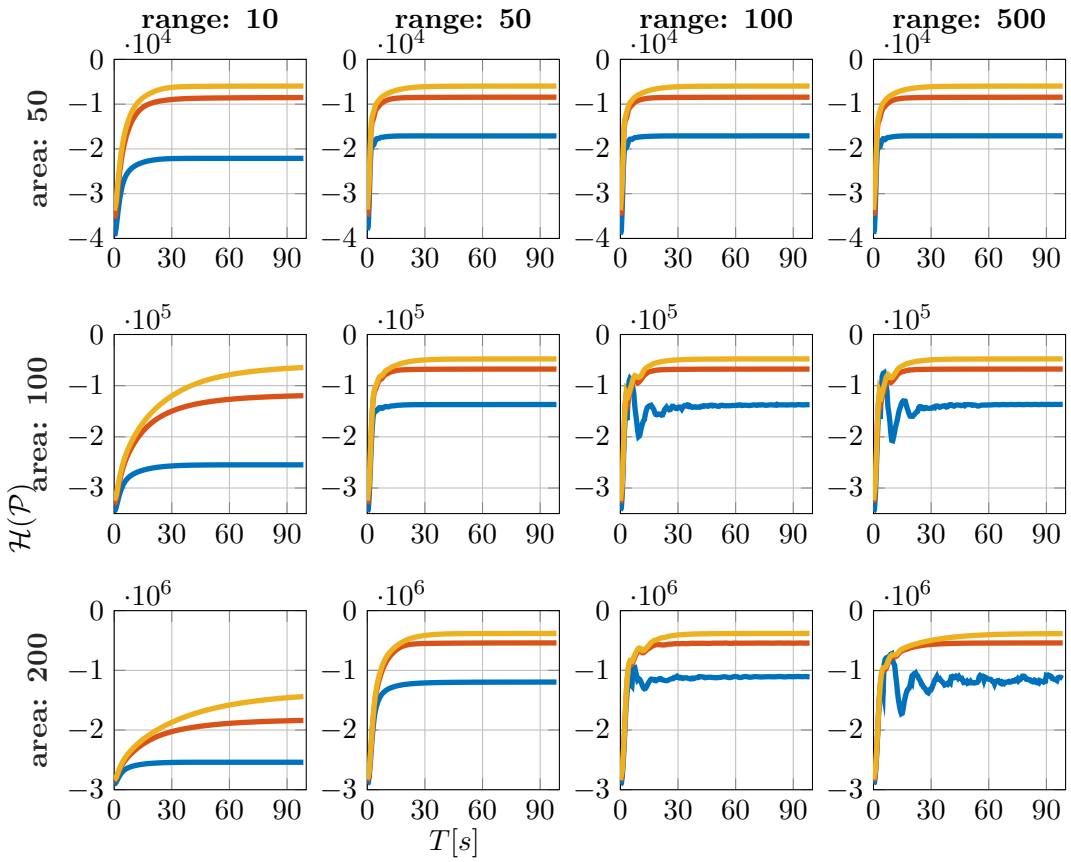


Figure 2.18: Average value of $\mathcal{A}(\mathcal{P})$ across all repetitions. The plots on the same line refer to the same total area to cover, while the plots on the same column refer to cases with same sensing range. The values for 8, 32 and 64 robots are displayed with blue, red and yellow lines respectively.

Simulations Results

We hereby present the results of the simulations. Figure 2.18 displays the average value of $\mathcal{H}(\mathcal{P})$ across the set repetitions. The blue line shows the value for 8 robots, the red

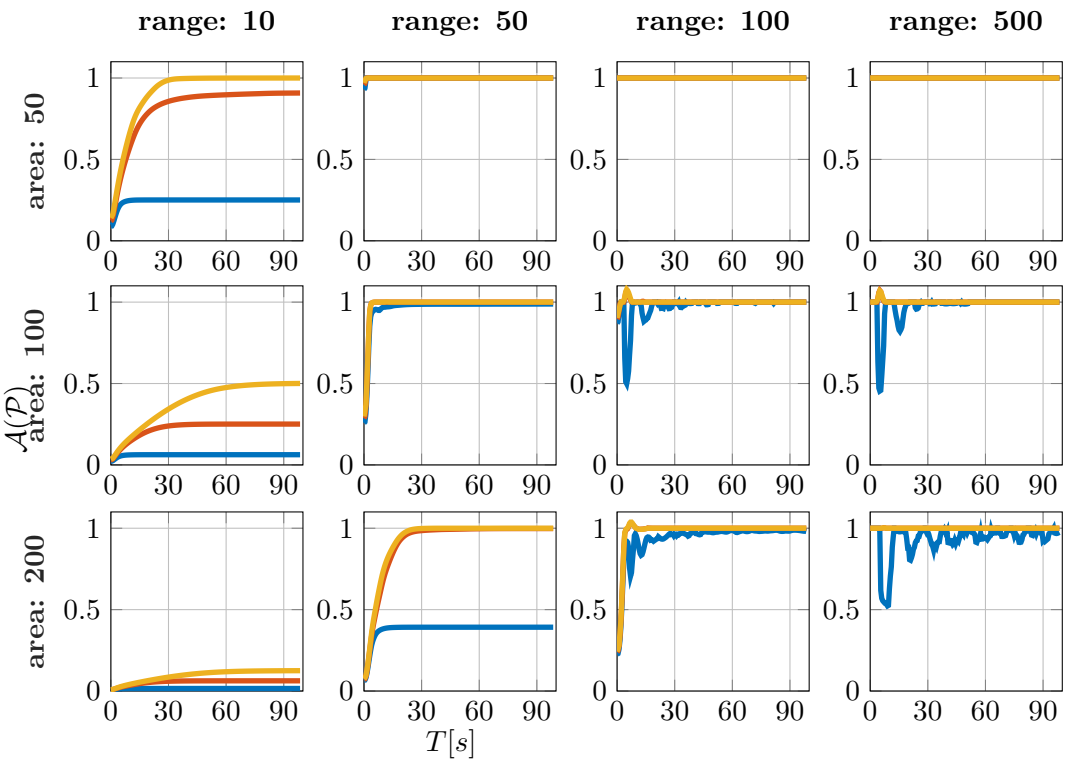


Figure 2.19: Average value of $\mathcal{A}(\mathcal{P})$ across all repetitions. The plots on the same line refer to the same total area to cover, while the plots on the same column refer to cases with same sensing range. The values for 8, 32 and 64 robots are displayed with blue, red and yellow lines respectively.

line for 32 robots, and the yellow line for 64 robots. Each plot is organized according to the size of the area to cover and the sensing range. Following the same color scheme, Figures 2.19 and 2.20 display the average value of $\mathcal{A}(\mathcal{P})$ and $\mathcal{E}(\mathcal{P})$ respectively.

Figure 2.18 clearly shows that increasing the number of robots increases the performance with respect to the coverage metric. This effect is intuitively attributable to a greater cumulative covered area. It is also possible to note that the rate of convergence: (i) decreases with the increase of area size, (ii) increases if the sensing range grows, and (iii) decreases the more robots are employed. The oscillations present on the blue line for the cases with large Q and big sensing range are caused by the dynamics of the drones. Inherently, each drone possesses a certain motion dynamic different from $\dot{p}_i = u_i$. The velocity controller that generates the motor commands controls the drone so that its motion is as similar to (2.2) as possible. However, if the requested velocity is too different from the current velocity, the controller encounters acceleration limits that render it impossible to closely track the requested velocity profile. With large Q and large sensing range, the resulting V_i can be very large and their centroid can be far away from the drone. Therefore, the input generated by the control law (2.12) is very large, requiring accelerations outside the limitations of the drone, causing the oscillating behavior depicted by the blue line. With a larger number of drones, this behavior is limited or totally absent since each drone has a larger number of neighbors, therefore its resulting cell V_i is smaller. This effect can be reduced by reducing the value of k_{prop} in (2.12) or by applying a saturation on the value of u_i. Nevertheless, the coverage algorithm is still capable of spreading drones across the area.

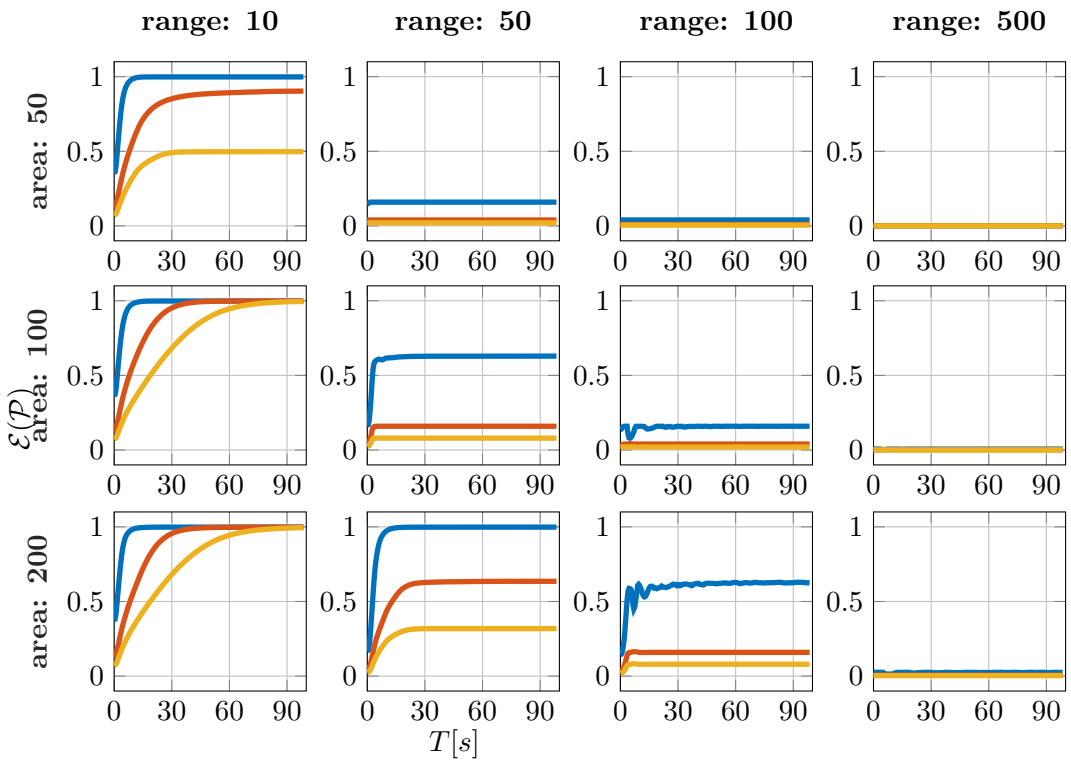


Figure 2.20: Average value of $\mathcal{E}(\mathcal{P})$ across all repetitions. The plots on the same line refer to the same total area to cover, while the plots on the same column refer to cases with same sensing range. The values for 8, 32 and 64 robots are displayed with blue, red and yellow lines respectively.

Figure 2.19 shows how well the area is covered by the drones. When the sensing range is small, the robots often struggle to cover the totality of Q. The covered area increases with the increase of the number of drones employed. This is expected since, obviously, more robots imply more cells V_i and the behavior for each drone is to distance itself from its neighbors, enlarging its cell V_i.

Figure 2.20 depicts the efficiency of the coverage. The plots clearly show that fewer robots are more efficient. A value of 1 for $\mathcal{E}$ means that each robot has reached the configuration where its cell V_i is the largest possible. With few robots there are fewer neighbors to interact with, ultimately allowing for the maximum enlargement of each cell. The condition for a team of robots to not reach maximum efficiency that the total area Q cannot accommodate all the cells of the robot.

These considerations ultimately allow us to set some design analysis to make when deploying a coverage application. A larger number of robots is not necessarily better in all scenarios, even if it increases the performance. Depending on the sensing range, which is often dictated by different factors, a larger number of robots can imply a better area coverage but a trade-off has to be considered, in order to not overly reduce efficiency.

Field Experiments Results

We continue our analysis of the results with real field experiments involving a group of seven drones. Results are presented in Figure 2.21 where all the metrics detailed in

Section 2.4.5 are utilized. The first line of graphs describes the metric $\mathcal{H}(\mathcal{P})$ for the different sensitivity range values taken into consideration. The second and third lines describe $\mathcal{A}(\mathcal{P})$ and $\mathcal{E}(\mathcal{P})$ respectively.

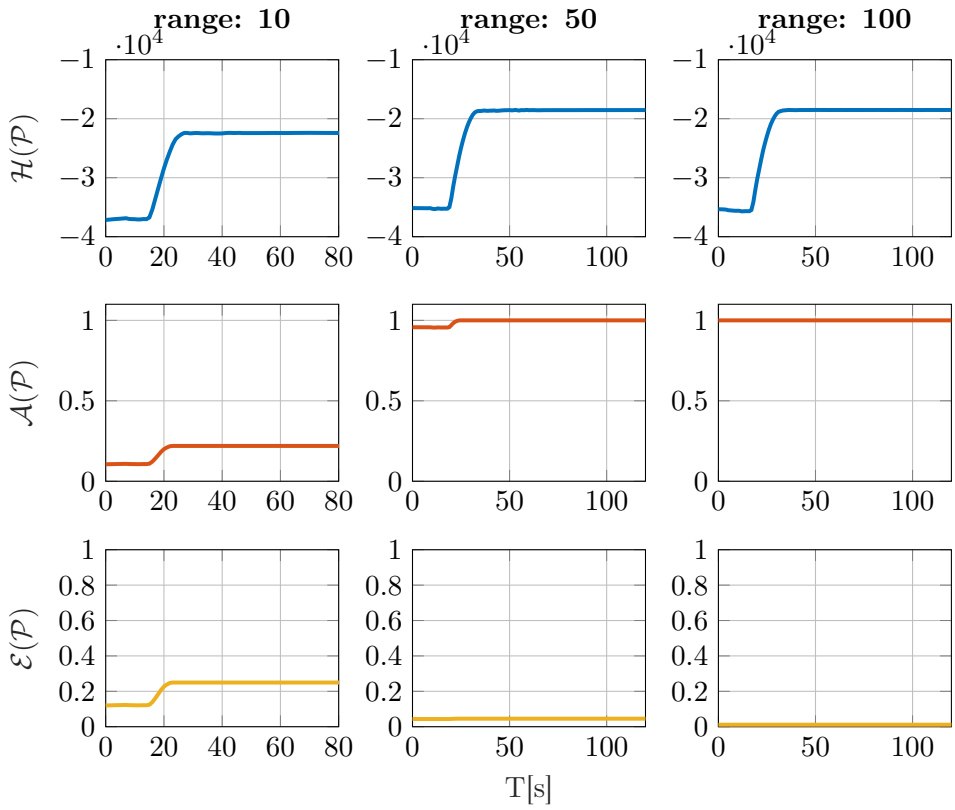


Figure 2.21: Average value of Equations (2.26), (2.27), (2.28) across all repetitions. The plots on the same line display the same metric, while the plots on the same column refer to cases with same sensing range.

Parallel to the simulated case, it can be seen how the metric of $\mathcal{H}(\mathcal{P})$, first row, blue line, increases as the value of the range increases. Unlike Figure 2.20 it can be seen that the convergence times of the optimization function in Equation 2.26 are longer. This is due to a cap to the maximum velocity of $2m/s$ introduced for safety reason and to avoid long overshoot with the real platform. Results for values of r equals to $10, 50$ and 100 show different convergence time. For $r = 50$ and $r = 100$ we note how the velocity cap is particularly detrimental since the vehicles unable to reach the speed outputted by the proportional control law and the optimization function requires more time to reach the steady state.

Figure 2.22 shows the initial and final conditions of the whole control process. Coloured dots show the position of the each drone in the group at start—leftmost figure—and end—rightmost figure—of the simulation. The circles surrounding the dots represent the sensing range of the robots. In particular, by visual analysis of Figure 2.22b, it can be noted that the final group configuration does not show a compact and close formation. Indeed, while in perfect conditions the final graph should have adjacent but non-overlapping sensing ranges—circles—this is not the case here. By analysis of the data, it has been noticed that, during the experiment, the GPS covariance matrix for $\sigma^2_{x,x}$ and $\sigma^2_{y,y}$ ranged in the interval $[1.0, 1.4]$ meters, while values for the EPH, the

standard deviation of horizontal position error, and EPV, the standard deviation of vertical position, were oscillating between 0.6 and 0.95 meters. These values, and the position uncertainty that derives from them, justify the distance between the sensitivity ranges.

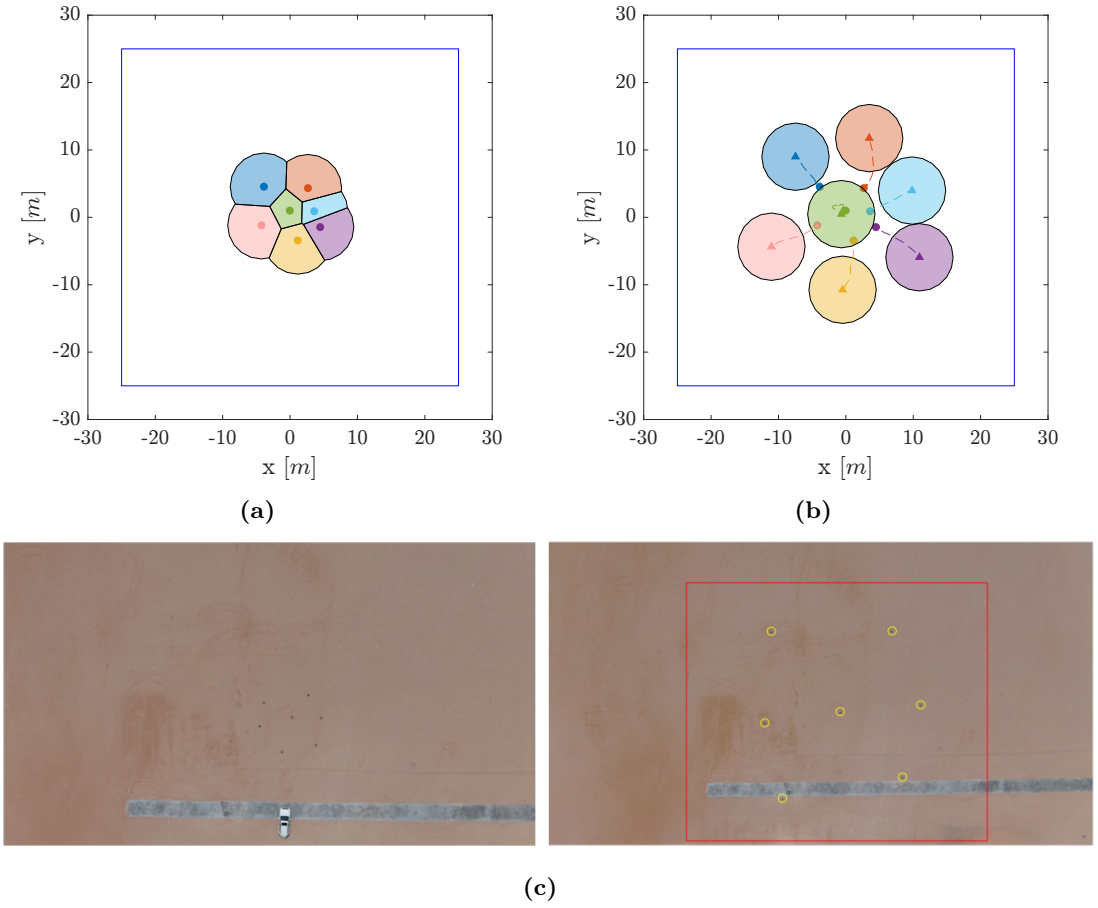


Figure 2.22: On image (a) the initial configuration of the vehicles during a real experiment with sensitivity range $R = 50$; on image (b), the final configuration of the drones in the same experiment. On image (c) an aerial photo of the convergence process is shown, with a different orientation of the axes.

2.4.7 Discussions

In this study, we tested our distributed coverage control algorithm capable of arbitrating a group of autonomous vehicles with limited sensing capabilities. Thanks to an extensive set of simulations and the implementation of real field experiments, the strategy has been proven to be robust and efficient. It is indeed capable of achieving close-to-optimal performances both in simulation and reality and, as shown, it is also capable of orchestrating a significant number of drones marking the approach as scalable. On the other end, real experiments have shown how the presence of non-ideal conditions, such as noisy GPS reading, have an effect on the final configuration but overall do not compromise the behavior of the coverage control function.

2.5 Coverage Control for Exploration of Unknown Non-Convex Environments

In this section we presents a novel methodology for the exploration and monitoring of unknown environments performed by a group of coordinated mobile robots. The proposed methodology builds upon the Voronoi-based coverage control methods, described in section 2.3, with time-varying probability density functions. In particular, we propose a methodology that, exploiting locally available information, allows robots to dynamically update the probability density function in order to drive the robots to explore previously unexplored areas. The proposed control strategy has been validated in extensive simulations, and by means of experiments carried out on a group of mobile robots. Results of simulations and experiments demonstrate the effectiveness of the proposed methodology in leading to the exploration of large portions of the environment.

2.5.1 Literature Review

The problem of coordinating multiple robots with the task to explore (and map) an unknown environment has been widely studied in the last years. One of the most widely adopted strategies, firstly described in [76], exploits potential fields to generate collision-free trajectories. Every robot is treated as a positive ion, while obstacles are assumed as positively charged and the goal as negatively charged, thus resulting in every robot being driven to the goal while rejecting both other agents and obstacles. Other control strategies can be derived from the well known random walk approach, as the ones proposed in [77], where robots move randomly within the environment. In [78] the exploration of the environment is based on the creation of a single global map from the sensor information of the individual robots. The robots control exploits the support of a central elaboration unit to decide the direction of motion to better create the map. An alternative approach is the one proposed by [79], where the authors study the problem of multi-robot exploration and coverage with limited communication. They propose a decentralized algorithm based on leaving marks on the unknown terrain, which can be sensed by the robots and allow them to cover the terrain. The algorithm is based on a division of the environment in grid cells, each of which is then dynamically marked as "explored" or "not explored". The limited communication issue has been considered also by [80], where the control of the multi-robot system relies on the communication with a base station. In order to assist the robot coverage of the environment, [81] proposes an algorithm that deploys radio beacons and, similarly, [82] proposes robots that, in the process of moving around, deploy network nodes into the environment.

One of the most relevant and studied exploration methodologies is the frontier based coordination approach [83]. The environment is discretized in standard occupancy grids and a frontier cell consists in each already explored cell that is an immediate neighbor of an unknown, unexplored cell. A mapping of the environment is required, usually generated exploiting grid maps or graphs, to keep updated the portion of the environment already explored and to localize the frontier nodes positions [84]. The communication between

the robots is a critical issue given the amount of data that needs to be shared to keep updated the explored grid map of the environment. Information nodes with sensing and communication capabilities can be deployed by the robot to deal with the communication network issues, especially in the exploration of large-scale environments [85].

Coverage based control strategies (see Sections 2.2 and 2.3), used to coordinate a group of autonomous robots with the aim of maximizing the coverage of the environment, are tools that can be exploited to deal with the exploration of unknown environments. An example is the study in [86], where the Voronoi diagram is used to generate an obstacle-free convex neighbourhood among strongly convex obstacles. A significant work is presented by [87] where a frontier-based exploration methodology is integrated with a Voronoi partitioning of the environment to avoid duplicated exploration areas. In both of these works the boundary of environment and the obstacles needs to be fully or partially known. An alternative approach was presented in [88], where they exploit a polygonal space representation based on the Voronoi partitioning. The map of the environment is assumed to be known and shared among the robots. A recent study presented in [89] deals with the problem caused by updating a single global grid map managing only local information, without considering limitations on the communication network. In [90] an exploration and navigation framework has been proposed to explore GPS-denied environments using the cooperation between UAVs and UGVs to increase the accuracy of the SLAM algorithm. Another widely studied methodology is the one that exploits potential fields [91]: control input for the robots is defined based on the gradient descent of an artificial potential field, constructed such that each robot is repelled by obstacles and other robots, thus forcing the networked system to spread in the unknown environment. This approach does not require global localization or communication, however, the robots will stop once sufficiently distant from each other and the unseen areas of the environment will be not explored.

The mentioned studies and the majority of the approaches in multi-robot exploration are based on a discretization of the environment, typically exploiting occupancy grids. This approach is not suitable for large environments, where the number of cells grows significantly, thus leading to severe computational and memory consumption. Moreover, these approaches require a strong communication network in order to keep updated the shared map among the robots [85]. A methodology that deals with communication issues has been presented in [92], where the total area is partitioned based on the topology exploiting the Generalized Voronoi Graphs for exploration purposes. However, the robots are constrained to move on the Voronoi diagrams edges. Moreover, this strategy is mostly effective in environments structured in a way that naturally drives robots during exploration, e.g. with corridors, losing effectiveness in more generic spaces.

An alternative and relevant approach to obtain a mapping and exploration of large environments is presented in [93]. A Gaussian mixture model for global mapping is used to model a complex environment. This approach, developed and tested for one robot monitoring, has the benefits to require a small memory footprint and a low volume of communication. Following this strategy, in this work, we propose a methodology that exploits a mixture of Gaussian distributions to coordinate a group of robots and explore

an unknown environment. Given the Gaussian function defined by only a few parameters and the lack of a grid discretization of the environment, the strategy aims to an efficient memory usage and a low communication data among robots.

Proposed strategy - Introduction

Our methodology builds upon the coverage control strategy described in 2.3 able to deal with unknown non-convex environments. The aim is to obtain a multi-robot system that dynamically explores an unknown environment optimizing the coverage of the most interesting areas in order to keep the environment constantly monitored. The proposed approach uses a mixture of Gaussian functions to model the already explored ares. The exploration and monitoring is, hence, performed without a discretization of the environment (grid map or graph) and without any knowledge of the boundaries. Moreover, the few parameters that define a Gaussian function can be easily shared among the robots keeping both the communication and the memory usage low. Finally, we extensively tested the proposed control in both simulated and real environment experiments. We compare the proposed approach with other typical ones that do not require a discretization or a partial knowledge of the environment, such as random walk and potential field approaches.

2.5.2 Problem Statement

Consider a multi-robot system constituted by n holonomic robots able to move in two dimensions, controlled with the objective of monitoring and exploring an unknown environment. We assume each robot to be modeled as a single integrator system (according to (2.2)), whose position $p_i \in \mathbb{R}^2$ evolves according to $\dot{p}_i = u_i$, where $u_i \in \mathbb{R}^2$ is the control input, $\forall i = 1, \ldots, n$.

We consider the following assumptions:

1. *Localization with respect to a global reference frame*: each robot is able to localize itself with respect to a global reference frame, which is shared among the robots in the team;[3]
2. *Limited sensing capabilities*: each robot is able to measure the position of neighboring robots and objects (including boundaries of the environment), within its limited sensing range (defined by a circle with radius $r_{sens} \in \mathbb{R}_{>0}$).

Based on these assumptions, the problem addressed in this study is then formalized as follows:

Problem. *Define a distributed control strategy that allows a multi-robot system with limited sensing capabilities to explore an arbitrary unknown, possibly non-convex, large environment. The methodology needs to consider the possibility to not be able to discretize the environment due to memory and communication limitations.*

[3]This can be achieved, e.g., by means of Global Positioning System (GPS) or cooperative localization techniques, which are however out of the scope of the study and will not be addressed.

2.5.3 Proposed Strategy: Time-Varying density function

As introduced in Section 2.5.2, the aim of this work is to monitor and explore an unknown and possibly non-convex environment with a group of robots with limited sensing capabilities. In order to fulfill the exploration of the whole environment, the robots need to keep updated the areas already explored by the multi-robot system. A probability density function can be exploited for this purpose focusing the attention on the areas to be explored and avoiding areas recently visited. The proposed distributed control algorithm uses this approach to keep updated and shared information about the explored environment.

The probability density function is defined as a combination of multiple Gaussian density functions. Each Gaussian function $\varphi(q)$ is determined by a mean value $\overline{q} \in \mathbb{R}^2$ and a standard deviation $\sigma \in \mathbb{R}_{>0}$, and is then defined as follows:

$$\varphi(q) = \exp\left(-\frac{\|q-\overline{q}\|^2}{2\sigma^2}\right). \tag{2.29}$$

Every agent in the environment must be able to access this data when $\overline{q}$ lies inside its sensing range, thus the exchange of information can be achieved exploiting a central storage unit or physically placing a marker or a beacon that each robot can easily localize. During the motion and the exploration, every robot places several Gaussian functions in the environment. The combination of all the shared Gaussian functions determines the motion of the multi-robot system. Every time a new area is explored by a robot, a new Gaussian function $\varphi(q)$, defined as in (2.29), is added by the robot to the shared density distribution following the algorithm described later (Algorithm 1), affecting the motion of the overall multi-robot system. During the exploration of an unknown environment, we consider two possible scenarios the robot can face: (1) the robot explores areas that need to be crossed only once during the navigation, and (2) areas that need to be crossed more than once to complete the exploration without deadlocks. Thus, we define two types of Gaussian functions to be added by the robot following the definition in (2.29): a time-constant Gaussian function and a time-varying Gaussian function. The first one intuitively remains constant over time during the whole exploration:

$$f_{F_i}(q) = \exp\left(-\frac{\|q-\overline{q}_i\|^2}{2\sigma_i^2}\right). \tag{2.30}$$

Conversely, the time-varying Gaussian function decreases over time: for this purpose, we make use of a *forgetting factor* $k_j(t) \in [0,1]$, which gradually decreases the intensity of the function over time and is defined as

$$k_j(t) = 1 - \frac{t - T_{0_j}}{T_{forget}}, \tag{2.31}$$

where T_{0_j} is the time instant at which the Gaussian component is added and T_{forget} represents how long the component should remain active. Thus, the time-varying Gaussian

function can be defined as

$$f_{V_j}(q,t) = \min\left(\exp\left(-\frac{\|q-\overline{q}_j\|^2}{2\sigma_j^2}\right), k_j(t)\right) \tag{2.32}$$

where the *forgetting factor* $k_j(t)$, decreasing over time, lowers the peak of the probability function until completely nullifying it at $t = T_{0_j} + T_{forget}$.

The definition of two different functions allows to specify whether an explored area needs to be crossed again multiple times or instead is needed to be visited only once, depending on the specific application. This differentiation is carried out marking explored areas that don't need to be visited again with the function $f_{F_i}(q)$, while areas required to be crossed multiple times are marked with the function $f_{V_j}(q,t)$: time variance decreases the intensity of the Gaussian function that models the area over time, allowing a specific region to be visited again in the future by the system. In this way, the robots will continue monitoring the environment even once the exploration is completed, keeping the gathered information up to date. Moreover, time variance is useful to avoid local minima which cause the robots to get stuck, making it difficult to achieve an entire exploration of the environment. Every robot stores two lists of Gaussian functions, time-varying and time-variant functions, whose combination generates the density distribution perceived by the robot and exploited to compute the control input. During the exploration process, the lists of Gaussian functions are shared among the robots' network. Each robot, during the motion, computes the Gaussian components and communicates their parameters to the network. Every robot's control input is continuously updated following the changes on the shared density distribution. The combination of the various different Gaussian components is computed as follows:

$$\phi(q) = c + m + 1 - \sum_{i=1}^{c} f_{Fi}(q) - \sum_{j=1}^{m} f_{Vj}(q), \tag{2.33}$$

where c is the number of constant density functions f_{Fi} and m is the number of time-varying density functions f_{Vj}. The presented formulation generates a density distribution ϕ that has smaller values in areas already visited. The main objective for each robot is to add Gaussian components while exploring the environment decreasing the intensity of the density distribution around the recently visited areas in order to drive the robots' attention towards more interesting areas. The density distribution ϕ is then used, following the control theory shown in Section 2.3, to determine the control input for each robot.

Every robot determines the addition of a new Gaussian component according to Algorithm 1. The main idea of the algorithm is to add new Gaussian functions in order to keep the robot in motion. A new component is added whenever a robot's position is sufficiently far from the mean point of existing components: the threshold is set equal to the standard deviation σ. The standard deviation σ can be seen geometrically as the width of the Gaussian function and is set to be half of the robot's sensing range r: $\sigma = r/2$. In a normal distribution, the 95.45% of the samples are inside a 2σ radius from the center point, therefore the value $\sigma = r/2$ reflects the information that is collected by

Algorithm 1: Exploration Algorithm

input : Robot's position $\mathbf{p}$
Time delay Δ
List of probability functions $f_V(q)$ and $f_F(q)$ with mean points $\{\overline{q}_1, \ldots, \overline{q}_{m+n}\}$ and variance σ^2
output: Updated probability distribution

```
begin
    if (‖p − q̄_i‖) > σ, ∀i ∈ [0, (m + n)] then
        if robot is moving then
            Add a new time-varying gaussian component with mean point
             corresponding to the position of p Δ seconds earlier
        if robot has stopped then
            Add a new time-varying gaussian component with q̄ in the most
             recent robot position that is at a distance ≥ 0.75σ from p
    if robot lies in an area that needs to be visited only once then
        Add a time-constant gaussian component with q̄ = p and σ = r/2.
```

the robot in its sensing range. It is worth noting that new components are not added in the actual position of the robot: in fact, this might lead to the definition of symmetric probability distributions, that would lead to have the centroid of the region in the same position as the robot, resulting in a null control input calculated in (2.12). To avoid this, the new components are centered in a previously occupied position: in particular, this delay Δ is calculated as the time needed for a robot to travel a distance of σ with maximum velocity v_{MAX}:

$$\Delta = \frac{\sigma}{v_{MAX}} \tag{2.34}$$

Moreover, if a robot gets stuck, a new component is added to introduce a perturbation in the probability distribution of the surrounding environment, moving the centroid in a different position. The result of using Gaussian functions described above is the definition of a probability distribution indicating whether an area has already been explored or not: an example of such a function is depicted in Fig. 2.23.

2.5.4 Simulations and Experiments

A series of rigorous simulations has been conducted to have a statistical analysis of the result and the performance of the proposed control strategy. Subsequently, several tests on real hardware robots have been carried out to validate the control algorithm.

Implementation

The proposed algorithm has been implemented using ROS2 [94], and has been validated by means of extensive simulations and real-world experiments. In particular, several simulation environments have been created using the Gazebo simulator, representing wide areas populated by a large number of robots.

Two series of rigorous simulations have been conducted, the first one aiming for a sta-

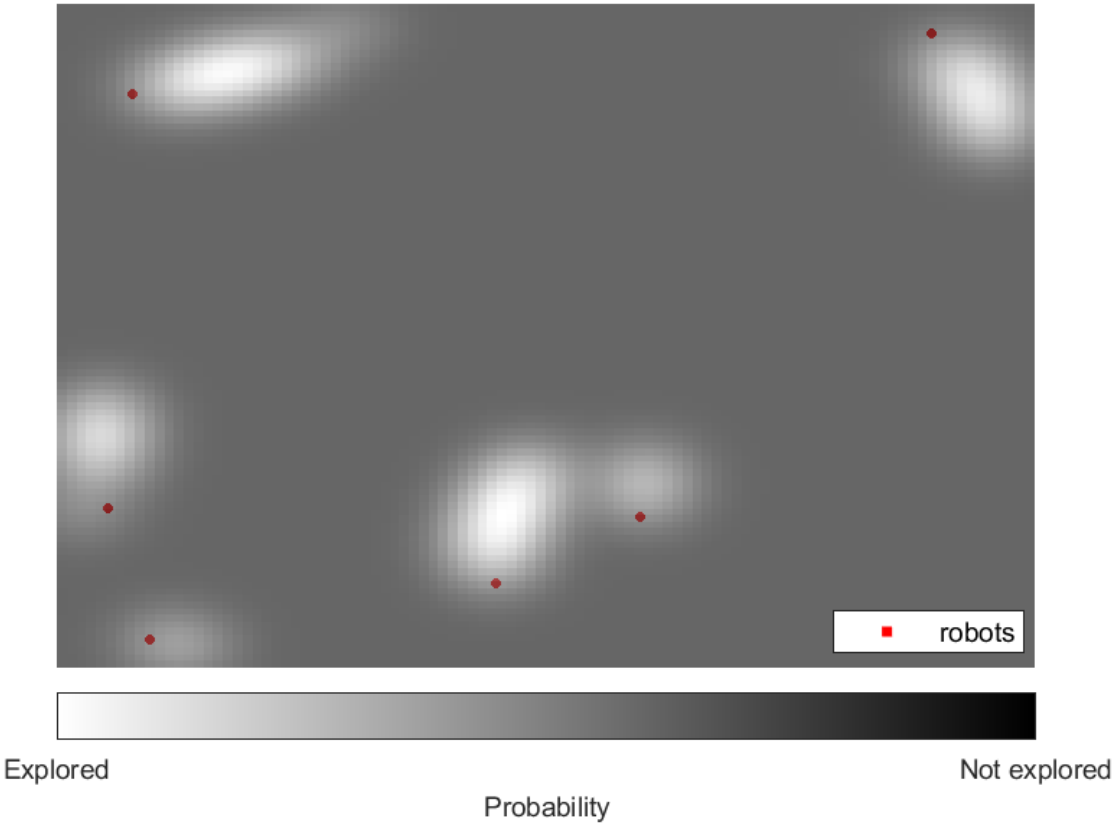
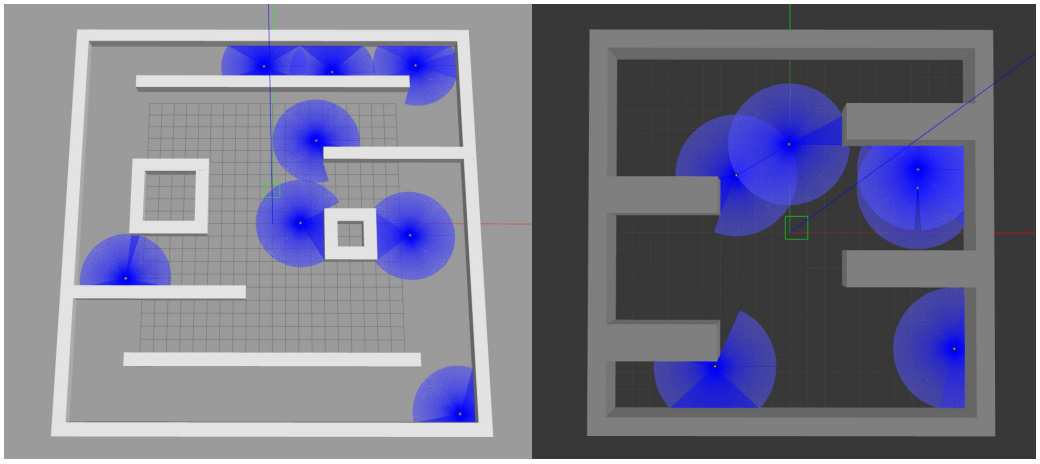


Figure 2.23: Probability distribution of the environment

tistical analysis of the results and the performance of the proposed control strategy, the other one to make a comparison with other strategies found in literature. As for the first series of simulations, the initial number of robots was randomly chosen, as well as their starting location in the environment. The simulations conducted include configurations with 2, 4, 6 and 8 robots with a sensing range $r_{sens} = 4m$ and a maximum linear velocity $v_{max} = 0.5\ m/s$.

Moreover, various environment dimensions have been tested along with different environment configurations and different obstacles locations. In particular, we considered non-convex polygonal environments with an area of $200m^2$, $400m^2$, $900m^2$ and $2500m^2$. Two of the scenarios tested in the simulations are shown in Fig. 2.24: they differ in number of robots, environment dimension and complexity. For each experimental scenario, we ran 20 tests of 5 minutes of simulation: overall 320 tests were conducted. For each experiment we saved the robots locations over time, useful to compute the portion of the covered surface.

The same statistical approach was followed to make a comparison with other strategies. In this case, our solution was compared with the classic potential field navigation algorithm [76] and a random walk solution presented in [95], where robots drive with a constant straight motion, only changing their direction when an obstacle is detected. As before, the same robots were used in teams of 6, 8, 10 and 12 agents placed in random starting positions within the same $2500m^2$ environment already implemented. 20 simulations were run for every team configuration, for a total number of 80 simulations for each method.



(a) 8 robots monitoring a $900m^2$ environment. **(b)** 6 robots monitoring a $400m^2$ environment.

Figure 2.24: Two of the scenarios tested in simulation exploiting Gazebo environment. The two experiments differ in number of robots, environment dimension and complexity. The blue circles represent the sensing areas of the robots.

We also conducted experiments on real robotic platforms exploiting the ROS2 framework. The tests took place in a limited size environment with polygonal obstacles, using three Turtlebot3 Burger differential drive robots (see Fig. 2.25). The robots localization with respect to a global reference system was obtained through the Optitrack motion capture system. Every robot, following the control algorithm, shares the locally added Gaussian function among the robots. We conducted several tests randomly choosing the initial locations and orientations of the robots. Both in simulations and real world experiments, information defining the Gaussian functions parameters were shared among the agents using a central storage unit, where each robot is able to both save and retrieve data. In addition, we considered non circulating areas such as corners or dead end streets as places that need to be visited only once, marking them with time-constant Gaussian functions $f_{F_i}(q)$ defined in (2.30). Assuming that every robot is capable of detecting corners in the environment adopting a state-of-the-art strategy such as the one proposed in [96], it's possible for the agents to understand when a non circulating area is visited and consequently define a new time-constant Gaussian function $f_{F_i}(q)$.

Results and Analysis

As previously mentioned, a large number of simulations were executed in order to have a rigorous statistical analysis of the results. We show how the performance of the control strategy depends on the starting number of robots and is not affected by the initial environment configuration or initial robots locations. In all the experiments and simulations, the multi-robot system was able to entirely explore the unknown non-convex environment. In particular, the percentage of the covered area during the exploration increases with the number of robots used. Figure 2.26 focuses on analyzing the results from the simulations conducted on the environment shown in Fig. 2.24b, which has an area of $400m^2$. From the results in Fig. 2.26, we can conclude that 4 robots are sufficient

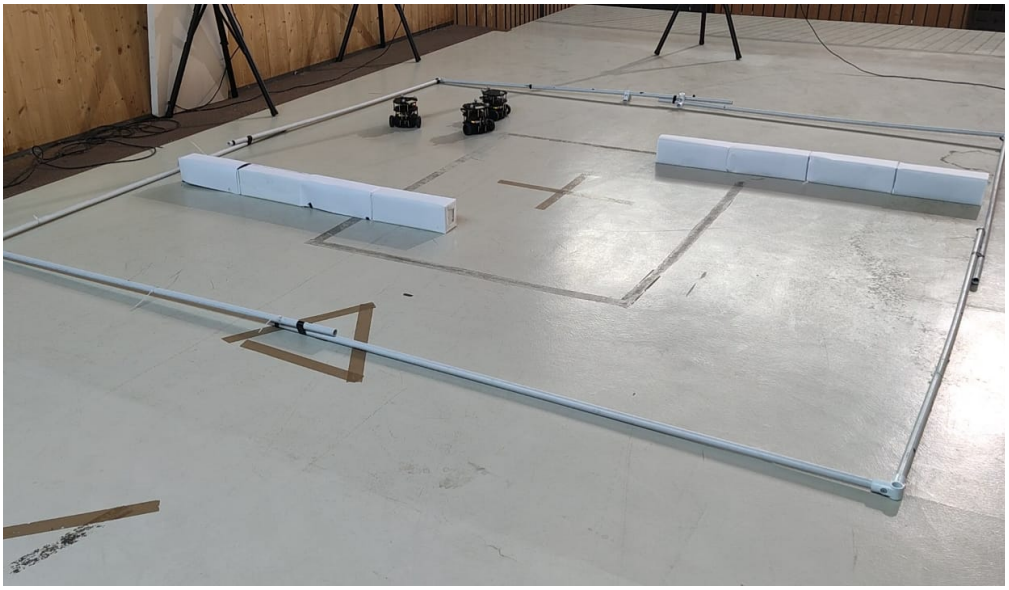


Figure 2.25: The experimental set-up of one of the experiments conducted with real robotic platforms. Three Turtlebot3 burger robots were tasked to explore an unknown non-convex area.

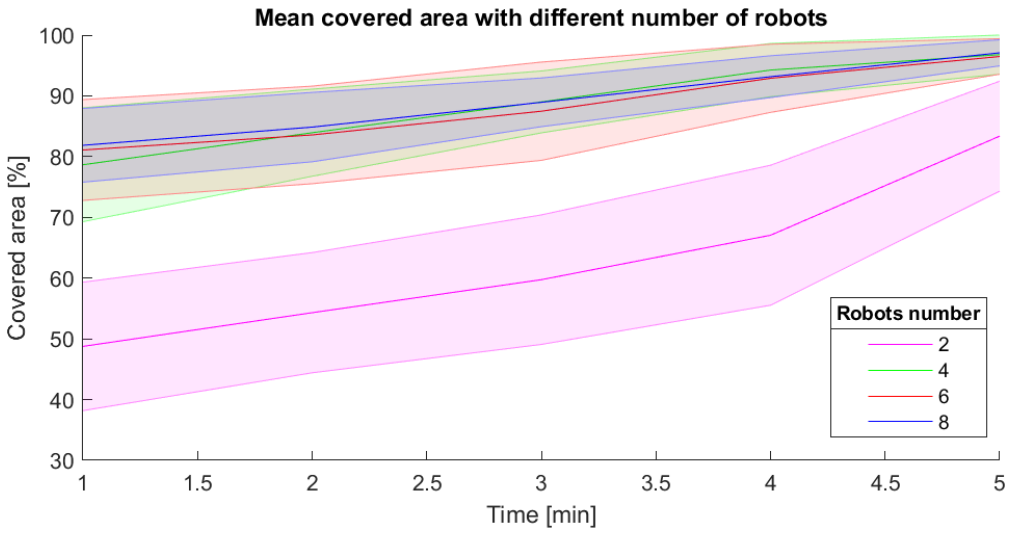


Figure 2.26: Explored area in $400m^2$ environment

to successfully explore the desired environment, while slightly better performances are obtained with 6 and 8 robots. In particular, the results show that a team of 2 robots was able to explore, on average, 65% of the total area. While, a team of 4 robots was able to averagely explore the 84% of the total area and almost the 90% in rare cases. Finally, a team of 6 and 8 robots several times explored almost the whole environment and averagely the team explored the 87% of the total area.

Figure 2.27 instead shows the comparison between the results obtained with our solution and other strategies. The advantages with respect to the Potential Field navigation are clear: this solution had a poor performance because of the large environment used, which results in a great number of local minima where robots got stuck. The comparison with the Random Walk approach shows that our solution can potentially achieve a better performance: the multi-robot system was able to explore a larger region of the environment thanks to the ability to coordinate the agents in an optimal way.

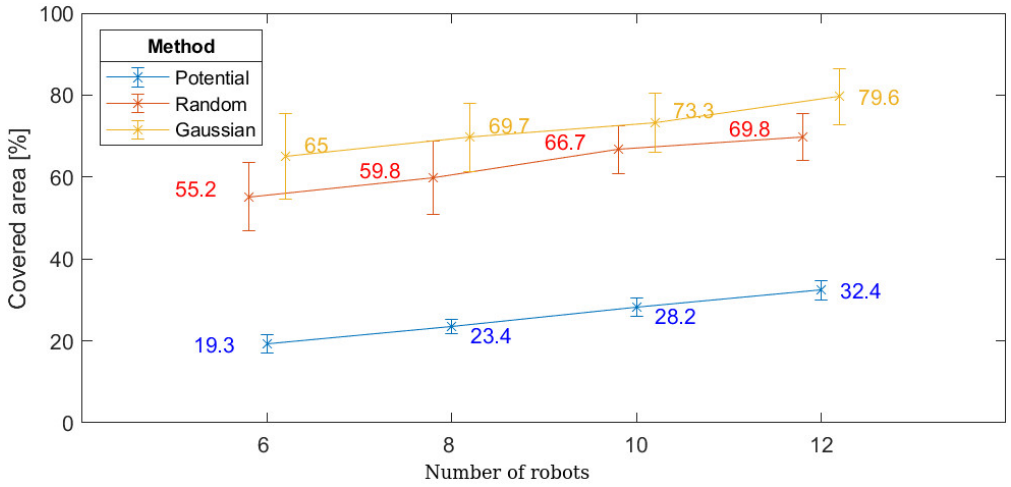


Figure 2.27: Comparison with other solutions

Finally, real-world experiments have also proven very good exploration of the environment: in the example in Fig. 2.28, where 3 robots were employed, the covered area reached 90% of totality. This value is perfectly in line with expectations from simulated trials, because the total area is much smaller than the one in simulated environments, but less robots were employed.

2.5.5 Discussions

In this work we presented a distributed coverage based control to coordinate a group of autonomous robots with limited sensing capabilities with the aim to explore an unknown environment. The proposed control strategy exploits a density distribution to drive the multi-robot system towards areas of the environment to be explored. The density distribution is build from the combination of multiple Gaussian functions placed by the robots in the environment during the motion. We show how the full exploration of the unknown environment can be achieved without a discretization of the environment through graphs or grid cells differently from the majority of the studies found in literature. Moreover, the robots need to store and share among the network only the Gaussian parameters, which require a low communication volume and a low memory consumption. Several experiments and simulations were performed on a group of mobile robots to validate the performance of the proposed control method and make a comparison with other solutions found in literature.

2.6 Coverage Control for Time-Varying Spatial Processes

In Section 2.5, we focused on the problem of coordinating a team of robots with the task to explore (and map) an unknown environment. The aim was to continuously monitor and explore the environment simultaneously maximizing the coverage of the domain. The study presented in this section aims to explore and estimate a feature or spatial process spread into the environment and to use this estimate to optimally cover the environment. We exploit the coverage control approach to coordinate the team of robots according to

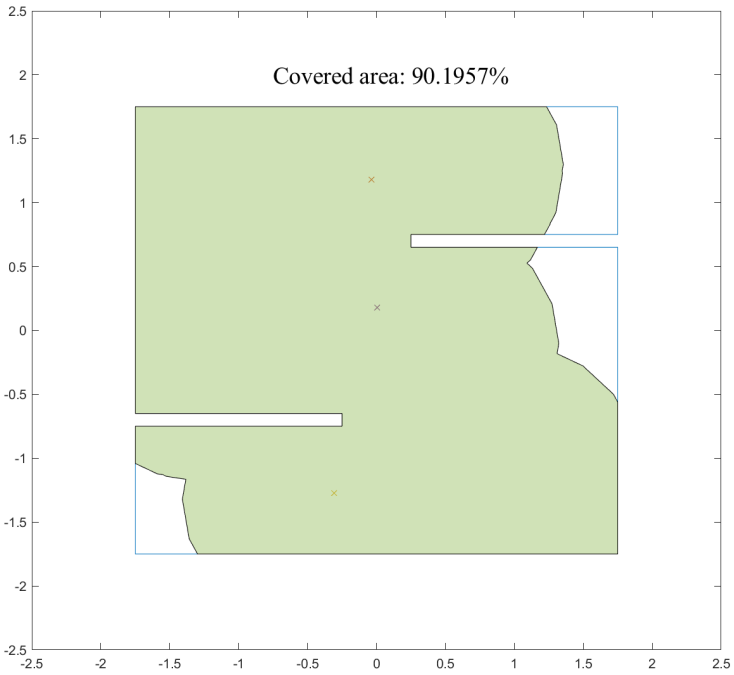


Figure 2.28: Explored area in field trials

a density distribution, which is used to model the spatial process in the environment.

In other words, this work addresses the problem of deployment a multi-robot team to optimally cover a domain where the density distribution, typically describing the interesting areas in the coverage problem, is initially unknown. We propose a coverage based control algorithm to coordinate a group of autonomous robots with the aim to simultaneously learn and estimate a spatial field over a domain. The control strategy exploits the Gaussian Process Regression (GPR) to build a model of the monitored spatial process in the environment. The multi-robot team, finally, distributes in the environment to optimally cover the estimated process. Hence, we propose an adaptive strategy that leads the multi robots team to an optimal solution to the coverage problem that efficiently balance the trade-off between the need to learn the spatial field and the need to optimally cover the estimated spatial field. Moreover, the proposed methodology is able to deal with time-varying spatial fields, which are modelled as distribution that change over time. The trade off between exploration of the spatial process and the exploitation of the estimated spatial process for the coverage control is continuously updated over time in order to keep up with the changes of the spatial process. The proposed strategy efficiently manage the set of older and newer samples to properly provide an estimation of the time-varying spatial field. Finally, the proposed strategy is able to deal with the well-known computational limits of the Gaussian processes. The control algorithm manages the set of sample limiting the GP training data to the ones that are relevantly improving the process estimation. The performance of the proposed algorithm is evaluated in several rigorous simulations.

2.6.1 Literature Review

Coverage control and autonomous exploration are important applications of multi-vehicle systems, where a team of networked robots are coordinated to explore an unknown environment or to maximize the coverage of an unknown environment, affording higher concentrations of robots in more interesting areas. These techniques have earned significant research interest due to their usefulness in search and rescue [97], precision agriculture [22, 98], environmental monitoring and exploration [26, 99, 100], localization and mapping [101], etc.

In this study we are interested in the spatial coverage problem for limited range multi-robot systems (see Sections 2.2 and 2.3). This approach, as previously described, based on the Voronoi partitioning of the environment, guarantees the convergence of the networked robots to a configuration that maximizes the coverage of the most important areas of the environment. These areas are encoded as a probability density function defined over the environment. Various coverage control strategies have been studied in the last years that address heterogeneous and limited robots capabilities [63] or consider various scenarios associated with different density functions [102, 103]. However a common assumption is to consider the density function known beforehand by each of the robot involved. This is not applicable when the multi-robot system is tasked to operate in an unknown or partially known environment. The problem of controlling robots for sampling data to estimate a process or a distribution in the environment is referred to as informative sampling [104] of information gathering [105]. These algorithms aim to efficiently obtain an accurate reconstruction of a physical process. Many recent works have proposed algorithms that employ Gaussian Processes (GPs) to efficiently obtain an accurate reconstruction of a physical process. An example is presented in [106], where a random based path planner allows a single robot to identify unvisited locations: moving in the environment the robot, learns the GP parameters. A relevant portion of the active sampling studies concerns the monitoring of processes in ocean environments. The aim is to monitor marine features (turbidity, dissolved oxygen, salinity, etc.) to evaluate the condition of the water. A control strategy to generate the trajectory for a single robot is generally studied to maximize the information gathering of relevant data [107, 108, 109]. A multi-robot approach has been presented by [110], which is based on a discretization of the environment exploiting Voronoi partitioning. Alternatively, the work presented recently by [111] coordinates a group of robots scheduling meeting points to exchange data. The majority of the works found in the literature about information gathering and active sampling are developed for a single robot application. Moreover, the process estimation is usually obtained by using the Gaussian Processes and, most of all, in all these studies the estimated and monitored process is considered to be time invariant.

In this work we investigate the coverage based control combined with a sampling strategy that estimates the unknown density function. The control strategy for a multi-robot system is hence updated accordingly to on-line observed sample data. Several works have been recently dedicated to the problem of multi-robot optimal coverage with the aim of cooperative exploration and estimation of unknown processes [112]. In [113, 114] the authors propose two sampling strategies that enable the multi-robot team to optimally

learn the spatial field and optimally cover it. Differently, in [115, 116] the robots are coordinated to optimally cover the spatial field, which is estimated from the data sampled along the way without an exploration strategy. Some works are more focused on a real world implementation without considering an optimal strategy to explore and cover the spatial process with a multi-robot system [117, 118]. Other approaches divide the control strategy in two different phases, an initial optimal exploration phase to estimate the density function and a final exploitation phase where the robots use the density function to optimally cover the environment [119, 120]. Hence, the problem of optimally coordinating a group of robots to estimate a spatial field and to optimally cover it at the same time remains an open problem under investigation. Moreover, to the best of the authors' knowledge, all the control strategies proposed in the literature are built on the assumption that the spatial process to be estimated remains constant over time. This is a strong assumption for real monitoring applications. The real world examples considered in the mentioned studies, such as the monitoring of pollution concentration, salinity, dissolved oxygen, etc., are variables that change over time. We propose a methodology that allows the multi-robot system to optimally learn and estimate the unknown process and uses this information to obtain a coverage of the environment influenced by a non-uniform density distribution. Differently from the works mentioned, the algorithm we propose addresses the estimation and the coverage of the spatial process in the environment simultaneously. Moreover, differently from [113], this trade-off between exploration and exploitation of the process is time independent and is influenced only by the uncertainties over the density function encoding the process. This allows us to consider spatial processes that change over time. We finally get rid of the assumption that the spatial process is time invariant and, hence, we have a density function that has features that are changing over time.

2.6.2 Notation and Coverage Control

The control strategy presented in this chapter has been built upon the solution to the coverage problem for the multi-robot system control described in Section 2.3. In this section, we introduce a slight modification to the notation used, aimed at streamlining the mathematical representation for the purpose of facilitating the discussion.

We consider an arbitrary point in Q denoted by $x \in Q$. Let then $\mathcal{P}$ be a set of n points $\{p_1, \ldots, p_n\}$ in Q. Therefore, we have that the performance function introduced in Section 2.3 indicates how reliable the measurement is at point $x \in Q$ of the sensor i located in p_i as a function of the distance $\|x - p_i\|$. Finally, given the limited Voronoi partitioning $\mathcal{V}(\mathcal{P})$ of the environment Q in Voronoi cells $\{V_1, \ldots, V_n\}$, the optimization function $H : Q \to \mathbb{R}$ becomes:

$$\mathcal{H}_{\mathcal{V}}(\mathcal{P}) = \sum_{i=1}^{n} \int_{V_i(\mathcal{P})} f(\|x - p_i\|)\phi(x)dx, \tag{2.35}$$

2.6.3 Problem Statement

As discussed in Section 2.2.1, we consider a multi-robot system constituted by n robots that move in a 2-dimensional space. We assume each robot to be modeled as a single integrator system according to (2.2): where each robot position $p_i \in \mathbb{R}^2$ evolves according to $\dot{p}_i = u_i$, with $u_i \in \mathbb{R}^2$ the control input, $\forall i = 1, \ldots, n$. The set of robots is represented by $\mathcal{P} = \{p_1, \ldots, p_n\}$.

We consider the following setting:

1. *Convex unknown environment*: the multi-robot system has to maximize the coverage of an unknown spatial field distributed in the environment, a convex polytope Q.
2. *Time varying spatial field*: The multi-robot system has to continuously learn and update the estimate of a spatial field that changes over time.

We consider the following assumptions:

1. *Limited sensing capabilities*: each robot is able to measure the position of neighboring robots and objects, to detect the boundaries of the environment Q within its limited sensing range. This allows the computation of the Limited Voronoi partitioning of the environment as defined in (2.10).
2. *Communication capabilities*: The team of robots is connected through a communication network, enabling them to exchange information and data.
3. *Spatial process behaviour*: The behavior of the spatial process in the environment exhibits a smooth characteristic, making it suitable for modeling through the use of a Gaussian Process with a Squared Exponential Kernel.

The problem addressed in this paper is then formalized as follows:

Problem. *Define a distributed control strategy that allows a multi-robot system to optimally explore the environment, to learn and estimate the unknown time-varying spatial field and to simultaneously perform coverage of the spatial field in the environment Q.*

2.6.4 Gaussian Process Regression

As mentioned before, in this work we get rid of the assumption that the density function $\phi(q)$ in (2.35) is known beforehand. For this reason, we need to define a strategy that allows the robots to learn and estimate online the spatial process from the data sampled during the motion. We assume each robot is provided with a sensor device able to detect and measure at its location features of the spatial process in the environment. Gaussian processes are a powerful tool that can be used to model an unknown function starting from a few samples and have the ability to provide the uncertainty over the predicted values of the function [121]. In particular, a GP defines a prior distribution

over the space of functions such that, together with the observed data, can lead to a posterior multivariate Gaussian distribution able to predict the function behaviour. Let us consider a one-dimensional scenario to simplify the notation, where an arbitrary point Q denoted by $q \in Q$ can be defined by x, and a set of n points in the domain can be defined by $X = [x_1, \ldots, x_n]$. Subsequently, the same notation can be easily used for a multi-dimensional approach.

A GP is completely specified by its mean function $\mu(x)$ and covariance function $k(x, x')$. Let $\phi(x)$ be the environmental spatial process that the Gaussian process has to model, then, $\mu(x)$ represents the expected value of ϕ at input x, and $k(x, x')$ represents the linear correlation between two variables x and x'. The sensory observation of the spatial process taken by the robots has a white Gaussian noisy measurement:

$$y = \phi(x) + \nu \tag{2.36}$$

where $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$. In Gaussian process regression, we assume the function $\phi(x)$ can be estimated as a Gaussian process:

$$\phi(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{2.37}$$

For notational simplicity and to reduce computational complexity we consider a zero-mean GP. A GP with a mean $\mu(x) \neq 0$ can be treated by a change of variables. The learning and the estimation of the process is defined by the so called *kernel* $k(x, x')$. The kernel has to be chosen according to the type of the function we want to estimate, e.g. periodic, linear, quadratic, etc. Since we are assuming the spatial process has a smooth and stationary behaviour we consider a squared exponential kernel:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{\|x - x'\|^2}{2\lambda^2}\right), \tag{2.38}$$

where λ and σ_f are hyper-parameters which can be estimated from the sampled data by maximizing the likelihood function. We have the observations $\mathcal{D}_t = \{X_t, y_t\}$ collected from the robots team until time t and we want to predict the values of the spatial process in the new locations $X^* = [x_1^*, \ldots, x_n^*]$. Hence, we can define the covariance function $k(X^*, X^*)$ and the predicted values $\phi^* = [\phi(x_1^*), \ldots, \phi(x_n^*)]$.

Exploiting the Bayesian theorem, as detailed in [121], we can compute the conditional distribution of ϕ^*:

$$(\phi^* | X_t, y_t, X^*) \sim \mathcal{N}(\mu(X^* | D_t), \Sigma(X^* | D_t)), \tag{2.39}$$

which is a multivariate normal distribution with mean:

$$\mu(X^*) = k(X^*, X_t)^T [k(X_t, X_t) + \sigma_\nu^2 I]^{-1} y_t \tag{2.40}$$

and covariance matrix:

$$\Sigma(X^*) = k(X^*, X^*) - k(X^*, X_t)^T [k(X_t, X_t) + \sigma_\nu^2 I]^{-1} k(X^*, X_t), \tag{2.41}$$

where $k(X^*, X_t)$ is a covariance vector. The spatial process estimation is associated with mean of the distribution, while the uncertainty of this estimation is described by the covariance matrix. The process predicted values are strongly influenced by how similar are the observed data X_t to the points X^* that we want to predict. This correlation is described by the kernel $k(X^*, X_t)$.

It is worth noting that the equations (2.40) and (2.41) require an inversion of a covariance matrix, which imply a computational complexity of $O(N^3)$. Hence, the computation can be a problem when the set of samples collected during the robots exploration becomes large [122, 123], in particular if the control algorithm is executed on board the robots.

The kernel hyper-parameters $\lambda, \sigma_f, \sigma_\nu$ are unknown and need to be inferred from the sampled data. The estimation of the hyper-parameters can be optimized by maximising the marginal (log) likelihood function. Given the observation $\mathcal{D}_t = X_t, y_t$ and the hyper-parameters $\theta = (\lambda, \sigma_f^2, \sigma_\nu^2)$, the log marginal likelihood function can be expressed in closed form:

$$\log p(y|X, \theta) = -\frac{1}{2} y^T K_y^{-1} y - \frac{1}{2} \log |K_y| - \frac{n}{2} \log 2\pi \tag{2.42}$$

where $K_y = K(X, X) + \sigma_\nu^2 I$. The maximum likelihood solution is normally obtained through a gradient-ascent based optimization and improves as more observations are added. For more details the reader is referred to [121, 124].

2.6.5 Exploration - Exploitation Problem

The aim of this work is to efficiently explore the unknown spatial field through the Gaussian process regression and to use this information to optimally cover the environment. We want the robots *exploit* the information from the GP to concentrate the coverage over the most relevant areas of the environment. Equation (2.39) gives the distribution of the density function, which encode the estimation (the mean function) and the uncertainty over the estimation (the covariance function). In this study we aim to combine these information with the coverage control to make each robot to both learn the field and cover it. In the Bayesian optimization (BO) an *acquisition function* is used as the key to the balance between exploration and exploitation [121]. An acquisition function combines the mean and the variance into a criterion that produces the highest utility, in terms of learning the function or using the function information. The multi-robot system uses in the coverage based control a substitute of the density function that coincides with an acquisition function. The strategy we used is the Upper Confidence Bound (UCB) algorithm [125], that uses the following acquisition function:

$$\phi'_t(x) = \sigma_{t-1}(x) + W_t \mu_{t-1}(x), \tag{2.43}$$

where $\sigma_{t-1}(x) = \sqrt{\Sigma_{t-1}(x, x)}$ is the standard deviation. The substitute density function $\phi'_t(x)$ aims to adapt the coverage control strategy to push the robots to the high utility areas. The utility distribution in the environment is more weighted towards the exploration if the density function is completely unknown ($\phi'_t(x) \sim \sigma_{t-1}(x)$) and is more weighted towards the exploitation is the density function has been sufficiently already

explored ($\phi'_t(x) \sim \mu_{t-1}(x)$). The weight W_t is chosen according to the preference over the trade off between exploration and exploitation. High values of W_t mean that the exploration is preferred over the exploitation:

$$W_t = C_1 \arctan(C_2 t), \tag{2.44}$$

where C_1 and C_2 are two constants, chosen by the user to tune the exploration-exploitation trade off according to the monitoring scenario. These constant can be learned similarly to the GP hyper-parameters, but this procedure is left as a topic for future works. The intuition behind the equation (2.44) is that we want the balance between exploration and exploitation to be generally independent from time (differently from [113]), while a dependence from time is used only at the beginning of the experiment where certainly the density function is unknown and the exploration is preferred over the exploitation. In this way the robots team generally prefers to exploit the information from the spatial field model for the coverage control only if the uncertainty over the estimation is sufficiently low. This feature of the control strategy becomes fundamental when dealing with time-varying spatial field, where the uncertainty can change over time.

2.6.6 Data Sampling Strategy

The Gaussian Process regression algorithm has $O(N^3)$ time complexity and $O(N^2)$ memory complexity, where N is the number of sampled data. The computational complexity is defined indeed by the inverse of the covariance matrix with size $(N \times N)$, as shown in (2.41). It is known that, for large training sets, the Gaussian Process problem computation becomes intractable on the current hardware. It becomes, hence, necessary to limit the data sampled by the robots during the exploration, with the aim to train the GP on the minimum amount of information required to successfully estimate the spatial process. In other words, the main question we need to answer is how frequently the robots need to sample the environment. In this work, we implemented a policy that filters the samples and elaborates the training set in order to keep the data that effectively improve the estimation and prediction of the spatial process. The samples and the data that are not or are slightly improving the GP model are discarded. During every robot movement in the environment, a new sample is added into the training set if the uncertainty of the estimation is sufficiently high of the considered location and hence the new data would considerably improve the model and reduce effectively the overall uncertainty. The issue becomes to define, in the interested location, the threshold where the uncertainty is considered to be high. Given the Z-score z_N corresponding to a predefined desired confidence interval, from the Normal Gaussian distribution, and the maximum error tolerance around the process estimation e_r, we have the threshold over the uncertainty and standard deviation given by $\sigma_{th} = e_r / z_N$. Meaningful values usually taken for these parameters are $z_N = 1.96$, which corresponds to a 95% confidence interval, and $e_r = 0.05$, which corresponds to a maximum error of 5% around the estimated signal [126, 127]. Hence, we have that the maximum allowed standard deviation value is $\sigma_{th} = 0.0204$ relatively to the mean estimation. Given a new sample p and the respective standard deviation of the estimated signal σ_p, we have that the sample is added to the

training set if the following condition is satisfied:

$$\sigma_p \geq \sigma_{th} * \mu_{max}, \tag{2.45}$$

where μ_{max} is the highest value reached by the estimated spatial signal. This condition determines, during the robots exploration in the environment, if the new data sampled is in a location that can considerably improve the signal estimation. We remark that the uncertainty is reduced to zero in the location where the data is sampled. In Section 2.6.7 we present the time varying approach to model time varying spatial processes and we shown how the uncertainty of the estimation increases over time. Following the criteria described by (2.45), we delete from the training samples all the data that generates a too high uncertainty, pushing the robots to take new samples. The threshold over the uncertainty is defined by the parameter σ_{th} and describes the error that is tolerated on the estimated signal around the sampled data.

2.6.7 Time Varying Spatial Field

In this work we consider a scenario in which the multi-robot system has to monitor a time varying spatial field or a scenario in which the robots are dropped in a non-stationary environment. The majority of applications for monitoring purposes have to deal with this problem, which has never been considered in the multi-robot research field, to the best of the authors' knowledge.

The control strategy allows the robots team to estimate and then to optimally cover the spatial process. During the estimation phase, the aim is to reduce the uncertainty over the model of the density function, which is represented by the standard deviation in (2.43). Hence, once this term is minimized the major component in the control is defined by the mean of the GP distribution, which brings to a exploitation phase and the robots team converges to the optimal configuration that maximize the coverage of the spatial field. In an stationary environment with a constant spatial field, the group of robots keeps its configuration that optimally cover the environment to monitor the most relevant areas. Hence, the control algorithm generally consists in an exploitation phase that follows the exploration phase. However, in a scenario where the spatial process changes over time, the observed data gets old, the uncertainty of the model increases and a new exploration is necessary to updated the process estimation. Hence, the robots are pushed to constantly sample the environment in order to detect any change in the spatial field and simultaneously are moved to optimally cover the interested areas of the environment exploiting the process estimation. The control algorithm needs to continuously deal with the balance between the exploration and the exploitation phase.

Taking inspiration from [128, 129], we propose a novel control strategy that takes into account how the uncertainty of the estimated density function can increase over time and how the old samples less affect the estimation. Hence, in the time-invariant setting all the previous samples and observations are being equally important, while according to our approach the samples become increasingly stale with time.

Our model is similar to the spatiotemporal kernel model described in [128], but, unlike them, we added an exponential time decay quantity into the control in order to consider how the observations get old and less beneficial for the estimation of the spatial process. In-fact, the model in [128] considers only a relative ageing between the samples (the older samples contribute less than the newer ones) without considering how actually old is every sample.

The mean and the covariance functions respectively from equation (2.40) and (2.41), and used to determine the acquisition function for the coverage control algorithm in equation (2.43), become:

$$\mu(x) = \hat{k}(x, X_t)^T [\hat{k}(X_t, X_t) + \sigma_\nu^2 I]^{-1} y_t \tag{2.46}$$

and covariance matrix:

$$\Sigma(x, x') = k(x, x') - \hat{k}(x, X_t)^T [\hat{k}(X_t, X_t) + \sigma_\nu^2 I]^{-1} \hat{k}(x, X_t), \tag{2.47}$$

where

$$\begin{aligned} \hat{k}(X_t, X_t) &= k(X_t, X_t) \circ D_t \circ T^D, \\ \hat{k}(x, X_t) &= k(x, X_t) \circ d_t \circ T^d, \end{aligned} \tag{2.48}$$

with

$$\begin{aligned} D_t &= \left[(1-\epsilon)^{|i-j|/2} \right]_{i,j=1}^{N}, \\ d_t &= \left[(1-\epsilon)^{(N+1-i)/2} \right]_{i=1}^{N}. \end{aligned} \tag{2.49}$$

Here N is the number of samples collected by the robot team, $\circ$ is the Hadamard product [130] and ϵ is a parameter that defines the strength of the connection between old and new samples. High values of ϵ bring the robots to almost ignore the old data compared to the new ones. T^d and T^D are respectively an array and matrix that describe how the samples and hence the training are ageing over time. The idea is that the older a sample is, the smaller the value in the corresponding entries of d_t and D_t, and hence the less it contributes to the values of the covariance function $\Sigma(x, x')$. Moreover, the older the collected samples are compared with the current time and lower is the value introduced by the time decay component in T^d and T^D. The array T^d is defined as the sequence of N time decay constants computed through the time decay function, as follows:

$$T^d = \left[T_\tau(\Delta t_0), \ldots, T_\tau(\Delta t_i), \ldots, T_\tau(\Delta t_{N-1}) \right], \tag{2.50}$$

where $\Delta t_i = t - t_i$, with t is the current time and t_i is the time at which the sample i is taken, and $T_\tau(t)$ is a predefined time decay function that describes how fast the data sampled at time t ages over time. We generally assume we have an exponential time decay function as follows:

$$T_\tau = e^{-\Delta t/\tau}, \tag{2.51}$$

where τ is a parameter that indicates how steep is the exponential function and hence

how fast we assume the data and the monitored spatial process change over time. By definition, the parameter τ can be understood as the period of time after which the value of a sample in the data set decreases to $1/e \sim 0.367$ times its original value. The higher is the value of τ, the slower the sampled data are ageing. Hence large values of τ are suitable for spatial processes that change slowly over time. Finally, the matrix T^D is defined as follows:

$$T^D_{i,j} = \begin{cases} T^d_i \cdot T^d_j, & i \neq j, \\ 1, & otherwise. \end{cases} \tag{2.52}$$

The result of this learning approach is simply shown is figure 2.29 with the estimation of a 1-D signal. In the time variant approach the samples are not equally considered, differently from the time invariant strategy, and the older the sample the higher is the uncertainty of the respective estimated signal. In the figure we are assuming we can model the signal changes over time with an exponential function as in (2.51). We set the two hyperparameters $\epsilon = 0.001$ and $\tau = 3000$. We have six points which are tidily sampled from left to right every ten seconds. Hence, for example, the first sample has a time decay constant of $T_\tau = e^{-(60-0)/3000} = 0.98$, while the last sample has a time decay constant of $T_\tau = e^{-(60-60)/3000} = 1$.

In the previous Section 2.6.6, we presented a strategy to limit and filter the data gathered by the robots during the exploration. Specifically, in the case of a time-varying spatial process where the uncertainty in estimation increases over time, the usefulness of the collected data decreases accordingly. Following the criteria outlined in (2.45), samples that are outdated and result in an excessive uncertainty are eliminated from the training set, thus pushing the robots to obtain more informative samples.

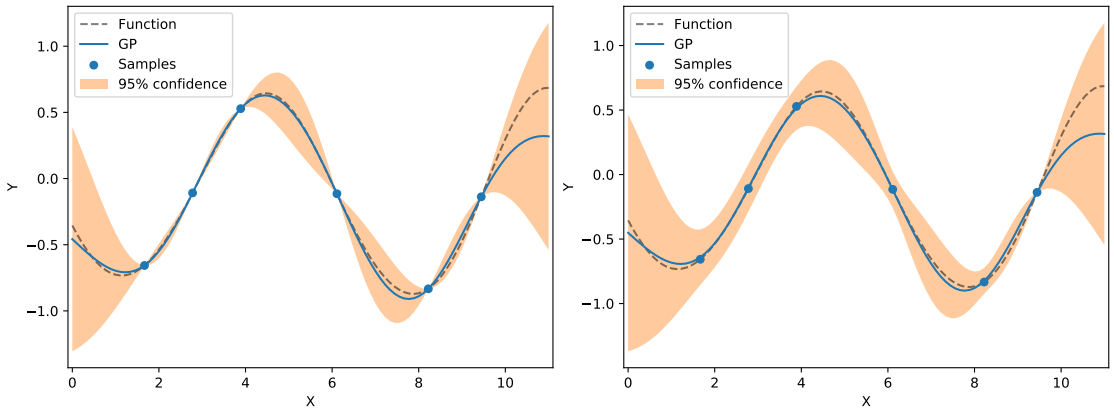
The time decay function topology is a feature that needs to be predefined according to the variability of the spatial process. We generally choose an exponential time decay function to have a control strategy responsive to any process change. However, we did some experiments with a step-like time decay function, emulating a scenario in which the spatial process is strongly affected by a singular event occurring periodically over time. The function is determined by two parameters as follows:

$$T_{\tau_{12}} = -\frac{1}{1 + e^{-\tau_1(\Delta t - \tau_2)}} + 1, \tag{2.53}$$

where τ_1 defines how steep is the step of the function and, hence, how fast it goes from one to zero, and τ_2 defines the time the spatial process changes and the decay function drops to zero.

2.6.8 Experimental Validations

In this section we report some of the simulations we carried out to verify the proposed control strategy.



(a) Time invariant, $t = 60s$.

(b) Time variant, $t = 60s$.

Figure 2.29: Comparison between a time variant and time invariant approach with a 1-D signal estimation. We presume the 6 samples are tidily taken from left to right every 10 seconds. Hence, the older sample is in $x \sim 2$ and the latest is in $x \sim 10$. In the figure (a) with the time invariant approach all the samples are equally processed and the uncertainty of the estimation is zero in the sample locations. In the figure (b) we are using a time-varying learning approach, in which the samples are processed differently depending on the time they were taken. The older samples have greater uncertainties than the newer ones. Moreover the latest sample has a zero standard deviation since it was taken at the current time $t_6 = 60s$ and hence $\Delta t_6 = 0$. The parameters used in this example for the time varying approach are $\epsilon = 0.001$ and $\tau = 3000$.

Simulation Setup

The simulations were built using a combination of Python and Cython and executed on a laptop with Intel core i7-10510U and NVIDIA Geforce MX230. The proposed methodology is strongly sensible on the choice of the hyperparameters and the time variant approach strongly depends on the topology of the time decay function. In summary, the feature and the hyperparameters that need to be defined before starting the control algorithm in a time invariant scenario are:

- the constant values C_1 and C_2 which define the weight W_t and the balance between the exploitation and exploration phases,
- the error tolerance about the signal estimation e_r which defines the uncertainty threshold σ_{th} and the data sampling frequency.

While, the hyperparameters that need to be defined, in addition to the previous ones, in a time variant scenario are:

- the time decay function, e.g. exponential or step-like, which defines how the control strategy reacts to a change in the spatial process,
- the parameters ϵ and τ, which defines how the data samples age over time.

We manage to carry out several simulations to test the time invariant implementation, the exploitation-exploration trade off, and the time variant approach with different time decay function and hyperparameters.

Exploration-Exploitation Trade-off

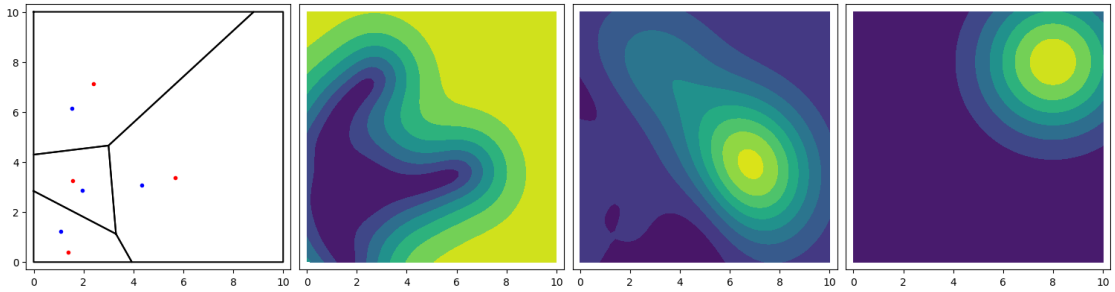
Firstly, we tested the capability of the proposed control strategy to balance the trade-off between the exploration and the exploitation of the spatial process. Figure 2.30 shows one the experiments, where the initially unknown spatial process modelled as a Gaussian-

like function is defined by a mean value $(8, 8)$ and a standard deviation of 2 (the fourth image from left in figures 2.30a, 2.30b, 2.30c). The hyperparameters are generally chosen as $C_1 = 3.2$ and $C_2 = 2$ for this type of simulation: during the initial time steps the exploration of the environment is preferred and subsequently the exploration-exploitation balance is defined by a constant value of W_t. For how long the initial exploration is preferred of the exploitation is decided by the parameter C_2, while the parameter C_1 affects the exploration-exploitation trade-off after the initial transitory. Increasing the C_1 value the exploitation of the estimated signal is preferred over its exploration. In this kind of experiments, since the spatial process is constant over time, once the estimate is sufficiently precise, the team of robots prefer exploiting the learnt density distribution in the coverage control to optimally cover it.

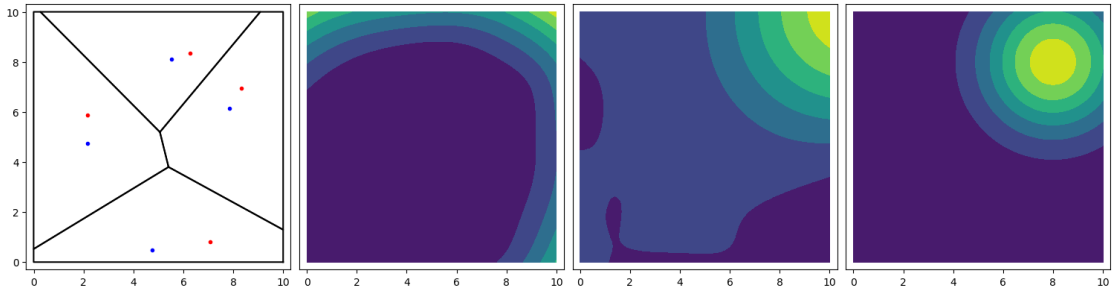
In Figures 2.30a, 2.30b, 2.30c, the first image on the left shows the robots locations (blue dots), the Voronoi partitioning of the environment (black lines) and the computed centroids, where the robots are headed (red dots) according to (2.8), one for each Voronoi cell. The second image from the left shows the standard deviation and uncertainty relative to the estimation of the spatial process given by the Gaussian Process regression as determined by (2.41) or (2.47). The third image from the left shows the mean of the GP and the result of the process estimation as determined by (2.40) or (2.46). The last and fourth image from the left shows the spatial field that the group of robots needs to estimate and to exploit for the coverage based control. The Figure 2.30 shows the whole experiment, where at the very beginning the robots know little about the spatial field and are able to approximate an estimate using only the data sampled from the surroundings. Given the high standard deviation and hence the high uncertainty over the environment the team of robots start an exploration phase. After approximately only 6 seconds, the multi-robot system is able to almost fully model the unknown spatial process. Since the uncertainty is low in the most of the environment the team of robots prefer an exploitation phase over the exploration, moving towards the high values of the spatial field. After less that 25 seconds, the multi-robot system has a reliable estimate of the spatial process. According to the coverage based control strategy the robots move towards the most interested area defined by the learnt spatial field. It is worth noting that the exploration component in the exploration-exploitation trade-off is never zero, since there is always uncertainty at some location in the environment. The trade-off is efficiently balanced so that the multi-robot system moves towards the estimated spatial process, once this is considered sufficiently known.

Time-Varying Spatial Field

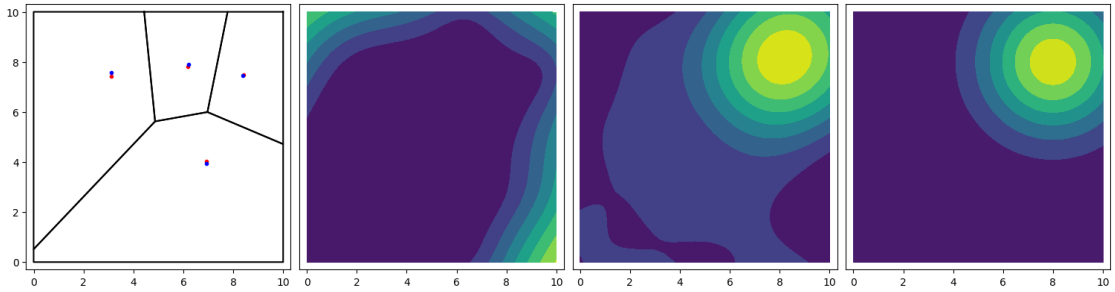
One of the most relevant experiment is the one shown in figure 2.31 where the group of robots and the control strategy have to deal with a spatial process that is changing over time. The spatial process and its changes are unknown and need to be estimated by the group of robots. The spatial process (the fourth image from left in figures 2.31a, 2.31b, 2.31c, 2.31d) consists in a Gaussian-like function initially centered in $(8, 8)$ that moves to $(8, 2)$ location. We generally choose the parameters $C_1 = 2.5$ and $C_2 = 2$: differently from the previous experiments, we shift the balance slightly towards



(a) $t = 1s$.



(b) $t = 6s$.

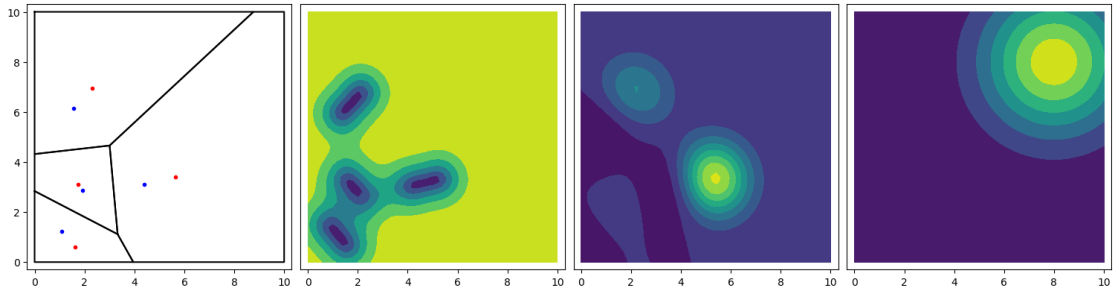


(c) $t = 25s$.

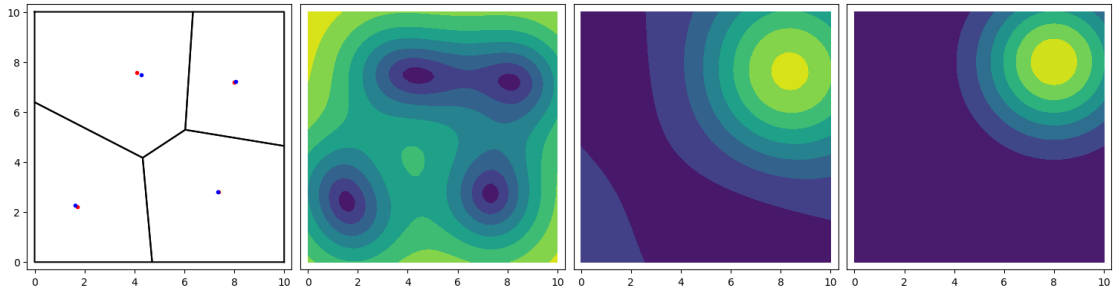
Figure 2.30: The figure shows three different subfigures, one for each time step, of a simulation where the team of robots is tasked to estimate a constant spatial process and optimally cover it. Each subfigure shows four aspects of the simulation: the first image from the left shows the team of robots (blue dots), the Voronoi partitioning (black lines) and the computed centroid for every Voronoi cell (red dots). The second image from the left shows the uncertainty over the domain: yellow areas mean a high standard deviation and a high uncertainty, blue areas mean a low standard deviation and low uncertainty. The third image from the left shows the robots estimation up that instant of time. The fourth image from the left shows the unknown spatial process that the team of robots is tasked to estimate.

the exploration, since we have a time-varying spatial process and the robots need to be ready for any change. Moreover, we generally choose the estimation error $e_r = 0.04$ and exponential decay function like in equation (2.51) with hyperparameters $\epsilon = 0.001$ and $\tau = 1000$ in order to have a function sensible to quick changes of the spatial field in the environment.

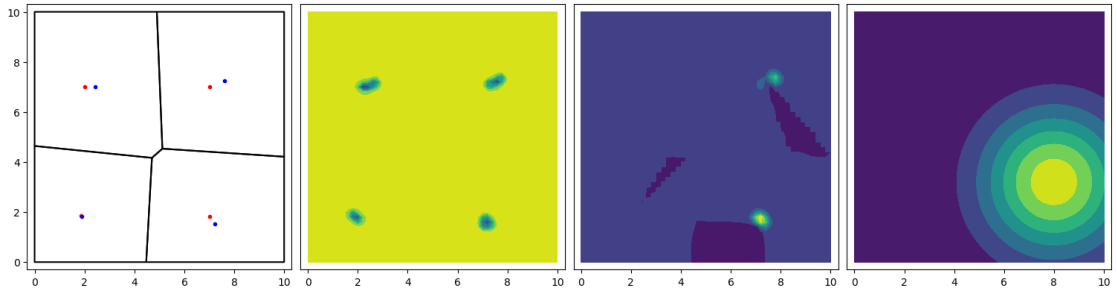
At the very beginning, Fig. 2.31a, the robots know little about the spatial field, the uncertainty is high everywhere in the environment except around the robots locations and the GP is able to generate a rough estimate of the process. After approximately 30 seconds the robots have a reliable estimate of the spatial process, the uncertainty is lower over the environment and the robots converge to a configuration that optimize the balance between exploration and exploitation maximizing the coverage of the environment according to the estimated spatial field. It is worth noting that the uncertainty never goes



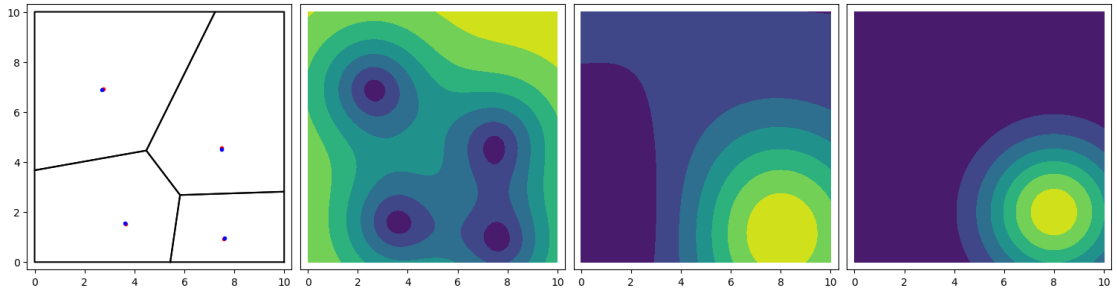
(a) $t = 1s$.



(b) $t = 32s$.



(c) $t = 48s$.



(d) $t = 108s$.

Figure 2.31: The figure shows four different subfigures, one for each time step, of a simulation where the team of robots is tasked to estimate a time-varying spatial process and optimally cover it. Each subfigure shows four aspects of the simulation: the first image from the left shows the team of robots (blue dots), the Voronoi partitioning (black lines) and the computed centroid for every Voronoi cell (red dots). The second image from the left shows the uncertainty over the domain: yellow areas mean a high standard deviation and a high uncertainty, blue areas mean a low standard deviation and low uncertainty. The third image from the left shows the robots estimation up that instant of time. The fourth image from the left shows the spatial process that changes over time and that the team of robots is tasked to estimate.

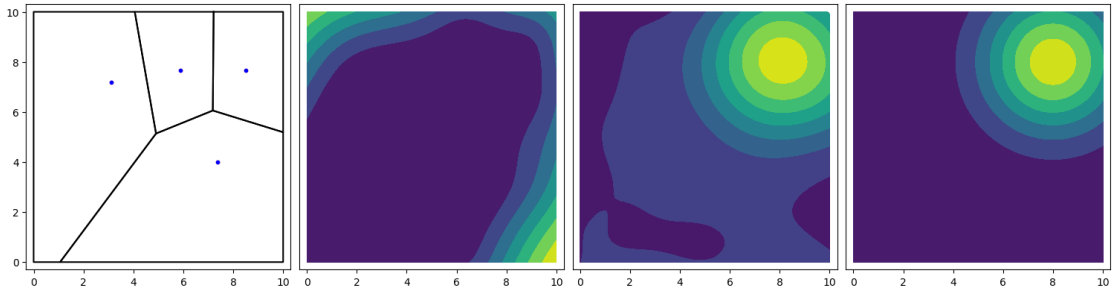
to zero everywhere given the time-variant control implementation. Hence, the covariance weight is always relevant in the exploration-exploitation trade off (eq. (2.43)) influencing the robots configuration accordingly. After 40 seconds the spatial field changes quickly its shape. In a few seconds all the robots realize the wrong estimation deleting from the training samples the old data that generate a high uncertainty. Consequently, the group

of robots is pushed towards a new exploration phase where they maximize uniformly the coverage of the environment (Fig. 2.31c). Finally, after approximately 100 seconds, the robots managed to estimate again the spatial field, converging in to a configuration that maximizes the coverage according to the exploration-exploitation trade off (eq. (2.43)). It is worth noting that the group of robots converged to similar configurations at times $t = 32sec$ and $t = 108sec$ (Fig. 2.31) since the two scenarios are similar.

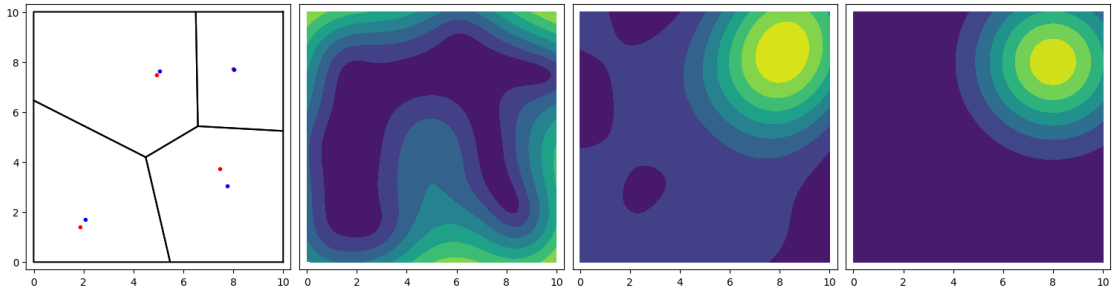
Finally, we did another experiment where we tested the eventuality of an unexpected event that can potentially change the constant spatial process in the environment (see Figure 2.32). An example of this scenario is a sudden wind during the monitoring of the pollution in the air or a sudden ocean current during the monitoring of the salinity in the water. A scenario in which the team of robots does not deal with a gradual but a sudden change of the spatial process is modelled exploiting a step-like time decay function (see equation (2.53)). Differently from the experiment shown in Figure 2.31 we consider the situation in which the spatial process does not change over time, but the team of robots still need to react to the potential change of the process and correct the balance between the exploration and exploitation to not keep a wrong estimation of the spatial field. As previously, this consists in a Gaussian-like function with mean in $(8, 8)$ and standard deviation as 2. The scenario considered in this experiment is the continuation of the scenario shown in experiment described in Figure 2.30, in which the team of robots, after converging to a optimal exploitation-exploration configuration, is perturbed by a sudden event that adds a high uncertainty on the process estimation. As previously, the hyperparameters are chosen as $C_1 = 3.2$ and $C_2 = 2$. The step time decay function parameters are chosen as $\tau_1 = 3$ and $\tau_2 = 50sec$: this means that at time $t = 50sec$ rapidly increase the uncertainty over the process estimate and the old sampled data rapidly become negligible for a new estimation. The Figure 2.32 shows the whole experiment as the continuation of the experiment described in Figure 2.30, where at the very beginning the robots have estimated the spatial field and the uncertainty is low in the most of the environment. Hence, the robots are more concentrated towards the spatial process location according to the coverage based control. After 50 seconds, the balance between exploration and exploitation is disrupted by the increasing in the uncertainty. The team of robots start an exploration phase and immediately gets new estimate of the spatial process, as it has not changed over time. One can see how the control strategy efficiently balances the exploration-exploitation trade-off by using some robots to explore the environments by sampling new data and some robots to monitor the estimated spatial process. After approximately 60 seconds, the multi-robot system has again a reliable estimate of the spatial field. According to the coverage based control strategy the majority of robots move towards the most interested area defined by the spatial field and one robot continues to explore the environment to progressively reduce the uncertainty.

Webots Simulations

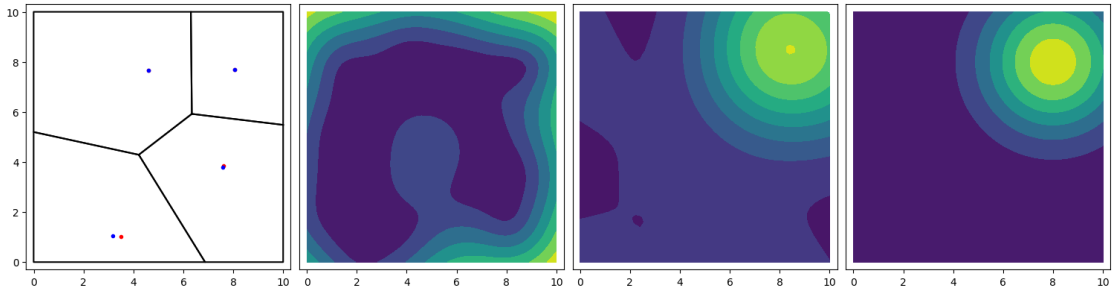
In this section we report some of the tests and simulation we did on Webots simulator. This is a mobile robotics simulation software that provides realistic simulation environ-



(a) $t = 30s$.



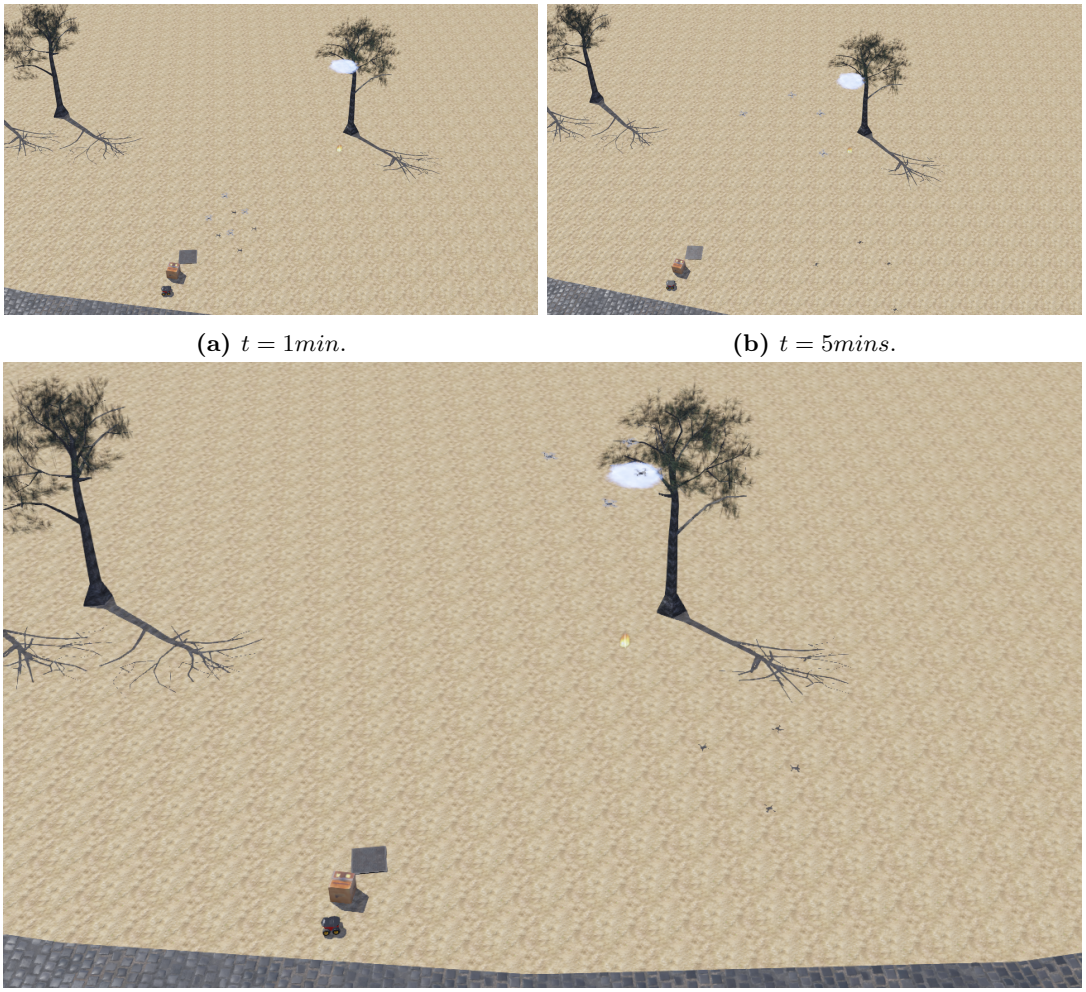
(b) $t = 55s$.



(c) $t = 65s$.

Figure 2.32: The figure shows three different subfigures, one for each time step, of a simulation where the team of robots is tasked to estimate a spatial process and optimally cover it. During the experiment we assume we have an unexpected event that is modelled as a step-like time decay function that can potentially change the spatial process. Each subfigure shows four aspects of the simulation: the first image from the left shows the team of robots (blue dots), the Voronoi partitioning (black lines) and the computed centroid for every Voronoi cell (red dots). The second image from the left shows the uncertainty over the domain: yellow areas mean a high standard deviation and a high uncertainty, blue areas mean a low standard deviation and low uncertainty. The third image from the left shows the robots estimation up that instant of time. The fourth image from the left shows the unknown spatial process that the team of robots is tasked to estimate.

ments, robots and devices [131]. Moreover, the provided robot libraries allow the user to transfer the control programs to several commercially available real robots. As shown in Fig. 2.33 we considered a simple scenario in which the team of drones has to detect and monitor the presence of a fire in a open environments. We assume the robots are able to detect the smoke particles concentration in the air and we assume the smoke has a Gaussian-like distribution in the environment. The experiment shown in Fig. 2.33 is similar to the one shown in Fig. 2.30, where the team of drones has to balance the exploration and exploitation to model the smoke distribution and optimally cover it to optimally monitor the fire distribution. The drones take off from an initial configuration. They start an exploration of the environment in order to learn the smoke concentration in the air and to model its distribution. After 10 minutes the drones uses the estimated



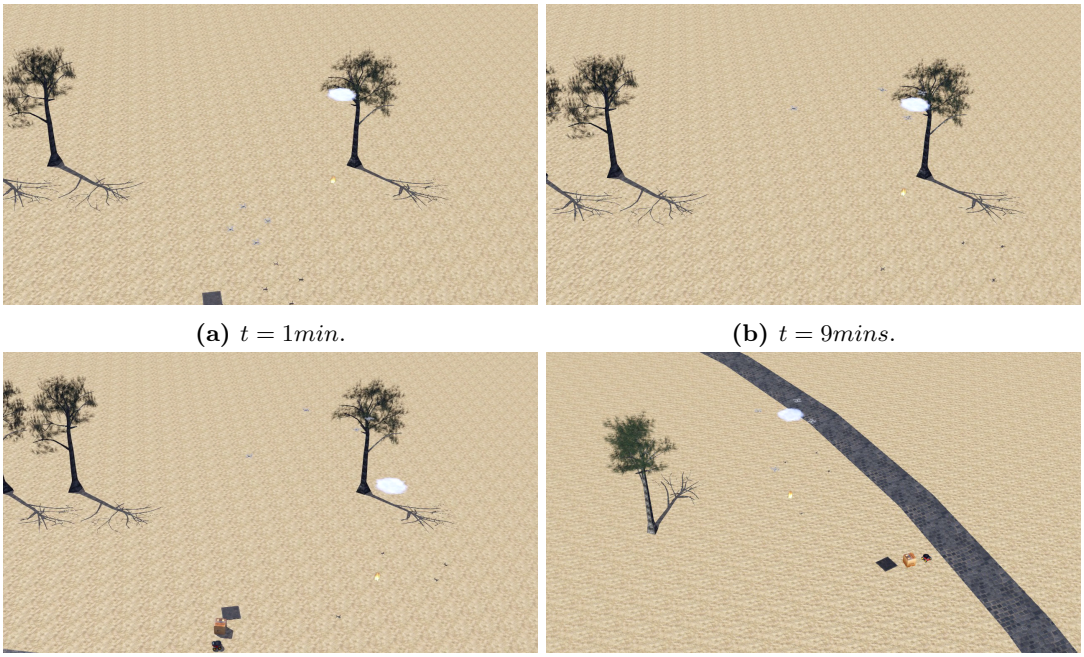
(a) $t = 1min.$ **(b)** $t = 5mins.$

(c) $t = 10mins.$

Figure 2.33: The figure shows three different stages of a simulation where the team of drones is tasked to estimate a smoke distribution in a open environment in order to detect and monitor the respective fire. We assume each drone is able to detect the smoke particles concentration in the air and we assume the smoke has a Gaussian-like distribution in the environment. The team of drones initially explore the environment in order to learn the smoke distribution and finally uses this estimation to optimally cover the environment, monitoring more the area where the fire is concentrated.

distribution to optimally cover the environment. Hence, the team of drones concentrate more over the fire, the origin of the smoke.

We also simulated the scenario in which the team of robots has to deal with a spatial process that is changing over time. The experiment shown in Fig. 2.33 is similar to the one shown in Fig. 2.31, where the drones has to balance the exploration and exploitation to model the smoke distribution that changes over time and optimally cover it to optimally monitor the corresponding fire. As described previously, the drones take off from an initial configuration and after an initial exploration phase they estimate the smoke distribution in 9 minutes. It is worth noting that in this scenario, the overall uncertainty is greater than in the previous experiment, owing to the fact that the robots are dealing with a time-varying spatial field. Therefore, the weight of exploration is augmented in the exploration-exploitation trade-off. After 11 minutes, there is a change in the fire location and in the corresponding smoke distribution. The drones control strategy shifts to the



(a) $t = 1min.$ **(b)** $t = 9mins.$

(c) $t = 11mins.$ **(d)** $t = 20mins.$

Figure 2.34: The figure shows four different stages of a simulation where the team of drones is tasked to estimate a time-varying smoke distribution in a open environment in order to detect and monitor the respective fire. We assume each drone is able to detect the smoke particles concentration in the air and we assume the smoke has a Gaussian-like distribution in the environment. The team of drones initially explore the environment in order to learn the smoke distribution and to use this estimation to optimally cover the environment. After an initial period of time, a change in the location of the fire and in the corresponding smoke distribution occurred. The team of drones was able to adapt to the changes and to update the estimation of the process distribution. Finally, balancing the exploration-exploitation trade-off, the drones shifted again their focus towards the origin of the smoke.

exploration phase in order to estimate the new distribution of the smoke. Finally, after approximately 20 minutes, the team of drones manages to successfully re-model the process, converging into a configuration that maximizes the coverage of the new smoke distribution.

2.6.9 Performance Evaluation

We evaluated the performance and results of the proposed methodology comparing it to other control strategies: a *random exploration* strategy, a coverage-based control strategy where the robots do not have any information about the spatial field (we refer to this as *plain coverage*), and a coverage-based control strategy where the robots possess a complete and global knowledge of the environment and the spatial process (we refer to this as *ideal coverage*). We compared the performance of these strategies to the proposed one in terms of their proximity to optimal coverage of the spatial process, which is defined by the minimization of the optimization function defined in (2.35). In the random exploration algorithm, the robots explore the environment randomly without any control strategy guiding their actions (green line in Fig. 2.35 and 2.36). Conversely, the plain coverage-based control strategy can only direct the team of robots towards a uniform coverage of the environment when they lack any information regarding the spatial field

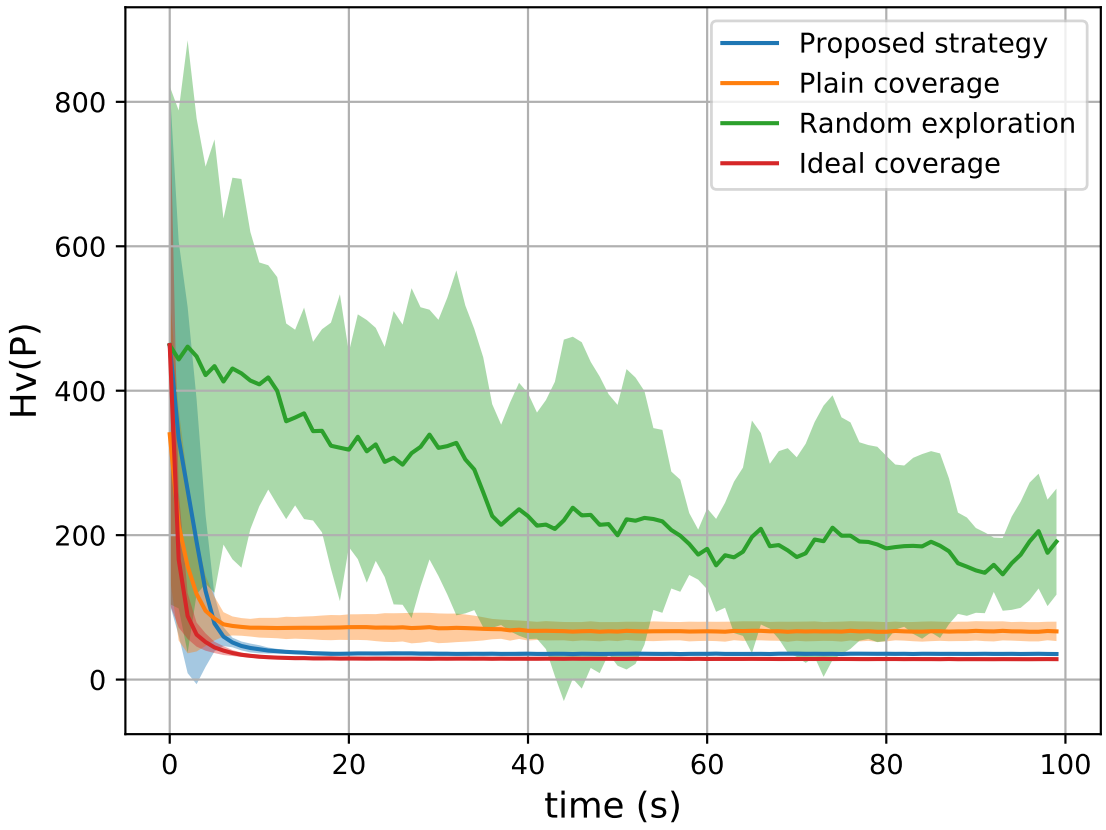


Figure 2.35: The figure shows the performance comparison between the proposed coverage-based control strategy with three other approaches in a scenario where the robots are tasked to monitor a spatial process. The performance is evaluated by the optimization function of the coverage control that evaluate the robots location with respect to the spatial process location (eq. 2.35). The graphs reports the mean ad the standard deviation with 95% confidence interval. The green line represents the random exploration algorithm, the orange line represents the plain coverage-based control strategy where the robots lack any information on the spatial field. The red line represents the ideal coverage-based control in a scenario where the robots have a complete and accurate knowledge of the environment and the spatial process. The blue line represents the proposed coverage-based control strategy.

(orange line in Figures). With the ideal coverage, the robots possess a complete and accurate understanding of the environment and the spatial process, thus allowing them to optimize the coverage of the spatial field through the coverage control strategy (red line in Figures). The proposed control strategy aims to achieve similar performance as the ideal coverage while lacking a priori global knowledge of the environment and the spatial field (blue line in Figures).

In Fig. 2.35 we evaluate the performance of the control strategies when dealing with a stationary spatial process, as in the case shown in Fig. 2.30. It is worth noting that the proposed control strategy performs considerably better than the plain coverage and random exploration, while it exhibits a slightly lower performance compared to the ideal approach (i.e., coverage with complete and global knowledge of the spatial field). This is due to the fact that the robots must concurrently carry out both exploration and estimation of the unknown spatial field, which impacts the coverage control as it is influenced by the uncertainty of the spatial field in the environment.

In Figure 2.36 we evaluate the performance of the control strategies when dealing with

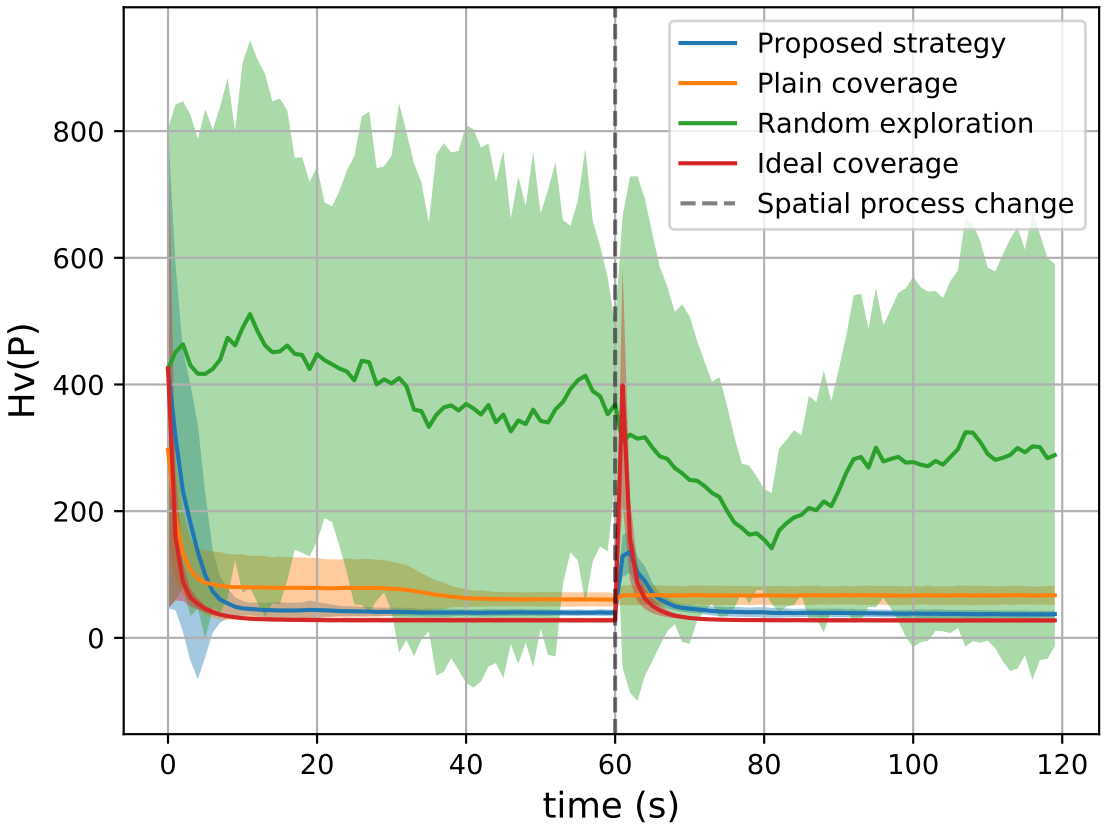


Figure 2.36: The figure shows the performance comparison between the proposed coverage-based control strategy with three other approaches in a scenario where the robots are tasked to monitor a time-varying spatial process. The performance is evaluated by the optimization function of the coverage control that evaluate the robots location with respect to the spatial process location (eq. 2.35). The graphs reports the mean ad the standard deviation with 95% confidence interval. The green line represents the random exploration algorithm, the orange line represents the plain coverage-based control strategy where the robots lack any information on the spatial field. The red line represents the ideal coverage-based control in a scenario where the robots have a complete and accurate knowledge of the environment and the spatial process. The blue line represents the proposed coverage-based control strategy. After 60s there is a sudden change of the spatial process. Both the ideal coverage control method and the proposed method exhibited a responsive behavior to this change.

a time-varying spatial process, as in the case shown in Fig. 2.31. At 60s, there is a sudden change in the configuration of the spatial process, causing it to relocate within the environment. With the ideal coverage, the robots are focused on the location of the spatial process. Upon the sudden change of the spatial process, a significant increase in the optimization function occurs, causing the robots to move towards the new location of the spatial process. The proposed control strategy demonstrates its ability to respond to the unexpected change in the configuration of the spatial process. Despite lacking any prior knowledge of the time-varying nature of the process, the robots are able to effectively learn and estimate again the spatial field through exploration. As a result, they are able to converge towards an optimal coverage of the new spatial field within a relatively short period of time, as evidenced at 60s. As in the previous case, the performance of the proposed control strategy is slightly inferior to the ideal coverage approach. This is because the robots must continuously explore a spatial field that changes over time, resulting in an increase in the uncertainty of the estimate due to the time decay component in the control strategy, which negatively impacts the coverage

control performance. However, it is significantly better than the plain coverage or the random exploration.

2.6.10 Discussions

In this study we presented a coverage based control algorithm to coordinate a group of robots with the aim to optimally learn and estimate a spatial field and optimally cover it. The proposed control strategy efficiently manage the trade-off between the exploration of the environment for the process estimation and the exploitation of this estimation to optimally cover the environment. Moreover, we presented a novel methodology to deal with time varying spatial processes. The multi-robot team adapts its configuration and the exploration-exploitation trade-off optimization depending on the spatial field changes over time. The methodology described in this section exploits the Gaussian Processes to learn and generate a model of the spatial field from the data sampled by the team of robot in the environment. We, finally, presented a novel approach to limit the training set to avoid the computational issues introduced by the Gaussian Processes when dealing with large training sets. We tested the performance of the described methodology on several simulations.

Chapter 3

Traffic Management

As previously discussed, in the near future, large group of robots, such as fleets of vehicles or swarms of drones will be used in real life and industrial applications to perform the challenging tasks that are too complex or dangerous for a single robot to handle [42]. In the previous chapter we focused on control algorithms and strategies suitable to solve real life problems such as monitoring, search and rescue or surveillance. In this chapter we focus more on the role of multiple robots in industrial environments. In particular, nowadays, industrial robots are becoming fundamental for high volume operations, to maximize productivity and throughput and to reduce costs. The use of multi-robot systems in industrial applications has the potential to significantly improve productivity and efficiency, as well as to reduce the need for human labor in some tasks. In particular, one of the most relevant applications of a multi-robot system is to increase the automation in warehouses and factories. In fact, fleets of autonomous vehicles are exploited in industrial scenarios to improve the efficiency and productivity, and to reduce the need of human labor. Automated warehouses and automated factories are spreading as the solution to the increasing demand and the Automated Guided Vehicle (AGV) systems are consequently gaining popularity and relevance. AGVs increase efficiency, flexibility, and reduce costs by helping to automate a manufacturing facility or warehouse [132]. As such, several strategies have been developed to deal with major issues in automated warehouses, such as traffic management or safety and performance guarantees [133]. Along these lines, motion coordination of a high number of vehicles has become a widely studied research topic in the field of multi-robot systems [134].

The study presented in this chapter concerns the coordination and the traffic management of a group of Automated Guided Vehicles (AGVs) moving in a real industrial scenario, such as an automated factory or warehouse. In particular, we focused on studying a methodology to bridge the gap between the multi-robot control strategies and the industrial world. In fact, the current state-of-the-art does not consider the typical issues and non-idealities introduced by a real world application. In most of modern automated factories, the AGVs are guided through manually defined routes with ad-hoc strategies for coordination. This increases installation costs and reduces the flexibility when a change in the environment is required. The challenge we address in this work is to define a flexible methodology able to automatize these industrial applications.

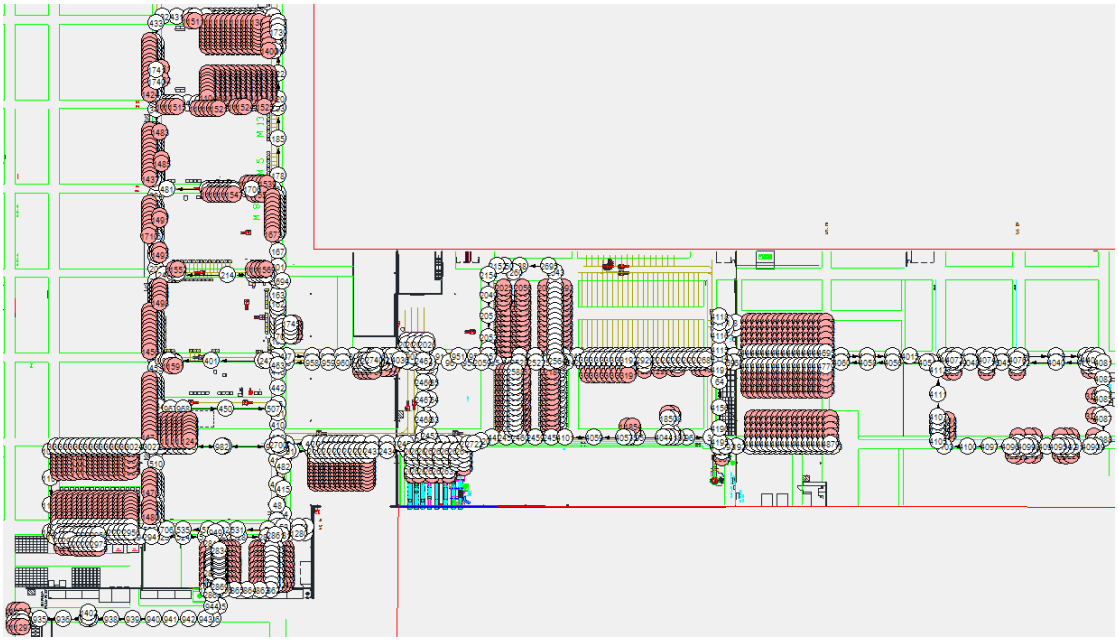


Figure 3.1: The figure shows the factory structure (red, green, and black drawings) and the plant graph $\mathcal{G}$ of our use case, which is elicited by a real application. In particular, red nodes indicate load or unload locations, while red nodes indicate transit areas.

The proposed methodology is based on a three-layer control architecture, which is described as follows: 1) the Top Layer (or Topological Layer) allows to model the traffic of vehicles among the different areas of the environment; 2) the Middle Layer allows the path planner to compute a traffic sensitive path for each vehicle; 3) the Bottom Layer (or Roadmap Layer) defines the final routes to be followed by each vehicle and coordinates the AGVs over time. In this chapter we describe our study about a novel flexible and robust control strategy for multi-AGVs traffic-aware management that aims at grounding the theoretical methods to an industrial application by facing the typical practical issues such as vehicles error, dynamical obstacles, etc. We finally present the studied coordination algorithm that does not rely on ad-hoc assumptions or rules and has the aim to avoid congestions, collisions and deadlock dealing with delays or vehicles motion errors.

3.1 Literature Review

Different strategies have been deepened to increase global performance indices such as efficiency, safety, scalability, or robustness to failures [135]. Two approaches can be mainly found in literature: centralized and decentralized approaches. The former strategy is able to find the optimal solution for the multi-robot system planning and control problem [136]. Thus, centralized approaches show generally better performances when compared to decentralized ones [137]. However, the problem complexity — and hence computational costs — quickly becomes impractical while increasing the number of robots over a few tens [138]. Moreover, centralized control methods are usually classified as coupled or decoupled. Coupled methods [132] control the whole system exploiting classical single-vehicle motion planning control methods. The task is to find a path for each agent

in the group, from a start to a goal position, which avoids collisions between agents. Two robots are in collision if they are located in the same position at the same time. This Multi-Agent Path Finding problem (MAPF) [139] can be solved for example exploiting the well-known Conflict-Based Search (CBS) optimal algorithm [140], a typical coupled approach. These approaches generally find optimal solutions, at the cost of a high computational effort and limited scalability. Decoupled methods [141, 142, 143] simplify the vehicles coordination and control problem by dividing it in two main phases: path planning and motion coordination. Decoupled control algorithms are generally much faster than coupled ones, however, they may find sub-optimal solutions and they are prone to generating deadlocks.

In decentralized approaches [142, 144, 145], each robot communicates with its neighboring ones to estimate the global performance of the system and to make local control decisions [146]. Decentralized methodologies are computationally less demanding that centralized ones. The computation for the multi-robot system coordination is shared among the agents, making the system easily scalable to large scenarios [147]. However, these methodologies present some disadvantages: they generally find a sub-optimal solution to the coordination problem and, especially, they may fail in finding a solution and a feasible path for every agent in the system due to deadlocks [148, 149]. Moreover, the complexity may still be high in terms of exchanged messages or message size.

For these reasons, although relevant studies have been proposed to manage the coordination of automated vehicles exploiting decentralized approaches, the traffic and coordination of AGVs in industrial scenarios is generally managed by a centralized supervisor. In other words, there exists a central computational unit that manages all the information coming from the environment and the vehicles to optimally coordinate the vehicles. Moreover, decoupled approaches are preferable to coupled ones, since these generally require a significant amount of computational effort, are generally more vulnerable to failures and are hardly scalable to large teams of robots [150]. The management of a fleet of AGVs includes dealing with a multi-robot path planning problem: in decoupled control strategies this is generally decomposed into modules to reduce the complexity [151]. A hierarchical architecture control strategy can be exploited to solve planning and coordination problems for large-scale infrastructures [148]. Every robot is coordinated along its planned path to its goal in order to avoid conflicts and dissolve deadlocks. Typically in modern automatic warehouses and factories the movements and trajectories of the automatic vehicles are defined by the roadmap, which consists in a set of predefined virtual paths, as illustrated in [152]. The roadmap depends on the plant structure and has a relevant role in the traffic management of the AGVs affecting the performance of the control system. Studies has been conducted to realize flexible navigation systems for AGV operation in partially structured warehouses or in automatic factories with frequent changes in the floor plant layout [153]. Thus, typically, in industrial applications, once a path has been computed, it is assigned to every AGV and their coordination is managed over the plant following a set of traffic rules manually defined during the installation of the system [154]. This approach requires a lot of personnel working during the AGV system deployment in the plant and when variations are required in the system, since all the exceptions have to be manually managed. The work illustrated in [155, 156] describes

the coordination diagrams, which are tools used for representing the possible collisions among the vehicles, useful to handle the coordination of the AGVs limiting the use of manual traffic rules. Several methodologies have been proposed for the definition of a roadmap, typically based on random sampling techniques [157, 158] or using probabilistic methods [159]. Another strategy is proposed by [160, 161] which is more focused on the roadmap definition for industrial scenarios. These works aim to find a solution for the automatic generation of the roadmap, building the framework for traffic management of the AGVs upon such solution. Thus, the roadmap is defined to help the coordination of vehicles in the warehouse or factory. However, this approach excludes the majority of the industrial scenarios where the roadmap is provided a priori and represents then a constraint for the traffic manager: e.g. the coordination strategy proposed in [160] can not be applied in these industrial scenarios. Moreover, the mentioned methods are designed for managing robots in static, dedicated environments and cease to work if key assumptions on the infrastructure or on the fleet are dropped, as typically happens in industrial applications [162].

In this paper we present a centralized and decoupled control strategy to efficiently coordinate the movements of a fleet of automated vehicles in a real industrial scenario. The proposed approach allows us to build a control system scalable to large fleets of vehicles and able to efficiently deal with the issues and performance requirements typical from an industrial application. This study described in [2] is built upon the work presented in [6], where preliminary results were introduced on a hierarchical approach and methodology to design a flexible traffic manager able to coordinate a fleet of real AGVs in an industrial scenario. The control strategy exploits a multiple layer approach, which allows to simplify the path planning computation, model the traffic and coordinate the vehicles on the planned routes. The proposed control architecture is built upon a predefined and fixed roadmap of the plant. In fact the structure and organization of an automated warehouse or factory is usually designed to optimize and simplify the coordination of multiple AGVs in the plant. The presented work considers the common case where the AGV system has to be deployed in an existing and structured plant, where the roadmap of the plant is given and cannot be modified. The proposed methodology aims to introduce a flexible and efficient system able to deal with the non idealities and issues introduced typically by a real and industrial implementation of the roadmap, such as the presence of locations where AGVs are required to perform loading/unloading operations (red nodes in Fig. 3.1 and Fig. 3.3), lack of redundancy or bidirectional single vehicle corridors. The proposed strategy aims to be robust to the issues that usually affect industrial scenarios, such as dynamical obstacles and communication errors [163]. In this chapter we present and describe the overall hierarchical control architecture, including the full coordination strategy and the vehicles path planning algorithms. Moreover, we present a detailed detailed description of the vehicles path planning algorithms and the coordination method implemented on the lowest layer of the hierarchical architecture. The aim is to avoid the collisions among the AGVs that are moving on the precomputed routes. The coordination algorithm, described in Section 3.6, exploits the prediction of future locations of the AGVs to solve the future conflicts. Furthermore, in this study we present a novel deadlock prevention algorithm, which exploits the time expanded graphs [164] to predict

and avoid deadlock scenarios. Deadlock situations arise when a group of tasks becomes interlocked in such a way that they cannot be completed. The authors in [165] theorize the general conditions sufficient to generate a deadlock situation. The most relevant one for a multi AGV system defines that a deadlock state is generated by a circular chain of tasks, such that each task holds the resources needed by the next task in the chain. Different approaches were studied to prevent deadlock occurrence in a multitasking system for various scenarios [165, 166]. In a multi vehicle system scenario, the authors expressed the traffic deadlock in terms of graphs and show how a circuit (directed loop) in the generated graph is a necessary and sufficient condition for a deadlock. In this work we propose a methodology based on time expanded graphs to deal with traffic deadlocks for every time step in the trajectory followed by the AGVs. The proposed methodology has been integrated into the control software of real industrial plants, in collaboration with Proxaut s.r.l.: this allowed us to validate the proposed methodology on a real use case. In this work we describe the experiments we conducted in one automated factory and we compare the performance of the developed software with the company's one usually implemented on its plants.

The rest of the chapter is organized as follows. Section 3.2 aims at providing the reader with the main background notions that will be used in this document. Section 3.3 gives an overview and a brief description of the control architecture proposed in this study. Section 3.4 describes in detail the proposed multi-layer architecture, the role of the layers in the coordination and planning and their interconnections. Section 3.5 focuses on the proposed path planning strategies adopted to consider the traffic status in the path computation. In Section 3.6 we describe the whole traffic manager algorithm, the AGVs coordination methodology and the proposed deadlock prevention policy. Finally, Section 3.7 reports the simulations and the experiments conducted on the real industrial plant and Section 3.8 deals with the conclusions of this work.

3.2 Algebraic Graph Theory

In this section, we recall some notions on graph theory that will be used in the chapter to model interconnections in our multi-layered architecture. The reader is referred to [167] for additional details.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a directed graph characterized by a set $\mathcal{N}(\mathcal{G})$ of vertices or nodes and a set $\mathcal{E}(\mathcal{G}) \subseteq \mathcal{N}(\mathcal{G}) \times \mathcal{N}(\mathcal{G})$ of edges. Given an edge $(i, j) \in \mathcal{E}$, then the node j is a neighbor of i. Let $\mathcal{K}$ be a subgraph of $\mathcal{G}$, then $\mathcal{K}$ is a graph where the node set $\mathcal{N}(\mathcal{K})$ is a subset of $\mathcal{N}(\mathcal{G})$ and the edge set $\mathcal{E}(\mathcal{K})$ is a subset of $\mathcal{E}(\mathcal{G})$. A path is a finite sequence of edges which joins a sequence of nodes, which are all distinct. In a path of length L, the nodes can be listed as $\{n_1, n_2, \ldots, n_L\}$, such that the edges are (n_i, n_{i+1}), where $i = 1, 2, \ldots, L-1$.

Let $\mathcal{G}$ be a directed and *connected* graph, then at least one path exists between each pair of nodes in $\mathcal{N}(\mathcal{G})$. Moreover, $\mathcal{G}$ is a *strongly connected* graph if, for any pair of nodes u, v in $\mathcal{N}(\mathcal{G})$, a directed path can be defined from u to v and from v to u. Contrariwise, $\mathcal{G}$

is a *weakly connected* graph if, for any pair of nodes u, v in $\mathcal{N}(\mathcal{G})$, a directed path exists from u to v, but not necessarily from v to u. Another feature of a graph graph is the *Redundancy*, which represents the existence of multiple directed paths that connect any pair of nodes. High values of redundancy implies multiple directed paths that connects the same pair of nodes. Given a directed graph $\mathcal{G}$, we assume each edge in $\mathcal{E}(\mathcal{G})$ is associated with a positive weight. Thus, given two nodes $u, v \in \mathcal{N}(\mathcal{G})$, the minimum cost path from u to v can be computed by using standard graph search algorithms (e.g., Dijkstra, A*, DFS — see [168, 169] for details). In this document, we exploit the graph to represent the predefined feasible routes to be followed by vehicles in the plant. In this sense, the graph nodes $\mathcal{N}(\mathcal{G})$ represent the reachable locations in the plant and the edges $\mathcal{E}(\mathcal{G})$ model the available roads between two nodes. Accordingly, the edge weight can be arbitrarily defined to quantify relevant information: in this document, we will use the weights to quantify the expected travel time. Standard graph search algorithms are used to compute the path from a start to a goal location in the plant minimizing the travel time.

Moreover, let us introduce the time-expanded approach [170] which constructs the time-expanded digraph in which every node corresponds to a specific time event and edges between nodes represent the connections between the two events. Every node of the time-expanded graph consists in a static graph, hence a time-expanded graph can be considered as a sequence of static graphs (see Fig. 3.2). Let $\mathcal{G}_t(t)$ be a directed time-expanded graph. We have that $\mathcal{G}_t(t) = \{\mathcal{G}(1), \mathcal{G}(2), \ldots, \mathcal{G}(i)\}$, where $\mathcal{G}(i)$ is a snapshot of $\mathcal{G}_t$ at time $t = i$ and is denoted by $\mathcal{G}(i) = (\mathcal{E}(i), \mathcal{N}(i))$. In the proposed control strategy, we exploit the time expanded graph to represent the *precedences* and the result of the negotiation between vehicles following the execution of the coordination algorithm. Each node of the graph represents an automated vehicle in a particular instant of time and the directed edges between the nodes represent the result of the negotiation determined by the coordination diagram (see Sec. 3.6.1). The utility of the time expanded graph is to represent the precedences and the negotiation over time. Hence, for example, in Figure 3.2, at the instant of time $t = 4$ the AGV represented by node A has to wait for the AGV C, and similarly, the AGV D has to wait for the AGV B.

3.3 System Overview

The control system we propose is composed of *(i)* a multi-layer architecture to model the environment, and *(ii)* a traffic-aware coordination and path planning strategy. The main objective is to effectively coordinate a high number of AGVs in an industrial environment, avoiding traffic congestion. The control approach is centralized to meet the performances and the efficiency required by an industrial application, hence the trajectories are optimally computed for every vehicle with a global knowledge over the industrial plant. Furthermore, the control architecture is decoupled in the path planning and in the coordination strategy to reduce the computational cost and make the system easily scalable. The multi-layer architecture, pictorially represented in Fig. 3.3, will be detailed in Section 3.4.

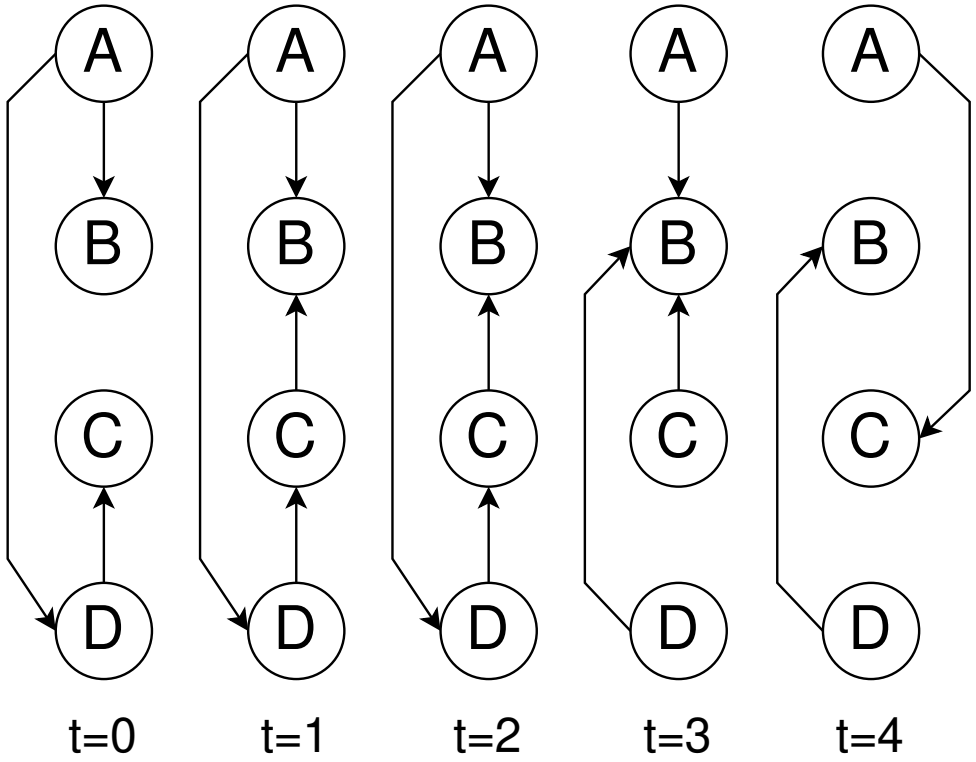


Figure 3.2: An example of directed time expanded graph used in the proposed coordination strategy. The graph considered in figure is composed by four nodes: A, B, C, D. The graph is considered over time and the edge set $\mathcal{E}(\mathcal{G}(i))$ at time $t = i$ can be different from the edge set $\mathcal{E}(\mathcal{G}(i+1))$ at time $t = i + 1$.

To model the traffic evolution quantitatively, we divide the environment into three layers: *(i)* the Top Layer (or Topological Layer), responsible for traffic management; *(ii)* the Middle Layer, responsible for the path planning of the AGVs; *(iii)* the Bottom Layer (or Roadmap Layer), containing all possible vehicle routes and handling the fleet coordination. The roadmap models all the possible trajectories the vehicles can follow and all the locations the vehicles can occupy in the plant. Hence, the roadmap model consists in a graph where the nodes represent the locations sampled in the plant. The accurate tracking of the AGV over the roadmap is guaranteed by the high number of points (the graph nodes) that sample the feasible trajectories. A trajectory is, indeed, modelled as a sequence of connected nodes.

Hereafter, the term "path" refers to a generic sequence of nodes computed by the planner over a graph, while "route" refers to the specific sequence of nodes computed over the roadmap, which is necessary to determine the trajectories to be followed by the vehicles. Upon the computation of the route and trajectory for each vehicle, it is possible to determine the location of the vehicles over time, since they move with a known velocity profile. As a result, it is possible to predict the future location of the vehicles and take action to prevent congestion and deadlocks.

The path planning and coordination strategy, detailed in Section 3.5, exploits information from the three layers, to define an optimal and feasible route for each AGV. In detail, the Top Layer is based on a discretization of the environment in topological areas (intersections, load/unload areas, congestion risk areas, etc.) and provides quantitative

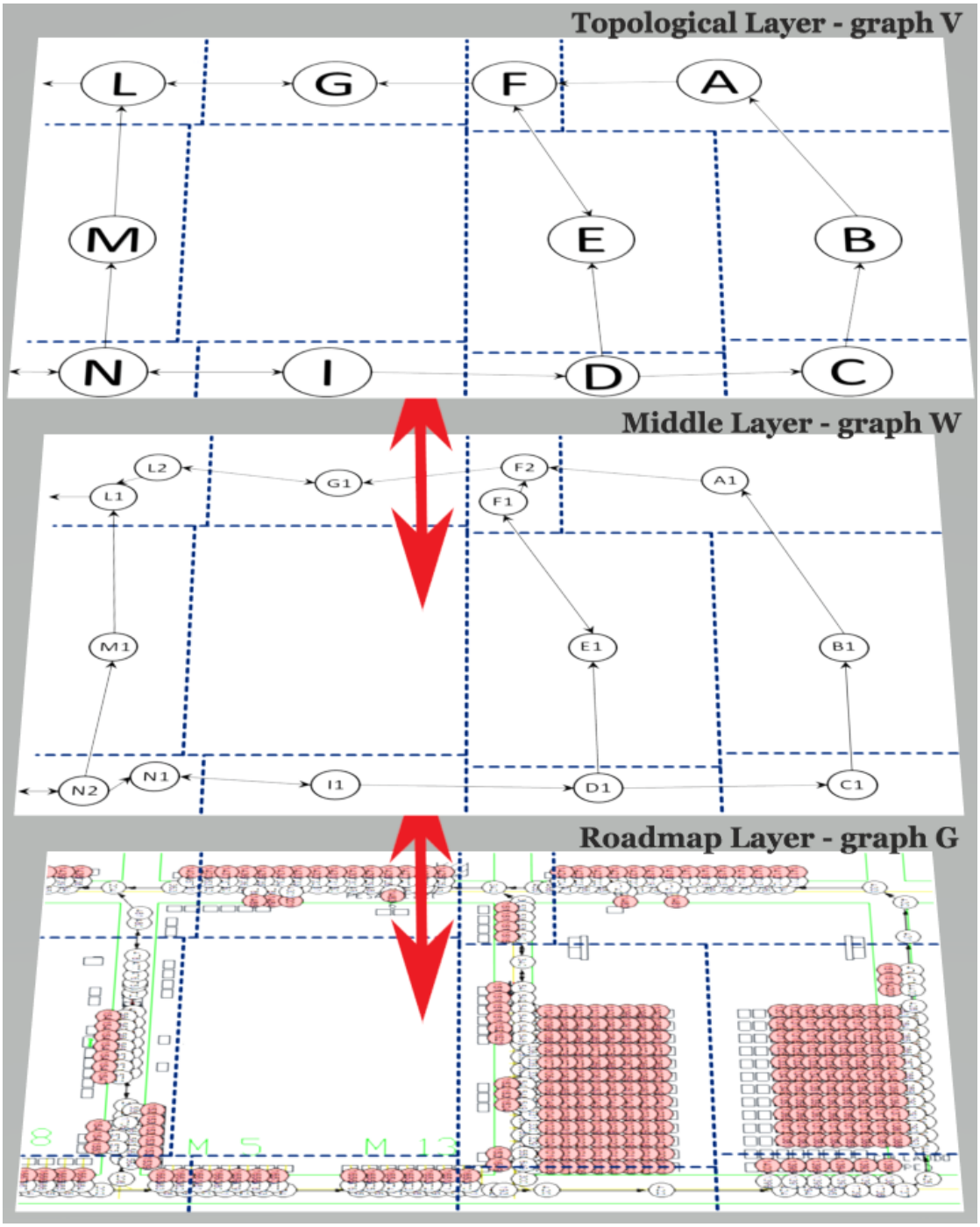


Figure 3.3: Overview of the proposed multi-layer architecture. The figure show a portion of the three graphs, one for each layer, of the studied use case, from the bottom: the roadmap layer, the middle layer, the topological layer. On the roadmap layer the red nodes indicate the load or unload locations in the factory, while the white nodes indicate the transit locations. The blue dotted lines show the different plant areas, divided by topology, each one assigned to a sector S.

information about the current AGVs traffic in each of them. Each AGV's path planning consists in two main steps. A path is computed over the Middle Layer, which models the actual connections among the areas of the Topological Layer and knows the traffic information of these areas. Subsequently, the computed path is exploited to plan the actual feasible route of the AGVs over the Bottom Layer, where the trajectories are built over time and conflicts among the AGVs are avoided by means of coordination.

Given a route, the trajectory following problem is the problem of computing a kynody-

namically feasible temporal profile of the vehicle control inputs to move the robot along the given route. This problem is beyond the scope of this study and will not be addressed. Thus, we assume each AGV is equipped with an on-board control algorithm capable of ensuring safety (i.e. [171, 172]) and accurate trajectory tracking (i.e. [173, 174]).

Once the route is computed from the path planner in the Bottom Layer, the AGVs have to be coordinated on the respective paths to avoid collisions and deadlocks. A collision between two AGVs occurs when they occupy the same location at the same time. The purpose of the coordination is to modify the time when the vehicle reaches the conflicting locations. In this work we propose a coordination strategy based on a future prediction of the AGVs trajectories able to solve future conflicting situation and avoid congestions or deadlocks. Deadlock situations arise when a group of vehicles becomes interlocked in such a way that they cannot complete their tasks. The coordination between AGVs is represented exploiting *precedence* graphs, where the nodes are the AGVs and the edges represent the precedences between them. This information is propagated over time through the time expanded graph where every layer coincides with a precedence graph at a definite time step. Avoiding cyclic loop inside the graph we prevent deadlocks [165].

We assume that the coordination strategy runs on a central elaboration unit, communicating with the AGVs. In particular, the central control process periodically sends a command to the AGV, which contains the future steps of the route the vehicle has to follow and the time the vehicle has to wait in a node (location) that the traffic manager has set to manage congestions and deadlocks. Moreover, we assume each on-board low-level control can deal with communication non-idealities, such as packet losses or transmission delays. Hence, communication issues are assumed non-critical for the low-level control. The central control process periodically sends a command to the AGV, which contains the future steps of the route the vehicle has to follow and the time the vehicle has to wait in a node (location) to manage the traffic. It may occur that the AGV is not able to execute a given command because of communication issues, robot hardware issues or safety issues. Since all these issues bring to a stop/slowdown of the vehicle, they can be equally treated by the central control algorithm as unexpected execution errors on the interested vehicle. The proposed control strategy has been tested to be robust to the delays introduced on the AGV motion by these errors.

3.4 Multi-Layer Architecture

We here detail the proposed multi-layer architecture by describing all layers, their interconnections and how they are used for coordination and traffic management.

3.4.1 Roadmap Layer (Bottom Layer)

The roadmap of the plant represents all the possible routes the AGV can follow and the locations the AGV can cross in the environment. The Bottom Layer provides a modelling of the roadmap through a graph.

This layer is responsible for the planning of the route to be followed by the AGV. The *plant graph* $\mathcal{G}$ is then introduced to abstract the description of the roadmap: the set of nodes $\mathcal{N}(\mathcal{G})$ represents the locations in the plant the vehicles can reach, and the set of edges $\mathcal{E}(\mathcal{G})$ represents the feasible trajectories connecting those locations.

Thus, let p_i, p_j be two locations in the plant, associated with nodes $i, j \in \mathcal{N}(\mathcal{G})$. Then, the edge (i, j) exists in $\mathcal{E}(\mathcal{G})$ if there exists a kinematically feasible route from p_i to p_j. Each edge in $\mathcal{E}(\mathcal{G})$ is weighted by the distance between the two corresponding nodes and it also contains the average time required by the vehicle to travel this distance. Therefore, the planned route traveled by the vehicle and modeled as a sequence of nodes in $\mathcal{N}(\mathcal{G})$ is able to describe the vehicle's motion over time. Therefore, the planned route traveled by the vehicles, which is represented as a sequence of nodes in $\mathcal{N}(\mathcal{G})$, serves as an effective means of describing the vehicle's motion over time. The traffic manager planner computes the route over the graph $\mathcal{G}$ that not only minimizes the time required by the vehicle to reach its goal, but also minimizes the traffic congestion in certain areas of the plant. This traffic sensitive path planning is performed thanks to the topological layer.

In this work we consider a common industrial scenario in which the AGV system has to be deployed in an already existing plant. The roadmap, hence, is fixed and can not be changed to optimize the traffic management of vehicles as in [160]. The plant graph, built upon the roadmap, is consequently treated as a constraint for the modelling, planning and the design of the control architecture. A representative example of a roadmap of a middle-size plant is depicted in Fig. 3.1: we will hereafter refer to this example as a case study for the proposed system.

3.4.2 Topological Layer (Top Layer)

The Top Layer is the most abstract layer, and considers the topological representation of the factory or warehouse as a set of *sectors*. Sectors correspond to plant areas, such as intersections, corridors, load or unload areas, relevant for traffic management. In particular, each sector $\mathcal{S}$ can be distinguished from the others according to either topological, geometrical or logistical characteristics or other particular constraints.

The set of interconnected sectors defines a directed graph $\mathcal{V}$, referred to as the *Sectors Graph*. Each node $S \in \mathcal{N}(\mathcal{V})$ represents a sector and is defined as follows. Let $\mathcal{N}(\mathcal{G}_S)$ be the subset of nodes in $\mathcal{N}(\mathcal{G})$ located inside the sector S, where $\mathcal{G}$ is the plant graph and $\mathcal{N}(\mathcal{G})$ the respective set of nodes (see Section 3.4.1). Thus, we have that $\mathcal{G}_S$ is a subgraph of $\mathcal{G}$ and $\mathcal{E}(G_S) \subseteq \mathcal{E}(\mathcal{G})$. We also assume that all nodes in $\mathcal{N}(\mathcal{G})$ cannot lay on the boundary of two sectors in $\mathcal{V}$, i.e., for each $u \in N(\mathcal{G})$, $u \in N(S_i) \wedge u \notin N(S_j) \iff S_i = S_j$. It is worth noting that the subgraph $\mathcal{G}_S$ is not necessarily connected.

Let every edge in $\mathcal{E}(\mathcal{V})$ be the connection between two sectors. Let $S_1, S_2 \in \mathcal{N}(\mathcal{V})$ be two sectors, then an edge in the plant graph $\mathcal{G}$ that connects a node in G_{S_1} with a node in G_{S_2} implies that an edge in $\mathcal{E}(\mathcal{V})$ exists to connect S_1 with S_2.

We remark that the main role of the *Sectors Graph* is to model the traffic of vehicles in

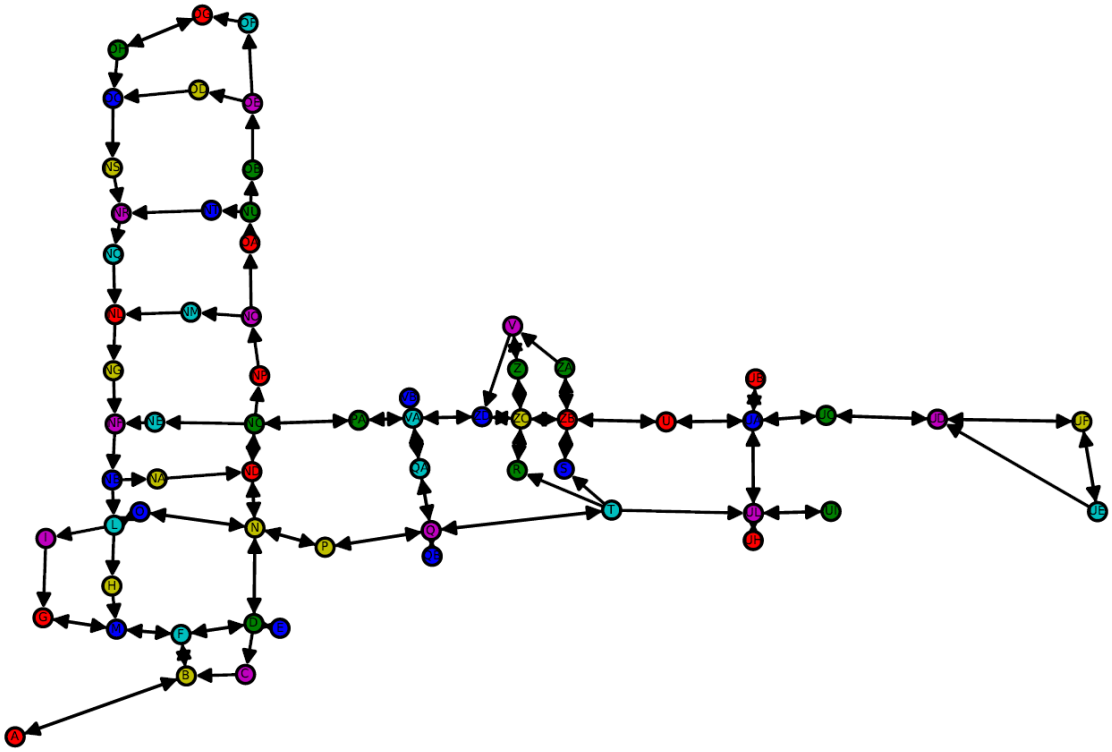


Figure 3.4: A full representation of the sector graph $\mathcal{V}$ used to model the traffic in our use case, which is elicited by a real industrial application.

the plant and, through the subsector graph (which will be defined in section 3.4.3), to define a traffic-sensitive path planner (i.e., to avoid congestions.) Each plant area defined by the respective sector can manage a maximum number of AGVs without occurring in traffic issues and congestions. This number is the capacity of the sector. In the proposed approach the sector's capacity is set by the user considering the plant area dimension, its topology and the presence of escape ways for vehicles in case of high traffic situations. A sector is labeled as congested if it contains more vehicles than its capacity.

The *Sectors Graph* of the considered use cases is shown in Fig. 3.4. Every node of the graph $\mathcal{V}$ keeps the information about the respective sector capacity. The weight of the edge that connects the sector i with the sector j is computed at time t as:

$$w_{i,j}(t) = T_j(t) + D_{i,j} \tag{3.1}$$

where T_j is the traffic weight for the specific sector j defined as follow:

$$T_j(t) = \begin{cases} K \cdot \frac{N_j(t)}{C_j - N_j(t)} & \text{if } C_j > N_j \\ K \cdot C_j & \text{otherwise} \end{cases} \tag{3.2}$$

where $N_j(t)$ is the number of AGVs within the sector j at time t, C_j is the capacity of the sector j, $D_{i,j}$ is the Euclidean distance computed between the centers of the two sectors i and j, which is proportional to the travel time given the constant velocity of the AGV, and K is a static gain. Therefore, the path planner has the possibility to compute the optimal path that not only minimizes the distance to the destination but also takes into

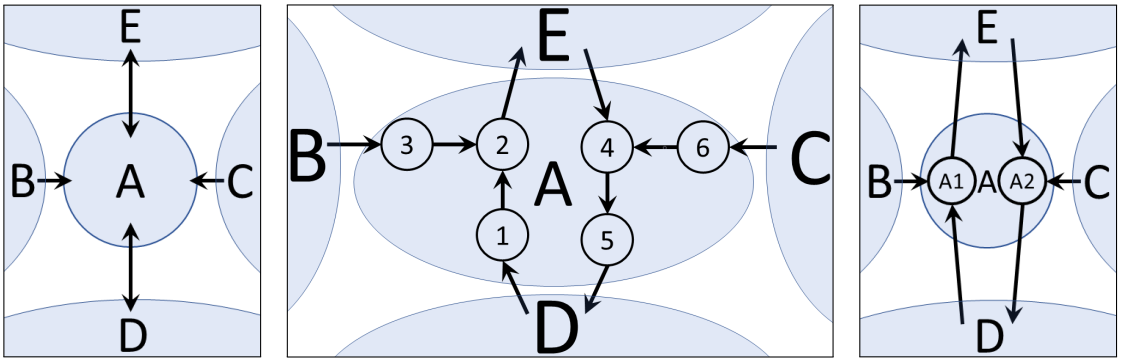


Figure 3.5: An example of subsector division process. (a) connections among sectors A, B, C, D, E in $\mathcal{V}$; (b) nodes of $\mathcal{G}$ belonging to sector A, the subgraph defined by the set of nodes $\{1, .., 6\}$ is not connected; (c) the result of the subsector partitioning algorithm.

account the traffic situation and the potential for congestion (see Sec. 3.4.3).

Partitioning the environment in sectors is a critical operation, that requires human intervention. The current algorithm for sector partitioning is not fully automated and still relies on human expertise. In particular, human operators are required to provide the boundaries between different sectors and each sector capacity since both are application dependent. Conversely, the set $\mathcal{N}(\mathcal{V})$ can be automatically computed, e.g., using the Voronoi partitioning proposed in [143]. The sector is individuated to isolate a particular area in the plant, such as intersections, corridors or load/unload locations or parking locations. Once the sectors have been defined, it is then necessary to tune each sector capacity based on the specific characteristics of the respective plant area.

3.4.3 Middle Layer

The Middle Layer has a similar definition to that of the Topological Layer and it allows to have a traffic sensitive path planner by exploiting the information stored in the Sectors Graph $\mathcal{V}$ thorough the parameter in (3.2). Over this layer, the controller integrates the traffic information collected on the Top Layer with the aim to obtain the routes, defined on the Roadmap Layer, to be followed by the AGVs.

Specifically, for each sector in the sectors graph $\mathcal{V}$, one or more subsectors are defined in the *Middle Layer*, which are collected in the *subsector graph* $\mathcal{W}$ (see Fig. 3.3). Thus, similarly to $\mathcal{V}$ on the Top Layer, we define the *subsector graph* $\mathcal{W}$, in which each node is a subsector (that will be defined hereafter) and each edge is a connection between two subsectors. The subsector graph is exploited by the controller to define a feasible route on the roadmap that minimizes the traffic of vehicles modelled by the sector graph. Finally, the planner generates from the path on the subsector graph $\mathcal{W}$ the optimal route on the plant graph $\mathcal{G}$.

In detail, a subsector is a subset of a sector and it is automatically generated from the clustering of the roadmap nodes belonging to that sector. In particular, Let $S \in \mathcal{N}(\mathcal{V})$ be a sector, and $\mathcal{G}_S$ the respective subgraph, then $\mathcal{N}(\mathcal{G}_S)$ is the set of nodes on the plant graph that belongs to the sector S, such that $\mathcal{N}(\mathcal{G}_S) \subseteq \mathcal{N}(\mathcal{G})$. Thus, let $U_S \in \mathcal{N}(\mathcal{W})$

be a subsector and $\mathcal{G}_{U_S}$ the respective subgraph: then, $\mathcal{N}(\mathcal{G}_{U_S})$ is the set of nodes in the plant graph that belong to subsector U_S, such that $\mathcal{N}(\mathcal{G}_{U_S}) \subseteq \mathcal{N}(\mathcal{G})$. Finally, we have that U_S is a subsector of S if $\mathcal{N}(\mathcal{G}_{U_S}) \subseteq \mathcal{N}(G_S)$.

As mentioned in Section 3.4.2, for a sector S, the associated graph G_S may not be connected: as a consequence, a path may exist on the sector graph $\mathcal{V}$ that does not correspond to a path on the plant graph $\mathcal{G}$. Hence, planning a path on the sector graph may lead to an unfeasible path on the plant graph (i.e., it does not exist any feasible route). Subsectors are defined to avoid this issue, ensuring that, if a path exists on the subsector graph $\mathcal{W}$, then it corresponds to a path on the plant graph $\mathcal{G}$. Figure 3.5 illustrates an example of subsector definition in order to better understand this process. Looking at the sector graph, shown in Fig. 3.5a, a path exists on graph $\mathcal{V}$ constituted by the sequence of nodes $\{B, A, D\}$. However, looking at the plant graph, depicted in Fig. 3.5b, it is possible to notice that, starting from a roadmap node in sector B, no path can be found in the plant graph $\mathcal{G}$ to reach a node in sector D. This is due to the fact that G_A, namely the subgraph associated with node A (i.e., the subgraph of $\mathcal{G}$ whose node set corresponds to the set of nodes that belong to sector A) is not connected. Considering again the example depicted in Fig. 3.5b, sector A is partitioned into two subsectors, namely A_1 and A_2, which include nodes $1, 2, 3$ and $4, 5, 6$, respectively.

The subsectors are determined from the sectors graph. We consider two neighboring sectors S_h and S_k, associated with subgraphs $\mathcal{G}_{S_h}$ and $\mathcal{G}_{S_k}$, respectively. Given a node $n_1 \in \mathcal{N}(G_{S_h})$, if a path exists that connects n_1 to any node in $\mathcal{N}(\mathcal{G}_{S_k})$, then we say that n_1 is *connected* to sector S_k. Thus, we define a subsector U_S of S_h as the subsector that contains all the nodes in $\mathcal{N}(\mathcal{G}_{S_h})$ that are connected to the same neighboring sectors.

Moreover, let U_{S_1}, U_{S_2} be two subsectors over $\mathcal{W}$. Let U_{S_i} belonging to S_i and $\mathcal{G}_{U_{S_i}}$ be the subgraph of $\mathcal{G}$ associated with S_i, $i \in \{1, 2\}$. Then, an edge from U_{S_1} to U_{S_2} exists in $\mathcal{E}(\mathcal{W})$ if there exists a path over $\mathcal{G}$ connecting two nodes $(u, v) \in \mathcal{N}(\mathcal{G}_{U_{S_1}}) \times \mathcal{N}(\mathcal{G}_{U_{S_2}})$. The weight of the edge directed from U_{S_1} to U_{S_2} coincides with the weight of the edge directed from S_1 to S_2, and, thus, it is defined according to (3.1). The weight on the edge that connects two subsectors belonging to the same sector is zero.

It is worth noting that, in the extreme scenarios, a subsector can coincide with a single node or with the whole sector it belongs to, depending on the connectivity of the graph defined on the nodes within the sector. Each sector contains at least one subsector. Moreover, let $\mathcal{G}_{U_S}$ be a subgraph of the plant graph $\mathcal{G}$, defined by a subsector $U_S \in \mathcal{W}$, whose node set is given by $\mathcal{N}(\mathcal{G}_{U_S})$, then $\mathcal{G}_{U_S}$ is at least weakly connected by definition.

3.5 Hierarchical Path Planner

In this section, we describe the proposed strategy based on the hierarchical architecture for the multi-AGV path planning. The method builds upon the architecture presented in Section 3.4 and consists of exploiting all the information provided by the layers to obtain a traffic sensible route planning and replanning strategy robust to errors and uncertainties introduced by a real scenario. The coordination of the fleet of vehicles to avoid conflicts

and deadlocks, performed by the Traffic Manager, is described in Section 3.6.

3.5.1 Path Planning

We assume that each AGV is assigned to a mission, which involves moving from a start location $v_s \in \mathcal{N}(\mathcal{G})$ to a goal location $v_g \in \mathcal{N}(\mathcal{G})$[1].

Let $S_s, S_g \in \mathcal{N}(\mathcal{V})$ and $U_{S_s}, U_{S_g} \in \mathcal{N}(\mathcal{W})$ be the subsectors and sectors containing the start and goal, respectively. Thus, a path is computed on $\mathcal{W}$ that connects subsectors U_{S_s} and U_{S_g}.

The traffic status, which characterizes every sector $S \in \mathcal{N}(\mathcal{V})$, is monitored and modeled on the Top Layer over the sectors graph and, at each time t is shared with the respective subsectors U_S in $\mathcal{W}$. Thus, the traffic weight is the same for all the subsectors belonging to the same sector, and is the one computed on the Top Layer for the respective sector according to (3.2). Finally, the edges in $\mathcal{E}(\mathcal{W})$, similarly to the edges in $\mathcal{E}(\mathcal{V})$, are weighted according to (3.1). Therefore, the path planner aims to optimize the balance between the distance, and the time, needed for the AGV to reach its destination and the level of traffic and the likelihood of congestion in certain areas of the plant.

Finally, the path, represented as a sequence of subsectors, is transformed by the planner into a corresponding route on the roadmap, represented as a sequence of nodes. The roadmap layer contains the information required to model the vehicles' trajectories over time. The route, composed of a sequence of sampled locations within the plant, describes the amount of time necessary for the vehicle to traverse each section of the road as defined by an edge on the plant graph. Therefore, as the average velocity of the AGVs is assumed to be known and constant, given a planned path, it is possible to predict the vehicles' locations over time, as well as the evolution of the traffic within each sector and, ultimately, throughout the entire plant. At every time step, the path for each AGV is replanned and the traffic status is updated consequently. Thus, the optimal solution found by the planner is the best compromise between the traffic conditions and minimal path to the goal.

The planner on the Middle Layer computes the optimal path $\pi_W = \{U_1, \ldots, U_n\}$ on graph $\mathcal{W}$, which aims for each AGV to avoid congested areas minimizing the time required to reach the goal. The path planning algorithm is based on the Dijkstra's algorithm [168] on $\mathcal{W}$, with edge weights defined as in (3.1). In other words, to exploit the traffic modelling, the properties of each sector are inherited by all its subsectors.

Finally, the path π_W computed over W becomes a constraint for the route π_G computed over the roadmap and plant graph $\mathcal{G}$. Hence, the route π_G is planned over a portion of the plant graph $\mathcal{G}$, which consists in a subgraph of $\mathcal{G}$ constituted by only the nodes in $\mathcal{N}(\pi_W)$, where $\mathcal{N}(\pi_W) \subseteq \mathcal{N}(\mathcal{G})$ is the union of the sets of nodes belonging to every subsector in π_W. The route planning consists in searching a feasible path from v_s to v_g on the defined portion of plant graph exploiting the Dijkstra's algorithm. It is worth noting

[1]Mission assignment is interesting yet beyond the scope of this work. The interested reader is referred, e.g., to [175] for previous studies on the topic.

that, since the subgraph of $\mathcal{G}$ computed on $\mathcal{N}(U_i)$ is guaranteed to be at least weakly connected, a route can always be found. Each AGV's path is dynamically replanned in order to face variations in the traffic status and occurrence of unpredictable events (e.g., the AGV slows down or stops due to the presence of human operators). Each time a path is recomputed, the traffic status is updated to include the last information about the current (and future) locations of the AGVs. The more frequent is the path replanning, the more robust is the traffic coordination, since a frequent path replanning allows a robust traffic coordination with respect to dynamic obstacles, traffic congestion, or tracking/communication errors.

The proposed planning and replanning strategy is summarized in Algorithm 2 and consists in the following steps:

i. The optimal path π_W from the start subsector to the goal subsector is computed/updated over $\mathcal{W}$ (Algorithm 2, line 2). The path planner minimizes a combination of the traveled distance, and hence the expected travel time, and the currently known traffic cost.

ii. The path expressed as a sequence of subsectors π_W corresponds to a sequence of sectors π_V, which are the sectors that will be crossed by the AGV following the planned path. Thus, the number of vehicles inside each sector can be monitored and a model of the traffic status over time can be easily built. The sector graph $\mathcal{V}$ weights are updated according to (3.1) following the traffic status changes (Algorithm 2, lines 3 and 4).

iii. The traffic information held by the sector graph V is inherited by the subsector graph W. Hence, the weights of the set $\mathcal{E}(W)$ are updated according to the new traffic information (Algorithm 2, line 5).

Algorithm 2: Traffic modelling and path planning

```
1 if state != goal then
2 |   π_W ← dijkstra(W);
3 |   T ← update_traffic_status(π_W);
4 |   V ← update_weights (V, T);
5 |   W ← update_weights (W, V);
6 end
```

3.6 The Traffic Manager

The centralized and decoupled controller requires an efficient coordination strategy in order to avoid collisions inside the paths computed previously by the planner. Thus, a proper coordination algorithm together with the described path planning algorithm complete the traffic manager proposed in this work. Inter-agent collisions are then avoided by exploiting the coordination strategy described in this section, which is inspired by the coordination diagram approach described by [155]. The proposed strategy aims to an

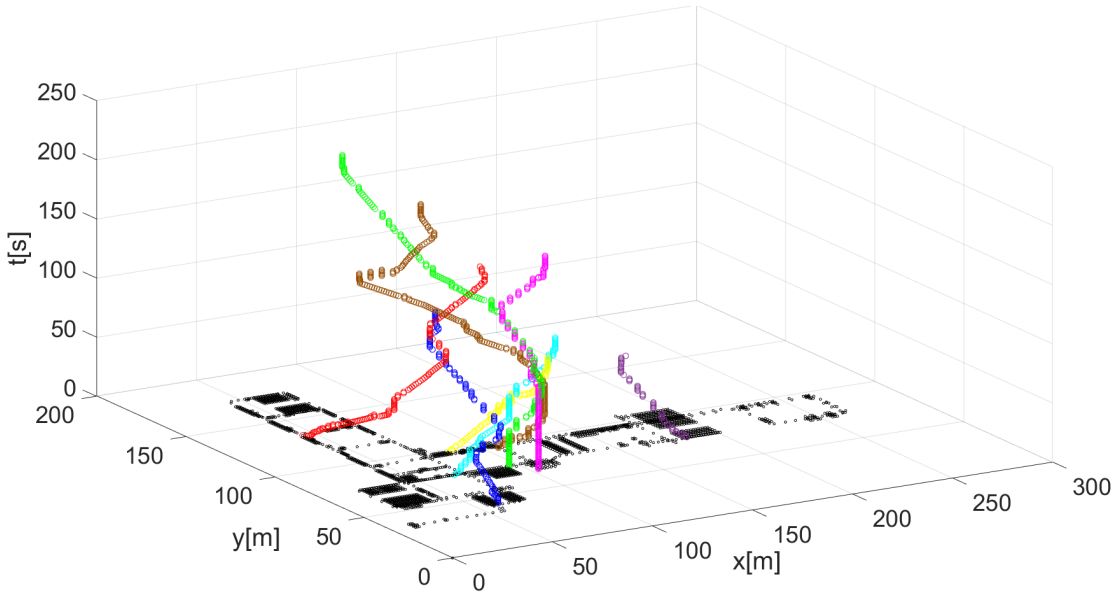


Figure 3.6: Visualization of coordination over time of 8 AGVs in the use case plant. The figure provides a visualization of the coordination of 8 AGVs evolved over time in the use case plant considered. On the plane XY the figure shows the node coordinates (the location) in the plant, while, on the Z axis the figure shows the time at which a node is crossed. Different colors indicates different vehicles, i.e. red is for AGV with number 1, blue is for AGV with number 6, etc. It is worth noting that the lines, as sequences of coloured nodes, never cross, since the simultaneous presence of two (or more) AGVs on the same node at the same time never takes place. This means that the coordination has been carried on properly and there are no collisions in the AGV trajectories.

efficient negotiation between colliding vehicles to manage the traffic and avoid deadlocks. The main traffic manager loop is resumed in Algorithm 3, where the coordination process is executed in line 18. A mission is assigned to an AGV only when idle. Each mission involves achieving a loading location, loading a cargo, moving to the unloading location and unloading the cargo. If the AGV has no more missions to be assigned, it is automatically sent to the battery charge location. At every time step, the AGVs positions are updated since the AGV location is frequently monitored in order to deal with delays in the trajectory following. Every time period T the path of each vehicle is computed again and the weights in the sector and subsector graphs are updated according to the new traffic status. The smaller is the time period T, the more frequent is the path computation and the traffic update in the execution of the traffic manager algorithm. The path planning is followed by the coordination algorithm (Algorithm 3, line 18) to deal with collisions and deadlocks. Every time the coordination algorithm is executed the vehicles motion over time is changed in order to face conflicts and, hence, the traffic status has to be updated accordingly (Algorithm 3, line 19).

The higher is the value of T, the lower are both the computational effort and the robustness to dynamical obstacles or vehicle errors. Low values of T imply frequent optimal paths computation and traffic status updates, which makes the control more responsive to unpredictable events and traffic congestions.

Algorithm 3: Traffic management

```
1  Compute_initial_traffic_and_paths();
2  t ← current_time;
3  T ← time step update;
4  while true do
5      update_AGVs_positions();
6      if current_time ≥ t + T then
7          foreach AGV do
8              startnode ← current_position;
9              if goalnode then
10                 update_path(Algorithm 2);
11                 π_G ← π_W;
12             else
13                 goalnode ← assign_new_mission();
14                 update_path(Algorithm 2);
15                 π_G ← π_W;
16             end
17         end
18         coordination_AGVs(Algorithm 4);
19         update_traffic();
20         t ← current time;
21     end
22 end
```

Algorithm 4: AGVs coordination

```
1  H ← time horizon window;
2  t ← current time;
3  collision_free ← false;
4  while not collision_free do
5      foreach t in H do
6          foreach AGV do
7              collision_free ← get_conflicts(t,AGV);
8              if not collision_free then
9                  AGV_winner, AGV_losers ← negotiation(t,AGV);
10                 AGV_winner, AGV_losers ←
                    deadlock_policy(t,AGV,AGV_winner,AGV_losers);
11                 solve_conflict(t,AGV, AGV_winner,AGV_losers);
12             end
13         end
14     end
15 end
```

3.6.1 AGVs Coordination

The path planner computes the route that optimizes the balance between the vehicles traveled distance, and hence the traveled time, and the level of traffic over the sectors and the plant. The path planner provides the necessary information to consider the evolution of the trajectory over time. Therefore, once the paths are computed for each AGV, we need to verify that no collision occurs during the vehicles trajectory following. We say that there is a conflict (or a collision) whenever two or more AGVs are required to occupy the same node/edge at the same time. Conflicts are detected while assuming the current paths are traversed with the robot's nominal velocity profile (e.g., minimum time) and accurate tracking.

More in details, neglecting for the moment delays or execution errors, and assuming a nominal trajectory profile, we are able to precisely predict the location of each AGV over a predefined time horizon H. Therefore, conflicts and congestions can be predicted and prevented by sequencing the robot access through the shared resource according to priorities and negotiation. It is worth noting that the vehicles stops or slowdowns are decided by the coordination strategy and, hence, the trajectory model over time and the prediction of the vehicles locations is updated accordingly. This computation over time on the future locations traveled by the vehicles implies longer execution time but ensures conflict prevention.

To ensure accurate and efficient operation, the location of each vehicle is continuously measured and monitored, in order to promptly identify any execution errors that may cause delays in the trajectory following and requiring an update in the prediction of the AGVs' locations over H. The potential delay of the vehicle on the followed trajectory is subsequently incorporated into the time model of the vehicle and into the coordination strategy to account for any execution errors, thus enabling a reliable prediction of the robot's location. This is crucial for managing future conflicts between the AGVs and preventing deadlocks and congestion.

The selection of the parameter H plays a crucial role to determine how large the prediction time window is, the computational cost and the effectiveness of the coordination algorithm. It follows that as the value of H increases, the system exhibits greater robustness against deadlock and congestion, as there is a larger temporal margin for preventing potential conflicts. However, the higher is the value of H, the higher will be the computational burden of the coordination algorithm. The selection of the parameter H is contingent with the structure of the factory, and, as such, the layout of the plant. In particular, the time horizon has to be chosen sufficiently large to allow the coordination strategy to anticipate and manage the vehicles movements in the longest dead-end present in the plant. Inadequate choice of H will result in suboptimal performance and may lead to vehicles being stuck in a dead-end. Furthermore, it should be noted that the value of H cannot be selected arbitrarily large due to the computational cost associated with the coordination algorithm and thus, a trade-off must be found between the desired performance of the algorithm and the computational resources that are available.

The result of a correct traffic management and, hence, of a correct coordination of AGVs

over time can be visualized in Fig. 3.6, where the locations (or nodes on $\mathcal{G}$) at every time step of each AGVs are plotted. The figure shows how the routes of the AGVs, distinguished with different colors, do not intersect each other when extended in a prediction over time. This means the vehicles are able to reach the final destination without occurring into conflicts and congestions.

Algorithm 5: Negotiation

```
1  AGV_winner ← None;
2  AGV_losers ← None;
3  AGV_winner,AGV_losers = motion_error(conflicting_AGVs);
4  if (AGV_winner is None) OR (AGV_losers is None) then
5  |  AGV_winner,AGV_losers = path_obstruction(conflicting_AGVs, T);
6  end
7  if (AGV_winner is None) OR (AGV_losers is None) then
8  |  AGV_winner,AGV_losers = time_to_conflict(conflicting_AGVs);
9  end
10 return AGV_winner,AGV_losers
```

The coordination strategy of the AGVs is resumed in Algorithm 4 and can be divided in 4 main phases:

i. *conflict identification*: the localization in time and space of a conflict between a group of AGVs, i.e. AGVs occupying the same area at the same time. The time horizon H defines how far in time a conflict can be detected and possibly avoided. H is lower-bounded by the topology of the plant, since it has to be chosen wide enough to avoid two AGVs stuck in a bidirectional single vehicle corridor. Every future time steps, hence, has to be checked for possible collisions (Algorithm 4, line 4).

ii. *negotiation*: once we know which are the AGVs in conflict and at which time step the collision will happen, we have to define a policy to assign the precedence on the motion to the vehicles. The negotiation to define which AGV has to wait to solve the conflict is based on simple rules. These rules consider the mission priority, if the AGV has to free an intersection, if the vehicle is stopped due to an internal error and the time required to reach the collision spot. The proposed algorithm incorporates a mechanism to assess with simple rules the status of vehicles with regards to their execution and movement. If a vehicle is determined to be stopped due to an execution error, it is deemed as the most probable loser and would have to wait for other vehicles to complete their motion (line 3, algorithm 5). Additionally, the algorithm assesses the potential path obstruction between vehicles in the portion of the route considered within the time window T. The vehicle that obstructs the path of another vehicle is considered as the most probable winner and is selected to clear the passage in order to prevent further obstructions (line 5, algorithm 5).

   Finally a simple FIFO policy is chosen to end the negotiation, which means that the first vehicle that reaches the collision spot is the negotiation winner, and other AGVs involved in the collision wait for their motion (line 8, algorithm 5). Hence, at the end of the negotiation there will be a winning AGV and (possibly multiple) loser

AGVs (Algorithm 4, line 9). Those rules can be chosen to minimize or maximize a performance index by solving an optimization problem online but, for this specific case, we consider some very basic simple rules, since the efficiency of the solution of the negotiation algorithm is checked by the deadlock prevention policy which guarantees the coordination of the vehicles.

iii. *deadlock prevention*: the use of simple rules for the negotiation policy could lead to the generation of deadlocks. Hence, the proposed coordination strategy is embedded with an effective deadlock prevention policy (DPP), which has the aim to identify and avoid deadlocks before they occure. The DPP we propose, described in details in the following section, extends the concept of precedence graph used in multitasking systems by introducing a temporal variable that is necessary and efficient for identifying and avoiding deadlocks in time.

iv. *solve conflict*: the conflict is ready to be solved once the conflicting time step is found and the loser(s) and winning AGVs are defined. The list of commands to be send to the loser(s) is modified to make the AGV(s) waiting for a time length sufficiently large to solve the collision. The process is repeated to check for other collisions and solve them until no more collisions are found in the time window H.

3.6.2 The Deadlock Prevention Policy

The implementation of a deadlock prevention policy is an essential strategy for maximizing the possibilities of finding an optimal solution in the coordination of AGVs. The negotiation process for resolving conflicts is commonly based on a set of straightforward rules and can potentially result in deadlock situations that cannot be resolved.

We can identify two types of deadlock scenarios: cyclic deadlocks and acyclic deadlocks, that differ in the nature of the resource locked between the processes involved [176]. Cyclic deadlocks represent a category of deadlock in which the resources that are locked in place exhibit a circular dependency. This is characterized by the existence of a chain of processes, each of which possesses a resource that is being sought by the subsequent process within the chain. Acyclic deadlocks, on the other hand, are a type of deadlock in which the resource dependency forms a tree or a directed acyclic graph (DAG) structure. Hence there are no circular dependencies and each process holds a resource that is not required by any other process in the chain. It is worth noting that in our scenario, acyclic deadlocks are generally generated by a vehicle that is halted as a result of an error, which subsequently locks the resource in question, which in this case is the moving space. Therefore, the coordination strategy lacks the capability to address this issue, and the traffic manager must take steps to update the trajectories of the vehicles in order to limit the congestions until the error is resolved. Finally, second level deadlocks or nested deadlocks are acyclic phenomena typically generated by a cyclic deadlock. In fact, the vehicle that locks resources and generates a tree dependency structure is halted in a cyclic chain of locked resources. Therefore, by preventing the emergence of cyclic deadlocks, it is also possible to prevent the occurrence of nested deadlocks. In the presented control architecture we propose a deadlock prevention policy based on time expanded graphs.

The result of the negotiation process, for every future conflict found in the time window H, sets which are the AGVs that have to stop (the losers) and the ones that have to move (the winners) in order to solve the future collision. The set of precedences between the conflicting AGVs in a particular time step t can be represented by a graph. In the proposed approach we exploit the time expanded graph in order to represent the set of precedences between conflicting AGVs in the whole time window H. The analysis of this graph of precedences becomes useful when dealing with deadlocks, since these are caused by cyclical precedences in the time window H [165]. Hence, having an acyclic time expanded graph representing the precedences ensures that the computed trajectories in H are deadlock free.

The time step t at which the collision is predicted to happen is denoted as $t_{conflict}$. In order to solve the conflict, the AGV that has lost the negotiation has to be stopped or slowed down a time window before the collision time in order to allow the winning AGV to cross. This time window is denoted as t_{delta}, and the time step t at which the AGV is stopped is denoted as t_{wait}, so that $t_{wait} + t_{delta} = t_{conflict}$. Every time the future trajectory to be followed by the AGV is changed to solve a collision, the prediction of the AGV motion has to be updated accordingly introducing the delay caused by the stop or slowdown of the vehicle.

The negotiation process brings to a set of possible solutions to solve the collisions in the time horizon H, and the deadlock avoidance process is responsible in finding the deadlock free solutions. The time expanded graph is generated over the set of precedences in the time window H obtained from the negotiation process. Every graph node indicates the AGV ID and every edge represents a precedence, with a winner AGV and a loser AGV: for each incoming (outgoing) edge the AGV is a winner (loser) for the respective precedence.

For every collision detected in the predicted trajectory, the time expanded graph is built iteratively, every precedence edge is added step by step from the $t_{conflict}$ until the t_{wait}, when the conflict is solved. For example, Fig. 3.7 shows a coordination between 3 AGVs: AGV 1 and AGV 2 have a conflict predicted at time $t = t_4$ and, to avoid the collision with sufficient space, AGV 1 has to wait 3 time steps before at time $t = t_1$. Similarly, AGV 2 and AGV 3 have a conflict at time $t = t_5$, where the AGV 2 is stopped 2 seconds before at time $t = t_3$. Hence, at time $t = t_4$ and $t = t_3$ we have two conflict situations which have been solved with AGV 1 stopped for AGV 2 and AGV 2 stopped for AGV 3.

After every negotiation at every time step the time expanded graph of precedences is updated and analyzed in order to ensure that the new edges introduced do not bring to a deadlock scenario. This is the case shown in Fig. 3.8, where the conflict between AGV 1 and AGV 3 at time $t = t_8$ could generate a loop cycle in the graph and, hence, a deadlock. The negotiation process gives the solution in which the AGV 3 is either a loser or a winner. In the first scenario, a loop cycle would occur at time $t = t_4$ generating a deadlock between the AGVs 1, 2 and 3. Exploiting the information stored in the time expanded graph, we can exclude the first scenario from the possible solutions and make the AGV 3 a winner adding an edge from node 1 to node 3.

Every time a new conflict is found in the predicted trajectories of the AGVs, the time

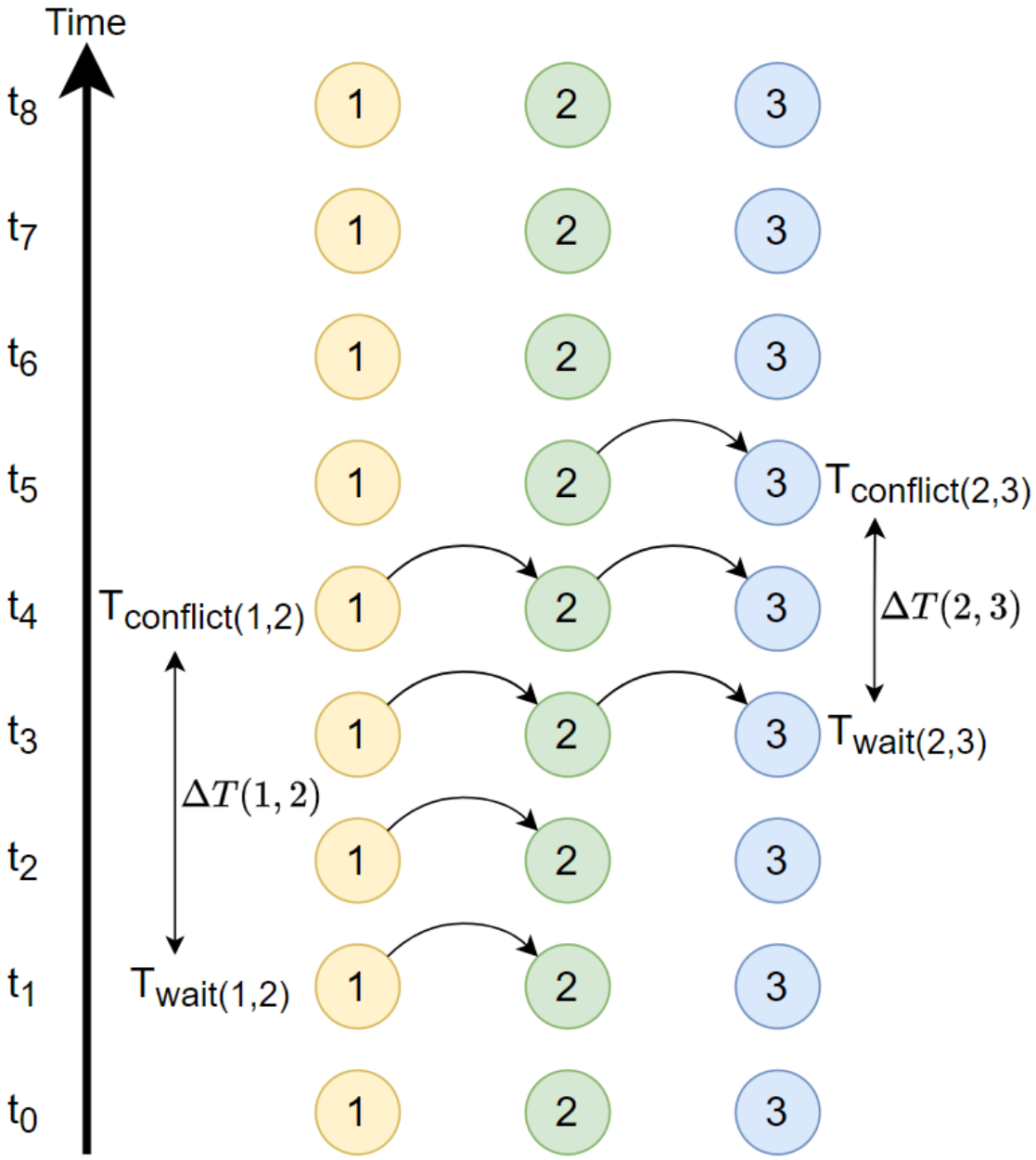


Figure 3.7: A representation through the time expanded graphs of the precedences defined to coordinate three AGVs. Every layer corresponds to the graph representing the precedences, as result of the vehicles coordination, at a particular time step. For example, at time t_4 the coordination is managed by making the AGV 1 waiting for the AGV 2 and the AGV 2 waiting for the AGV 3.

expanded graph is updated accordingly and every layer is checked to be acyclic to avoid the occurrence of deadlocks. Hence, if a cycle is detected, the found solution from the negotiation is not admissible and another one is evaluated.

3.7 Software Implementation

The proposed architecture was implemented, for validation and evaluation, on an industrial use case, exploiting the NetworkX library [177] in Python language. This library provides the tools to easily manage the graphs. The graphs are strongly related to the

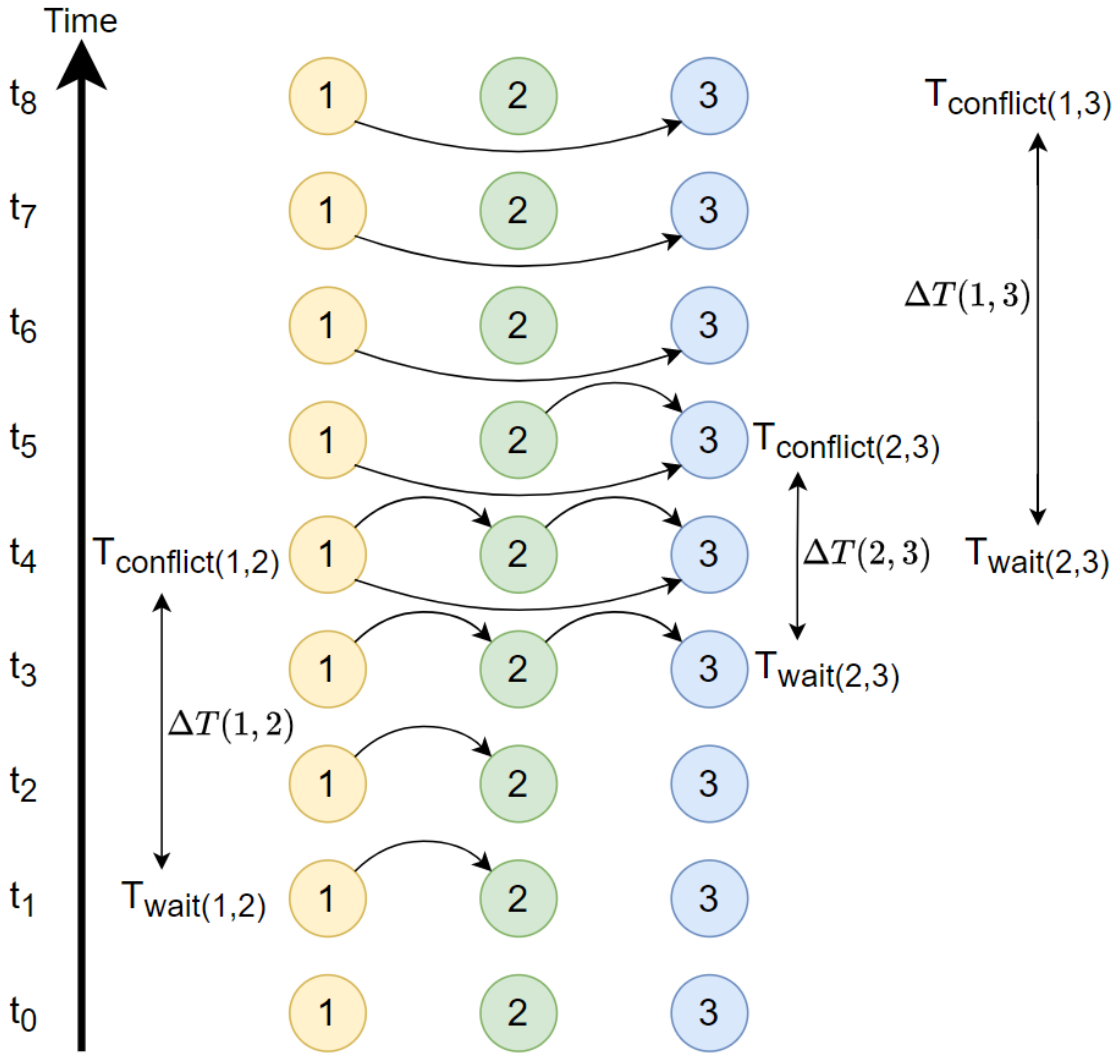


Figure 3.8: A representation through the time expanded graphs of the precedences defined to coordinate three AGVs. Every layer corresponds to the graph representing the precedences, as result of the vehicles coordination, at a particular time step. The time expanded graph helped to avoiding a cyclic set of precedences and hence a deadlock at time t_4. The coordination is managed by making the AGV 1 waiting for the AGV 3 and not vice versa to avoid a deadlock scenario between the three vehicles.

layout of the infrastructure (warehouse or factory) and are computed offline once the feasible routes in the plant are defined. Thus, the planner in the traffic manager does not see any change in the structure and connectivity of these graphs. However, the path computation is sensible to the changes in terms of attributes and weights linked with nodes and edges, which follow the changes in the traffic status. We remark that the route of each AGV is processed as a sequence of connected nodes. Every node occupied by an AGV means that a location in the plant is occupied by the vehicle. All the directions the AGV can follow from its location (node) are represented by the edges outgoing from the respective node.

The proposed control architecture makes the multi-AGV system able to deal with unexpected events, such as dynamic obstacles or vehicles errors, that change the expected traffic scenario. A frequent path replanning and traffic status updating is the key to allow the vehicles to avoid unforeseen congested areas due to dynamic objects. Thus, the coordination algorithm is frequently executed to keep reactive the coordination of

vehicles over the planned routes.

A software library has been developed to model the AGV motion over the time steps. Since the average of the AGV velocity is constant and known, the time required to travel every edge in the roadmap is known. Hence, the planned route, which consists in a list of adjacent nodes over the plant graph, can be projected over time. The coordination algorithm manages the vehicle possible conflicts stopping or slowing it down in free collision nodes, so that the time the AGV takes to reach the conflicting node is modified. Moreover, every time an unexpected event occurs, the schedule of the AGVs changes and an execution of the control algorithm is necessary to update the model of the AGVs motion over time. A frequent execution of the control algorithm is computationally demanding, but makes the systems able to compensate the delays introduced by vehicles alarms, obstacles or communication errors.

3.7.1 Computational Complexity

Coupled approaches typically have a time complexity that grows with the size of the configuration space, which grows exponentially with the number of robots and plant dimension [178]. The computational effort of the proposed software is mainly defined by two components: the path planning and the coordination of vehicles. The complexity of the algorithm to coordinate the AGVs avoiding conflicts and deadlocks is analyzed below. The path computation is lightened by the hierarchical architecture, specially in large environments and complex and high redundancy roadmaps: the Dijkstra's algorithm is modified to compute a path π_G over the plant graph $\mathcal{G}$ constrained to be contained into the subgraph of $\mathcal{G}$ defined by π_W, see Section 3.5. The path π_W is computed exploiting the Dijkstra's algorithm over the graph of subsectors $\mathcal{W}$. The computational efficiency of the proposed multi-layer path planning architecture is compared with the conventional approach, in which the Dijkstra search algorithm is applied directly on the roadmap graph(see Figure 3.9). In the conducted test, executed on a average laptop with Intel core i7-10510U and NVIDIA Geforce MX230, a range of graphs with varying dimensions and complexities were considered. The time needed to complete the path planning computation for 8 AGVs between two of the most distant nodes in the graph was analyzed using both the standard approach and the proposed multi-layer architecture. The results, depicted in Figure 3.9, demonstrate that as the dimension and complexity of the graph increase, the difference in computation time between the standard and proposed approach becomes increasingly pronounced. It is noteworthy that a low computational time for path planning is crucial for the effectiveness of the replanning feature in the proposed control strategy.

The computational effort incurred by the coordination algorithm, which includes the deadlock prevention policy, must also be taken into consideration. Analyzing Algorithm 4, similarly to [179], it consists in two main inherited loops over the number of AGVs N and over the time steps t in the time window H. The processes inside these loops are mainly independent from H and N and require a constant amount of resources to be executed. The deadlock avoidance algorithm requires a relevant computational effort.

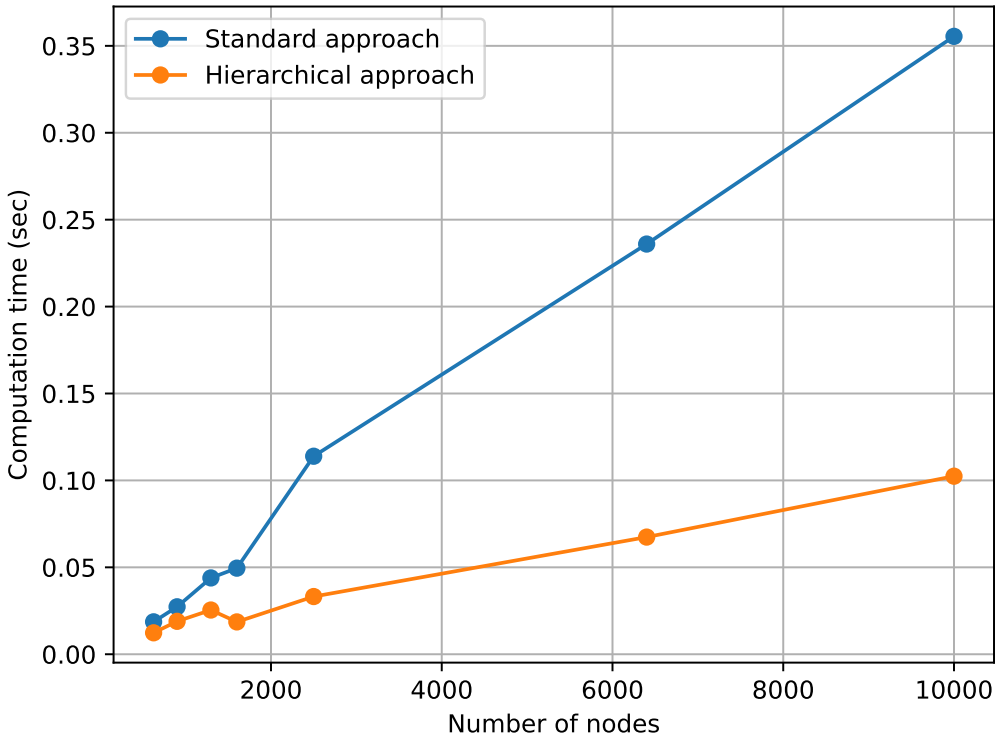


Figure 3.9: The figure illustrates the computational benefits of a decoupled approach as compared to a standard coupled approach in terms of the path planning computation. The graph displays the computation time in seconds required to calculate the path for 8 AGVs in a roadmap with varying size between two of the most distant nodes in the graph. The discrepancy in computation time becomes more pronounced as the dimension and complexity of the graph increases.

In order to determine the computational effort we consider the worst case scenario for the iterative generation of the time expanded graph: this is constituted by $H+1$ layers, each one containing N nodes. In the worst case, we suppose all the AGVs are in collision for the whole time window H. Thus, once an AGV winner is chosen, every layer of the time expanded graph has to be updated with all the edges representing the precedences between the AGVs. The computational complexity of the deadlock avoidance algorithm becomes $O(NH)$. Since this process is executed inside Algorithm 4, the computational complexity of the whole coordination strategy becomes $O(N^2H^2)$

The deadlock avoidance algorithm computational cost has been tested in some simulations since its high expected computational demanding. Starting from a set of precomputed paths from a set of critical mission previously chosen for the test, we evaluated the number of iterations required to end properly the coordination. The tests were executed varying the number of AGVs from 2 to 8. The results are shown in Fig. 3.10 and indicates that the average and the maximum number or iterations are well below the maximum theoretical values estimated previously. In fact, as a result of the efficient planning and traffic modeling described in Section 3.5, only a part of the AGVs operating in the system needs to be coordinated in order to avoid deadlocks, and therefore the worst-case scenario previously theorized is efficiently avoided.

We notify that the time required to complete the traffic manager computation is, typically, in the range of few seconds. In particular, the computation time has to be inferior to the time period T, where T determines how frequently the control algorithm is ex-

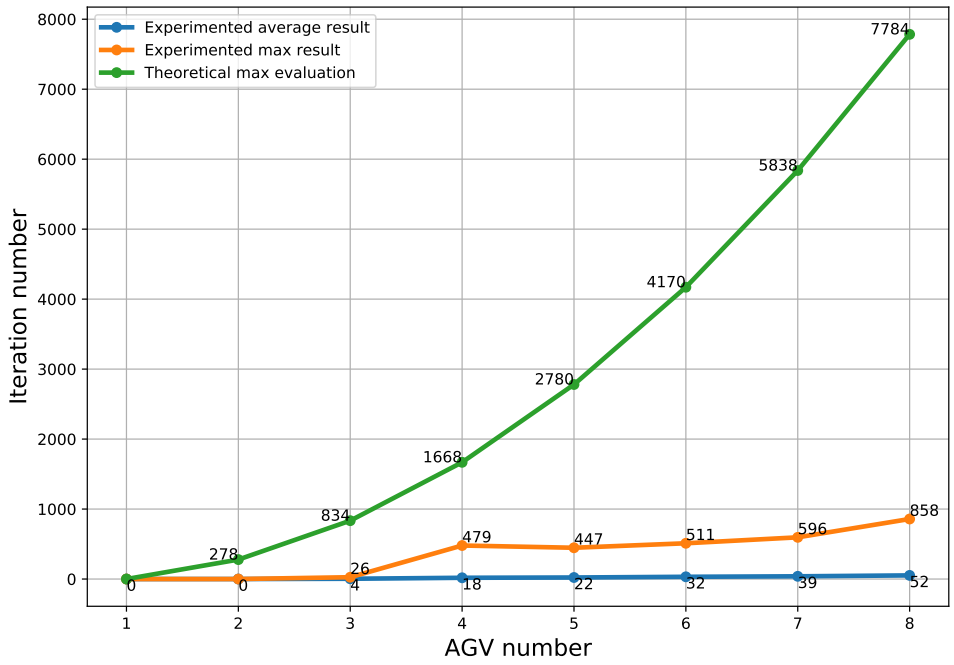


Figure 3.10: The figure show an evaluation of the deadlock prevention policy algorithm computational cost. We compared the theoretical exponential computational cost with the one tested with several simulations. The orange and blue line show respectively the highest and the average number of iterations tested from simulations. The green line shows the theoretical expected number of iteration in the worst computational scenario.

ecuted and, hence, how frequently the path planning algorithm and the coordination algorithm are executed (see Algorithm 3).

Finally, we conducted an evaluation of the efficiency in the traffic management and coordination of a variable fleet of AGVs in the considered industrial use case. We studied a comparison between the time the AGV is ideally required to reach its goal (i.e. without stops due to traffic, obstacles, ecc.) and the actual time, tested in simulations, with delays introduced by the vehicle coordination. In particular, we compute the management efficiency as the ratio between the time the vehicle is effectively moving and the overall time required to complete the task, i.e. the combination of the moving time and the waiting time of the vehicle: $\eta = T_{moving}/(T_{moving} + T_{waiting})$. Figure 3.11 shows the results about the efficiency η from the tests conducted with up to 8 AGVs. The efficiency decreases with a large fleet of AGVs, since the traffic management and the vehicles coordination increase in complexity and congestions are more likely to happen.

3.7.2 Simulations

The software developed on the proposed control architecture has been initially tested on a simulation environment provided by Proxaut s.r.l., which emulates the nominal activity of a working day in the automated factory. Moreover, the simulator allows to test unexpected events that normally occur during the factory operation, such as alarms, vehicle

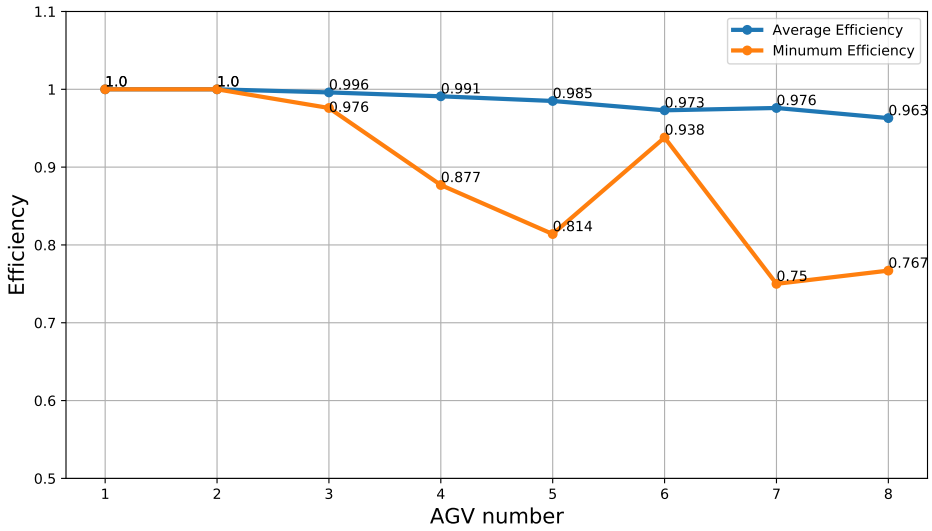
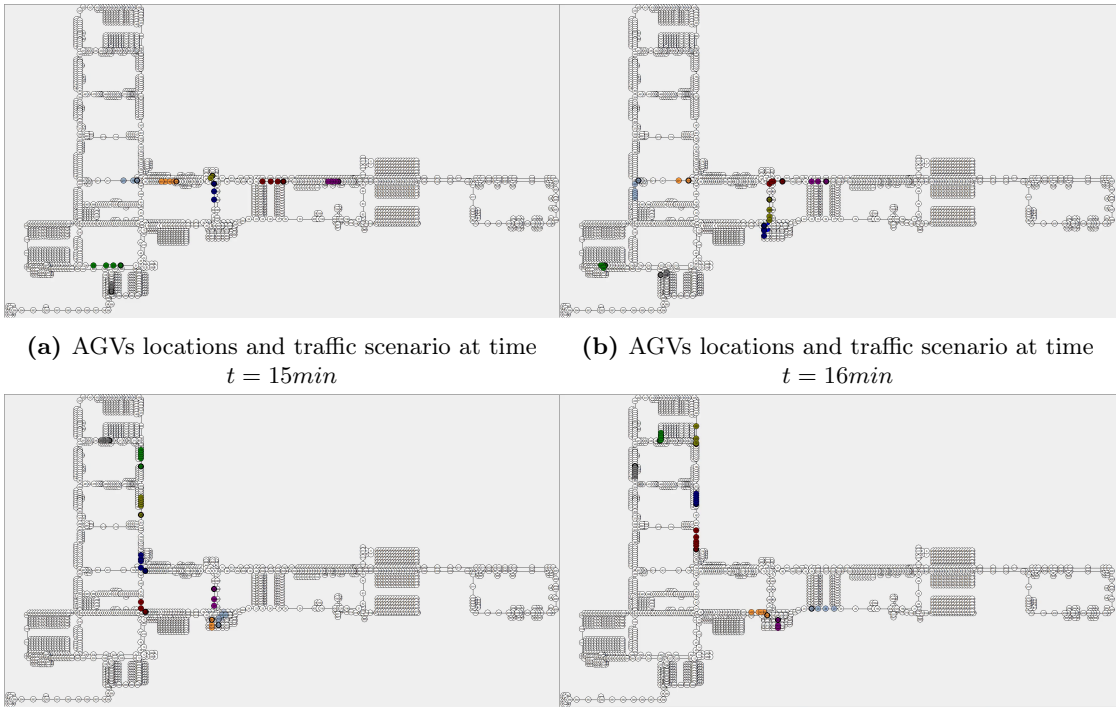


Figure 3.11: The figure show an evaluation of the efficiency in the traffic management and vehicles coordination. The results were obtained by a serie of simulations conducted on the industrial use case scenario. The efficiency parameter aims to determine how much time is wasted due to the coordination. The blue line shows the average efficiency seen in several tests. The orange line shows the worsts values obtained from some tests. It is worth noting how the efficiency only slightly decreases, increasing the number of AGVs.



(a) AGVs locations and traffic scenario at time $t = 15min$

(b) AGVs locations and traffic scenario at time $t = 16min$

(c) AGVs locations and traffic scenario at time $t = 20min$

(d) AGVs locations and traffic scenario at time $t = 21min$

Figure 3.12: The figure shows the simulator provided by the company exploited to test the software built upon the proposed control architecture. We tested the software with several hours of simulations managing the coordination and the traffic of 8 automatic guided vehicles in the studied plant. Every group of coloured dots indicates an AGV location, the immediate future location and its occupied space due to its size.

errors, communication errors and delays introduced by operators. Several simulations



Figure 3.13: A photo taken during the experiments conducted in the automated factory. The 4 AGVs are coordinating in a load and unload cargo area.

were conducted in order to test and stress the software built upon the proposed control architecture. The aim was to make the control software sufficiently robust, efficient and reliable to be implemented on the real industrial environment. Figure 3.12 shows an experiment where we tested the traffic management of 8 automatic guided vehicles over the studied use case. The simulation environment consists in the automated factory roadmap and simply considers all the locations that the AGV can travel in the plant. Every time step the simulator allocates to every vehicle the node it occupies, the nodes it will occupy in the near future and the neighbouring nodes modelling the size of the vehicle. These simulations shown the capabilities of the proposed control strategy to manage the coordination and the traffic of a fleet of 8 AGVs in the studied automated factory for several work shifts. Hence, these tests made the software ready for some experiments in the real plant.

3.7.3 Experimental validation: Industrial Scenario

Finally, the software has been implemented and validated on a real industrial environment in collaboration with the company, see Fig. 3.13. The critical parameters that require careful selection to optimize the performance of the software for a specific application are the time horizon H and update frequency T of the traffic manager. These parameters are chosen based on the dimensions and layout of the plant under consideration. For the considered use case, a time horizon of 100 seconds was selected, as the longest dead-end in the plant takes approximately 80 seconds to be traversed by the AGV. The computational complexity of the traffic manager algorithm is within the range of 7 seconds of execution, and thus, a time frequency of 10 seconds was selected, which is confirmed through multiple simulations. In the nominal operation, the software manages 8 AGVs over the provided plant. The performance of the control architecture proposed in this work has been compared to the commercial software developed by the company. The company's software implements a traffic coordination of the AGVs based on manually synthesized traffic rules, which are compared among all the AGVs at every iteration. Given the path for each AGV, that is a sequence of nodes, each iteration

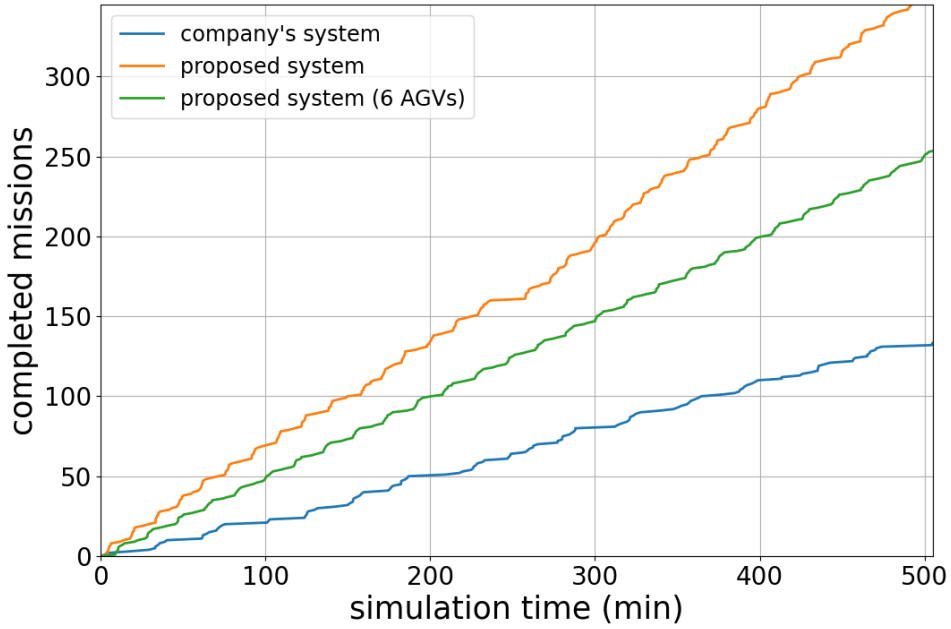
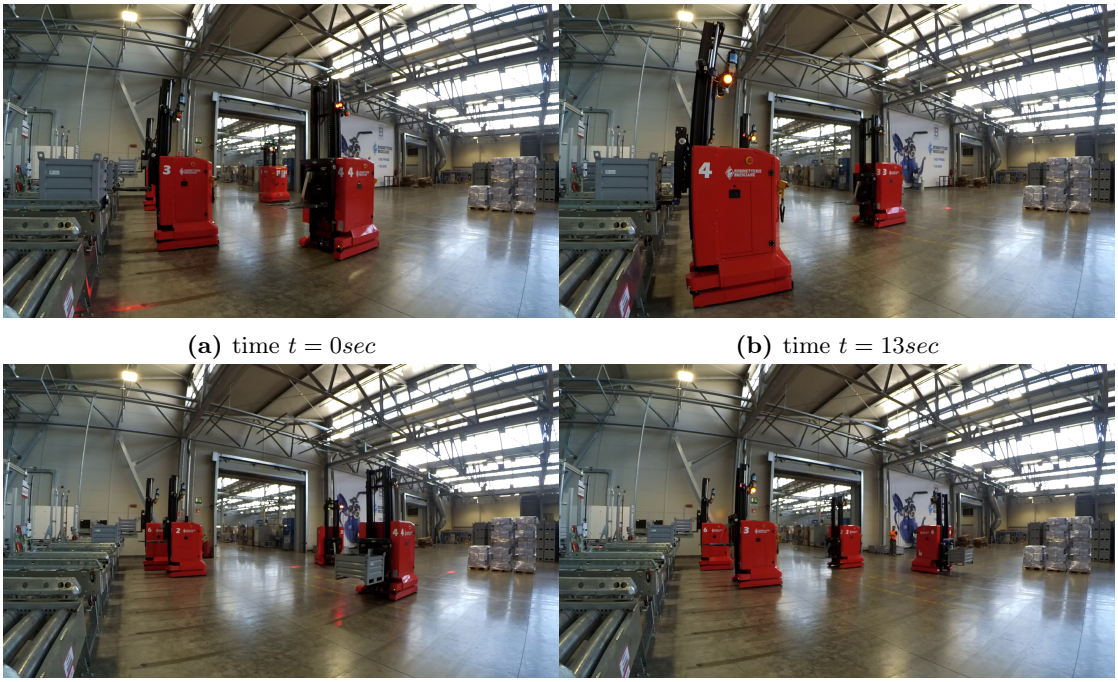


Figure 3.14: The figure shows the number of missions completed over time by the fleet of AGVs. The blue line indicates the company's software performance at full potential, while the orange one indicates the proposed software performance at full potential. The proposed software managed to coordinate 8 vehicles simultaneously. The company's software managed to coordinate 6 vehicles simultaneously. The green line shows the performance of the proposed software when constrained to move only 6 AGVs simultaneously.

consists in assigning the following node to each AGV. Moreover, this software does not consider the traffic status, and does not allow replanning the path: hence, the shortest path is assigned to each AGV. At each iteration, future conflicts are detected as intersections among the planned paths. Such conflicts are solved imposing stops to one or more AGVs. The AGVs to be stopped are defined by means of a greedy algorithm, computing all the possible combinations. Since such computation is performed at each iteration, the computational cost is very large, which makes it difficult to manage the traffic in large factories, as the considered use case. The developed software, based on the architecture proposed in this work, has been installed at the automated factory managing the traffic of the fleet of 8 AGVs.

In order to compare the two systems, a set of critical missions has been chosen and repeated over time. A mission consists in a load and unload task the AGV has to accomplish. Every mission chosen for the experiment has very distant load and unload positions, in order to generate long paths. However, all the load positions are close to each other, in order to generate high traffic scenarios (the same holds for the unload positions). Figure 3.14 shows the number of missions completed over time. A mission is considered completed when the AGV has accomplished both loading and unloading tasks. The performance difference, which was evaluated in terms of the total number of missions completed per hour, is evident: with a set of critical missions, the proposed system was able to complete more than 340 missions in 8 hours of simulation, while the company's system was able to complete approximately 130 missions in the same amount of time. In fact, the company's software, under a critical set of missions, is able to coordinate no more than 6 AGVs, leaving 2 vehicles parked. In order to perform a fair comparison, the proposed system was tested in the same conditions, leaving 2 AGVs



(a) time $t = 0sec$ **(b)** time $t = 13sec$

(c) time $t = 46sec$ **(d)** time $t = 61sec$

Figure 3.15: The figure shows a heavy traffic scenario in which the control software has to deal with the coordination of 4 vehicles in a small area avoiding deadlocks. In fact, this area is modelled as a sector that can simultaneously contain a maximum of 4 vehicles. The time values indicated for each image aims to give an idea of the coordination timing.

parked, and approximately 250 missions were accomplished in the same amount of time.

In order to properly validate the robustness and efficiency of the control software based on the proposed approach we conducted various experiments in the industrial plant. In particular, various traffic scenarios and congestion were faced during the tests. Figure 3.15 shows a heavy traffic scenario in which the control architecture has to deal with the coordination of 4 vehicles in a small area, which has been modelled as a sector that can simultaneously contain a maximum of 4 vehicles. This is a cargo loading and unloading area which usually demand a high number of AGVs. The sequence of images shows the coordination timing and an example of vehicles precedence management. In particular, the AGV 3 and AGV 4 are alternatively waiting for each other in order to allow the neighbouring vehicle to unload and load the cargo. Moreover, the control software was tested to deal with unexpected events that cause the vehicle to stop affecting the traffic and the coordination. These events are typically static or dynamic obstacles, vehicles alarms or errors, and communication errors. When a vehicle is prevented to move due to a static obstacle or a failure (see Figure 3.16), the passage in the respective plant corridor or area is interrupted. The traffic manager and the path planner need to deal with this issue in order to avoid congestions and deadlocks making the fleet to travel an alternative path when possible. Alternatively, a vehicle or a group of vehicles is temporarily stopped due to a dynamic obstacle, such as human workers, communication errors or alarms. In this scenario, the coordination strategy and the traffic manager need to deal with the delay generated by the AGV in fulfilling its task. In fact, this delay affects the AGV motion prediction over time and hence the congestions and deadlock prevention.



Figure 3.16: Several experiments were conducted to stress the control strategy when dealing with static obstacles or vehicles errors. In this scenario a path or location is occupied for a long period by a vehicle. The coordination of the fleet of AGVs needs to consider this interruption in order to avoid congestions and adapt the traffic management consequently.



Figure 3.17: Several experiments were conducted to stress the control strategy when dealing with dynamic obstacles, communication errors or vehicles alarms. In this scenario a vehicle or a group of vehicles is temporarily interrupted in its motion delayed in fulfilling its task. The coordination strategy needs to deal with this delay since it affects the prevention of congestions and deadlocks.

3.8 Discussions

The work describes a novel strategy for the coordination and the traffic management of a fleet of AGVs in an industrial scenario. We propose a multi-layer control architecture in order to improve the efficiency and the flexibility of the global system. The coordination strategy is centralized and based on a hierarchical structure, where a hierarchical path

planning is achieved. The new methodology proposed aims to the realization of a flexible control software implementable on the majority of the industrial layouts. Unlike what is usually found in literature, no assumptions are made on the roadmap and the presence of dynamical obstacles has been considered. A new sector definition has been introduced to better model the traffic evolution on the plant and a new subsector division has been introduced to design a robust traffic-sensitive path planner. Hence, the presented control architecture addresses the typical practical issues such as graph difficulties, a predefined roadmap (constrained by the plant layout), vehicle errors, dynamical obstacles, etc. The coordination strategy does not rely on ad-hoc assumptions or rules and exploits a novel deadlock prevention policy to avoid collisions and congestions.

The validity of the developed software is supported by different simulations and, especially, by tests executed in a real automated factory. The results show the hierarchical planner and traffic manager performs better compared to the company's control software based on common planning algorithms and predefined traffic rules. Moreover, the proposed strategy was able to cope with high traffic situations and unpredictable events, such as dynamic obstacles or alarms, during the experiments conducted in the real environment. No assumption was made on homogeneity of the AGVs: while evaluation has been performed using homogeneous AGVs only, the proposed coordination strategy can be easily adapted to deal with heterogeneous AGVs. This can be achieved, for instance, opportunely tuning the weights of the graph. The sector definition proposed in this study requires a human operator assistance to differentiate the plant areas, such as intersections, corridors, load/unload positions, etc. The proposed approach allows to predict over time the traffic status and vehicle positions. This allows to avoid congestions and deadlocks. However, deadlock situations sporadically appear, due to some pathological circumstances involving a large number of vehicles (during our tests, such cases happened, typically, a few times a week).

Chapter 4

Main Findings and Future Directions

4.1 Main Findings

In this thesis the main focus was to bridge the gap between the control theory of multi-robot systems and real world applications. In particular, we presented the control strategies we developed to coordinate a team of robots for real life applications, such as exploration or monitoring of unknown environments, and for industrial scenarios, as automated warehouses and factories. In Chapter 2 we presented our work to overcome the issues and the non-idealities that arise when deploying a group of robots in a real life scenario. We proposed novel control methodologies, built upon the coverage based control approach, able to coordinate a limited sensing range multi-robot system for optimizing the coverage of non-convex unknown environments or for exploration purposes, and to coordinate a team of agents to estimate an unknown time-varying spatial process in an unknown environment. We widely demonstrated the validity of our methodologies by several rigorous simulations and real world outdoor experiments. In Chapter 3 we focused on the industrial aspects of the multi-robot systems applications. We presented our work that aims to increase the efficiency and productivity of an industrial scenario dealing with the several issues that arise when integrating the developed control architecture in an already existing system.

In particular in Section 2.3 we proposed novel control strategy, based on a limited Voronoi partitioning, that uses only the information directly detected by the robot. We formally show how the control law drives the group of robots to the respective centroid of the limited Voronoi cell. In this work we get rid of some of the typical assumptions made to coordinate a multi-robot system: the proposed methodology does not require a reliable communication network, the robots do not need to known the environment or the obstacle boundaries and they do not need a global knowledge of the surroundings but they exploit only locally detected information. Finally, a series of experiments and simulations were performed on a group of mobile robots to validate the performance of the proposed control method. In fact, we managed to conduct an extensive set of simulations and to implement

real field experiments where the strategy has been proven to be robust and efficient (Section 2.4). It is indeed capable of achieving close-to-optimal performances both in simulation and reality and, as shown, it is also capable of orchestrating a significant number of drones marking the approach as scalable. Real experiments have shown how the presence of non-ideal conditions, such as noisy GPS reading, have an effect on the final configuration but overall do not compromise the behavior of the coverage control function. We note that, according to the literature, this represents the first large-scale deployment of a team of drones performing fully decentralized and distributed coverage control in an outdoor environment. In Section 2.5 we used the proposed coverage control strategy to coordinate a group of autonomous robots with limited sensing capabilities with the aim to explore an unknown environment. The proposed methodology exploits a density distribution to drive the multi-robot system towards areas of the environment to be explored. The density distribution is build from the combination of multiple Gaussian functions placed by the robots in the environment during the motion. We show how the full exploration of the unknown environment can be achieved without a discretization of the environment through graphs or grid cells differently from the majority of the studies found in literature. Moreover, the robots need to store and share among the network only the Gaussian parameters, which require a low communication volume and a low memory consumption. Several experiments and simulations were performed on a group of mobile robots to validate the performance of the proposed control method showing how our strategy outperforms solutions that do not require a discretization or a partial knowledge of the environment, such as random walk and potential field approaches. In Section 2.6 we improved the coverage based control algorithm to coordinate a group of robots with the aim to optimally learn and estimate a spatial field and optimally cover it. This work aims to overcome the issues when dealing with real-world environments that are usually highly variable, unknown and unpredictable. The proposed control strategy, differently from the works found in the literature, efficiently manages the trade-off between the exploration of the environment for the process estimation and the exploitation of this estimation to optimally cover the environment. Moreover, we presented a novel methodology to deal with time varying spatial processes. The multi-robot team adapts its configuration and the exploration-exploitation trade-off optimization depending on the spatial field changes over time. The introduced methodology exploits the Gaussian Processes to learn and generate a model of the spatial field from the data sampled by the team of robots in the environment. We, finally, presented a novel approach to limit the training set to avoid the computational issues introduced by the Gaussian Processes when dealing with large training sets. We tested the performance of the described methodology on several simulations.

Finally in Chapter 3 we presented our studies that aimed to improve the efficiency and productivity of a real industrial scenario. In particular, we presented a novel methodology for the coordination and the traffic management of a fleet of AGVs deployed in an automated factory or warehouse. The proposed multi-layer control architecture has the aim to improve the efficiency, the robustness and the flexibility of the global system. The control strategy consists in two main parts: a hierarchical path planning approach and a coordination strategy able to deal with collisions, deadlocks and vehicles errors.

The methodology proposed aims at the realization of a control software implementable on the majority of the industrial layouts and able to deal with the non idealities of a real implementation. Unlike what is usually found in literature, no assumptions are made on the roadmap and the presence of dynamical obstacles or execution errors has been considered. A novel hierarchical approach based on a sector and subsector division of the plant has been introduced to better model the traffic evolution and to design a traffic sensitive path planner. We propose a coordination strategy which is not based on predefined ad-hoc rules able to manage the vehicles over time avoiding collisions and congestions. Moreover, the coordination methodology exploits a novel approach to recognise and deal with deadlocks based on time expanded graphs. Finally, the whole control architecture is robust to unpredictable events (typical of a real industrial implementation), such as dynamical obstacles, vehicles delays, communication errors. The validity of the developed software is supported by different simulations and, especially, by tests executed in a real automated factory. The results show that the proposed hierarchical planner and traffic manager performs better compared to the company's control software based on common planning algorithms and predefined traffic rules. Moreover, the proposed strategy was able to cope with high traffic situations and unpredictable events, such as dynamic obstacles or alarms, during the experiments conducted in the real environment. No homogeneity assumption was made on the AGVs: while evaluation has been performed using homogeneous AGVs only, the proposed coordination strategy can be easily adapted to deal with heterogeneous AGVs. This can be achieved, for instance, by tuning the weights of the graph.

4.2 Future Works

Several improvements and studies have yet to be done in order to fully take advantage of multi-robot systems in our every-day life and in all the possible real scenarios. About the coverage based control algorithms in Chapter 2 various studies can be deepened in order to improve the proposed methodologies.

For example, analyzing the distributed control strategy proposed in Section 2.3 where only local information is exploited to coordinate the team of robots, future works should aim to further investigate how this strategy can deal with non-convex environments, since promising results have been obtained in a few preliminary simulations. Moreover, future works should aim to improve the control method in order to consider a group of robots characterized by heterogeneous limited sensing capabilities. Several tests and simulations have been conducted to evaluate this control strategy, discussed in Section 2.4. However, future studies can include a more extensive analysis of the real experiments and the investigation of an alternative control law. Indeed, in this work, the final output velocity is generated by the proportional law in Equation (2.2). In some situations, the generated velocity is too high—e.g., when the point to reach is far from the drone's actual position—and does not fulfill the physical constraints of the platform. This represents an intuition for a future investigation: analyzing control strategies that take into account the dynamics of the drone through a more detailed and realistic kinematic and dynamic

model. An evolution of the proposed coverage control strategy has been described in Section 2.5, where we proposed a novel methodology to explore an unknown environment. This control approach has been evaluated with several simulations and real experiments, and its performance has been compared with two other simple solutions able to perform a similar task. However, future works should aim to further investigate the proposed approach in comparison with the more relevant frontier-based exploration methodologies. The only work that has not yet been tested on real robots, is the one described in Section 2.6. Several simulations have been conducted to evaluate the performance of the proposed algorithm and its ability to adapt the coordination of robots to the variability of the environment. Hence, as a future works we aim to conduct experiments on real unmanned ground vehicles and unmanned aerial vehicles. Moreover, we aim to improve the control methodology to consider a multi-robot system characterized by heterogeneous sensing capabilities.

Finally, in Chapter 3 we described a novel hierarchical control architecture to manage the traffic of a fleet of automated guided vehicles in a automated factory or warehouse. The control strategy has been extensively tested and validated through experiments conducted in a real industrial scenario showing the improvement in efficiency and performance. However, the proposed control architecture has a margin of improvement, especially in the definition of the sectors graph. In fact, the sector definition proposed in this work requires a human operator assistance to differentiate the plant areas, such as intersections, corridors, load/unload positions, etc. Future work should aim at developing a reliable method to automatically define the different plant areas and automatically generate the sectors clustering the roadmap nodes. The control strategy presented in this study adopts a decoupled and centralized approach. Several research studies have been conducted to resolve the conflict and deadlock problem of multi-AGV systems using decentralized and distributed control approaches [180, 181]. Thus, there is a need for further investigation of the proposed hierarchical architecture from a decentralized perspective. Specifically, future research should aim to develop a more decentralized strategy and conduct comparative analyses between the proposed approach and other similar distributed approaches found in the literature.

Bibliography

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[2] F. Pratissoli, R. Brugioni, N. Battilani, and L. Sabattini, "Hierarchical traffic management of multi-agv systems with deadlock prevention applied to industrial environments," *second revision round at IEEE Transactions on Automation Science and Engineering*, 2023.

[3] F. Pratissoli, B. Capelli, and L. Sabattini, "On coverage control for limited range multi-robot systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[4] F. Bertoncelli, M. Belal, D. Albani, F. Pratissoli, and L. Sabattini, "On limited-range coverage control for large-scale teams of aerial drones: Deployment and study," in *The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2022.

[5] M. Catellani, F. Pratissoli, F. Bertoncelli, and L. Sabattini, "Coverage control for exploration of unknown non-convex environments with limited range multi-robot systems," in *The 16th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2022.

[6] F. Pratissoli, N. Battilani, C. Fantuzzi, and L. Sabattini, "Hierarchical and flexible traffic management of multi-agv systems applied to industrial environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 009–10 015.

[7] F. Pratissoli, L. Sabattini, and A. Prorok, "Distributed coverage control for time-varying spatial processes," in *submitted to Robotics: Science and Systems (RSS), 2023*, 2023.

[8] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, *et al.*, "Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots," *Journal of Field Robotics*, vol. 30, no. 1, pp. 44–63, 2013.

[9] I. A. Hameed, "A coverage planner for multi-robot systems in agriculture," in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2018, pp. 698–704.

[10] A. Ribeiro and J. Conesa-Muñoz, "Multi-robot systems for precision agriculture," in *Innovation in Agricultural Robotics for Precision Agriculture*. Springer, 2021, pp. 151–175.

[11] S. Jeon, J. Lee, and J. Kim, "Multi-robot task allocation for real-time hospital logistics," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2465–2470.

[12] "Industry trends and market potential – what's next?" https://www.automate.org/industry-insights/industry-trends-and-market-potential-what-s-next, [Online; accessed 20-12-2022].

[13] "Robotics: Latest trends & growth areas in the post pandemic world," https://analyticsindiamag.com/robotics-latest-trends-growth-areas-in-the-post-pandemic-world/, [Online; accessed 20-12-2022].

[14] "Robotics outlook 2030: How intelligence and mobility will shape the future," https://www.bcg.com/publications/2021/how-intelligence-and-mobility-will-shape-the-future-of-the-robotics-industry, [Online; accessed 20-12-2022].

[15] H. Christensen, N. Amato, H. Yanco, M. Mataric, H. Choset, A. Drobnis, K. Goldberg, J. Grizzle, G. Hager, J. Hollerbach, *et al.*, "A roadmap for us robotics–from internet to robotics 2020 edition," *Foundations and Trends® in Robotics*, vol. 8, no. 4, pp. 307–424, 2021.

[16] "Robotics technology market (by component: Hardware, software, service; by robot type: Traditional industrial robots, cobots, professional service robots, others; by application: Manufacturing, healthcare, aerospace & defense, media & entertainment, logistics, others) - global industry analysis, size, share, growth, trends, regional outlook, and forecast 2022-2030," https://www.precedenceresearch.com/robotics-technology-market, [Online; accessed 20-12-2022].

[17] M. Intelligence, *Global Robotics Market*. Mordord Intelligence, 2020, vol. 1.

[18] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[19] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[20] L. E. Parker, "Multi-robot team design for real-world applications," in *Distributed Autonomous Robotic Systems 2*. Springer, 1996, pp. 91–102.

[21] Y. U. Cao, A. B. Kahng, and A. S. Fukunaga, "Cooperative mobile robotics: Antecedents and directions," in *Robot colonies*. Springer, 1997, pp. 7–27.

[22] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[23] V. Puri, A. Nayyar, and L. Raja, "Agriculture drones: A modern breakthrough in precision agriculture," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 507–518, 2017.

[24] I. Maza, F. Caballero, J. Capitán, J. R. Martínez-de Dios, and A. Ollero, "Experimental results in multi-uav coordination for disaster management and civil security applications," *Journal of intelligent & robotic systems*, vol. 61, no. 1, pp. 563–585, 2011.

[25] S. Trigui, A. Koubâa, O. Cheikhrouhou, B. Qureshi, and H. Youssef, "A clustering market-based approach for multi-robot emergency response applications," in *2016 International conference on autonomous robot systems and competitions (ICARSC)*. IEEE, 2016, pp. 137–143.

[26] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[27] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[28] D. Portugal and R. Rocha, "A survey on multi-robot patrolling algorithms," in *Doctoral conference on computing, electrical and industrial systems*. Springer, 2011, pp. 139–146.

[29] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 592–606, 2012.

[30] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[31] J. Cortés, "Coverage optimization and spatial load balancing by robotic sensor networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 749–754, 2010.

[32] S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective uav path planning for search and rescue," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5569–5574.

[33] J. L. Baxter, E. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous Robots and Agents*. Springer, 2007, pp. 9–16.

[34] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 638–639, 2012.

[35] A. Farinelli, E. Zanotto, E. Pagello, *et al.*, "Advanced approaches for multi-robot coordination in logistic scenarios," *Robotics and Autonomous Systems*, vol. 90, pp. 34–44, 2017.

[36] J. J. Roldán-Gómez and A. Barrientos, "Special issue on multi-robot systems: Challenges, trends, and applications," p. 11861, 2021.

[37] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*. IEEE, 2012, pp. 1–5.

[38] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–31, 2019.

[39] M. Garzón, J. Valente, J. J. Roldán, D. Garzón-Ramos, J. d. León, A. Barrientos, and J. d. Cerro, "Using ros in multi-robot systems: Experiences and lessons learned from real-world field tests," in *Robot operating system (ROS)*. Springer, 2017, pp. 449–483.

[40] X. Yun and Y. Yamamoto, "Internal dynamics of a wheeled mobile robot," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, vol. 2. IEEE, 1993, pp. 1288–1294.

[41] K. P. Valavanis and G. J. Vachtsevanos, *Handbook of unmanned aerial vehicles*. Springer, 2015, vol. 1.

[42] C. W. de Silva, "Some issues and applications of multi-robot cooperation," in *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2016, pp. 2–2.

[43] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[44] A. Okabe, "Spatial tessellations," *International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology*, pp. 1–11, 2016.

[45] R. Soukieh, I. Shames, and B. Fidan, "Obstacle avoidance of non-holonomic unicycle robots based on fluid mechanical modeling," in *Proceedings of the European Control Conference*, 2009.

[46] D. Lee, A. Franchi, H. Son, C. Ha, H. Bulthoff, and P. Robuffo Giordano, "Semi-autonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 4, pp. 1334–1345, Aug 2013.

[47] M. Schwager, J. McLurkin, and D. Rus, "Distributed coverage control with sensory feedback for networked robots." in *robotics: science and systems*, 2006, pp. 49–56.

[48] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the lloyd algorithm for computing centroidal voronoi tessellations," *SIAM journal on numerical analysis*, vol. 44, no. 1, pp. 102–119, 2006.

[49] J. Cortes, S. Martinez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 11, no. 4, pp. 691–719, 2005.

[50] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[51] E. Teruel, R. Aragues, and G. López-Nicolás, "A practical method to cover evenly a dynamic region with a swarm," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1359–1366, 2021.

[52] K. Guruprasad and P. Dasgupta, "Distributed voronoi partitioning for multi-robot systems with limited range sensors," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 3546–3552.

[53] B. A. Bash and P. J. Desnoyers, "Exact distributed voronoi cell computation in sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 236–243.

[54] W. Alsalih, K. Islam, Y. N. Rodríguez, and H. Xiao, "Distributed voronoi diagram computation in wireless sensor networks." in *SPAA*, 2008, p. 364.

[55] Q. Li and D. Rus, "Navigation protocols in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 1, no. 1, pp. 3–35, 2005.

[56] F. Aurenhammer, "Power diagrams: properties, algorithms and applications," *SIAM Journal on Computing*, vol. 16, no. 1, pp. 78–96, 1987.

[57] O. Arslan and D. E. Koditschek, "Voronoi-based coverage control of heterogeneous disk-shaped robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4259–4266.

[58] J. Chen and P. Dames, "Distributed multi-target tracking for heterogeneous mobile sensing networks with limited field of views," in *IEEE International Conference Robotics and Automation*, 2021.

[59] Y. Kantaros, M. Thanou, and A. Tzes, "Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints," *Automatica*, vol. 53, pp. 195–207, 2015.

[60] M. Cao and C. Hadjicostis, "Distributed algorithms for voronoi diagrams and application in ad-hoc networks," *UIUC Coordinated Science Laboratory, Tech. Rep. UILU-ENG-03-2222, DC-210*, 2003.

[61] S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, and G. S. Sukhatme, "Multi-robot coordination through dynamic voronoi partitioning for informative adaptive sampling in communication-constrained environments," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2124–2130.

[62] C. He, Z. Feng, and Z. Ren, "Distributed algorithm for voronoi partition of wireless sensor networks with a limited sensing range," *Sensors*, vol. 18, no. 2, p. 446, 2018.

[63] M. Santos and M. Egerstedt, "Coverage control for multi-robot teams with heterogeneous sensing capabilities using limited communications," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5313–5319.

[64] F. Farzadpour, X. Zhang, X. Chen, and T. Zhang, "On performance measurement for a heterogeneous planar field sensor network," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2017, pp. 166–171.

[65] M. Rudolph, S. Wilson, and M. Egerstedt, "Range limited coverage control using air-ground multi-robot teams," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[66] K. Laventall and J. Cortés, "Coverage control by multi-robot networks with limited-range anisotropic sensory," *International Journal of Control*, vol. 82, no. 6, pp. 1113–1121, 2009.

[67] N. Hayashi, K. Segawa, and S. Takai, "2d voronoi coverage control with gaussian density functions by line integration," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 2, pp. 110–116, 2017.

[68] S. Bhattacharya, N. Michael, and V. Kumar, "Distributed coverage and exploration in unknown non-convex environments," in *Distributed autonomous robotic systems*. Springer, 2013, pp. 61–75.

[69] A. Breitenmoser, M. Schwager, J.-C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 4982–4989.

[70] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1, pp. 153–174, 1987.

[71] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[72] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[73] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.

[74] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The mrs uav system: pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 1, pp. 1–28, 2021.

[75] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[76] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[77] M. Kegeleirs, D. Garzón Ramos, and M. Birattari, "Random walk exploration for swarm mapping," in *Towards Autonomous Robotic Systems*, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 211–222.

[78] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Aaai/Iaai*, 2000, pp. 852–858.

[79] K. Senthilkumar and K. K. Bharadwaj, "Multi-robot exploration and terrain coverage in an unknown environment," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 123–132, 2012.

[80] J. De Hoog, S. Cameron, and A. Visser, "Dynamic team hierarchies in communication-limited multi-robot exploration," in *2010 IEEE Safety Security and Rescue Robotics*. IEEE, 2010, pp. 1–7.

[81] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2, pp. 181–196, 2004.

[82] ——, "The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3478–3485.

[83] C. Stachniss, "Coordinated multi-robot exploration," in *Robotic mapping and exploration*. Springer, 2009, pp. 43–71.

[84] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME transactions on mechatronics*, vol. 14, no. 2, pp. 163–175, 2009.

[85] J. Kim, "Cooperative exploration and networking while preserving collision avoidance," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4038–4048, 2016.

[86] O. Arslan and D. E. Koditschek, "Sensor-based reactive navigation in unknown convex sphere worlds," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 196–223, 2019.

[87] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[88] L. Wu, M. Á. García García, D. Puig Valls, and A. Solé Ribalta, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, 2007.

[89] H. Ryu, "Graph search-based exploration method using a frontier-graph structure for mobile robots," *Sensors*, vol. 20, no. 21, p. 6270, 2020.

[90] H. Qin, Z. Meng, W. Meng, X. Chen, H. Sun, F. Lin, and M. H. Ang, "Autonomous exploration and mapping system using heterogeneous uavs and ugvs in gps-denied environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1339–1350, 2019.

[91] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[92] K. Masaba and A. Quattrini Li, "Gvgexp: Communication-constrained multi-robot exploration system based on generalized voronoi graphs," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021.

[93] M. Corah, C. O'Meadhra, K. Goel, and N. Michael, "Communication-efficient planning and mapping for multi-robot exploration in large environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1715–1721, 2019.

[94] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022.

[95] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari, "Automode: A novel approach to the automatic design of control software for robot swarms," *Swarm Intell*, vol. 8, pp. 1–24, 06 2014.

[96] C. Yan, B. Shao, H. Zhao, R. Ning, Y. Zhang, and F. Xu, "3d room layout estimation from a single rgb image," *IEEE Transactions on Multimedia*, vol. 22, no. 11, pp. 3014–3024, 2020.

[97] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*. Springer, 2003, pp. 529–538.

[98] N. Bagheri, "Development of a high-resolution aerial remote-sensing system for precision agriculture," *International journal of remote sensing*, vol. 38, no. 8-10, pp. 2053–2065, 2017.

[99] M. Trincavelli, M. Reggente, S. Coradeschi, A. Loutfi, H. Ishida, and A. J. Lilienthal, "Towards environmental monitoring with mobile robots," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2210–2215.

[100] A. Molchanov, A. Breitenmoser, and G. S. Sukhatme, "Active drifters: Towards a practical multi-robot system for ocean monitoring," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 545–552.

[101] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[102] S. G. Lee, Y. Diaz-Mercado, and M. Egerstedt, "Multirobot control using time-varying density functions," *IEEE Transactions on robotics*, vol. 31, no. 2, pp. 489–493, 2015.

[103] W. Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 2542–2547.

[104] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative planning and online learning with sparse gaussian processes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4292–4298.

[105] M. Lauri, E. Heinänen, and S. Frintrop, "Multi-robot active information gathering with periodic communication," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 851–856.

[106] A. Viseras, D. Shutin, and L. Merino, "Robotic active information gathering for spatial field reconstruction with rapidly-exploring random trees and online learning of gaussian processes," *Sensors*, vol. 19, no. 5, p. 1016, 2019.

[107] C. Xiong, H. Zhou, D. Lu, Z. Zeng, L. Lian, and C. Yu, "Rapidly-exploring adaptive sampling tree*: A sample-based path-planning algorithm for unmanned marine vehicles information gathering in variable ocean environments," *Sensors*, vol. 20, no. 9, p. 2515, 2020.

[108] M. Bresciani, F. Ruscio, S. Tani, G. Peralta, A. Timperi, E. Guerrero-Font, F. Bonin-Font, A. Caiti, and R. Costanzi, "Path planning for underwater information gathering based on genetic algorithms and data stochastic models," *Journal of Marine Science and Engineering*, vol. 9, no. 11, p. 1183, 2021.

[109] E. Guerrero, F. Bonin-Font, and G. Oliver, "Adaptive visual information gathering for autonomous exploration of underwater environments," *IEEE Access*, vol. 9, pp. 136 487–136 506, 2021.

[110] C. Xiong, D. Chen, D. Lu, Z. Zeng, and L. Lian, "Path planning of multiple autonomous marine vehicles for adaptive sampling using voronoi-based ant colony optimization," *Robotics and Autonomous Systems*, vol. 115, pp. 90–103, 2019.

[111] H. Rovina, T. Salam, Y. Kantaros, and M. A. Hsieh, "Asynchronous adaptive sampling and reduced-order modeling of dynamic processes by robot teams via intermittently connected networks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4798–4805.

[112] D. A. Paley and A. Wolek, "Mobile sensor networks and control: Adaptive sampling of spatiotemporal processes," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 91–114, 2020.

[113] M. Santos, U. Madhushani, A. Benevento, and N. E. Leonard, "Multi-robot learning and coverage of unknown spatial fields," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2021, pp. 137–145.

[114] A. Benevento, M. Santos, G. Notarstefano, K. Paynabar, M. Bloch, and M. Egerstedt, "Multi-robot coordination for estimation and coverage of unknown spatial fields," in *2020 ieee international conference on robotics and automation (icra)*. IEEE, 2020, pp. 7740–7746.

[115] W. Luo and K. Sycara, "Adaptive sampling and online learning in multi-robot sensor coverage with mixture of gaussian processes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6359–6364.

[116] W. Luo, C. Nam, G. Kantor, and K. Sycara, "Distributed environmental modeling and adaptive sampling for multi-robot sensor coverage," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 1488–1496.

[117] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, "Decentralized multi-agent exploration with online-learning of gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4222–4229.

[118] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim, "Multi-robot active sensing and environmental model learning with distributed gaussian process," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5905–5912, 2020.

[119] L. Wei, A. McDonald, and V. Srivastava, "Multi-robot gaussian process estimation and coverage: Deterministic sequencing algorithm and regret analysis," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9080–9085.

[120] M. Schwager, M. P. Vitus, D. Rus, and C. J. Tomlin, "Robust adaptive coverage for robotic sensor networks," in *Robotics Research*. Springer, 2017, pp. 437–454.

[121] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[122] C. Park and J. Z. Huang, "Efficient computation of gaussian process regression for large spatial data sets by patching local gaussian processes," *1foldr Import 2019-10-08 Batch 6*, 2016.

[123] K. Jakkala, "Deep gaussian processes: A survey," *arXiv preprint arXiv:2106.12135*, 2021.

[124] M. Liu, G. Chowdhary, B. C. Da Silva, S.-Y. Liu, and J. P. How, "Gaussian processes for learning and control: A tutorial with examples," *IEEE Control Systems Magazine*, vol. 38, no. 5, pp. 53–86, 2018.

[125] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[126] F. Proschan, "Confidence and tolerance intervals for the normal distribution," *Journal of the American Statistical Association*, vol. 48, no. 263, pp. 550–564, 1953.

[127] S. Chakraborti and J. Li, "Confidence interval estimation of a normal percentile," *The American Statistician*, vol. 61, no. 4, pp. 331–336, 2007.

[128] I. Bogunovic, J. Scarlett, and V. Cevher, "Time-varying gaussian process bandit optimization," in *Artificial Intelligence and Statistics*. PMLR, 2016, pp. 314–323.

[129] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1313–1326, 2012.

[130] R. A. Horn, "The hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

[131] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[132] H. Andreasson, A. Bouguerra, M. Cirillo, D. N. Dimitrov, D. Driankov, L. Karlsson, A. J. Lilienthal, F. Pecora, J. P. Saarinen, A. Sherikov, *et al.*, "Autonomous transport vehicles: Where we are and what is missing," *IEEE Robotics & Automation Magazine*, vol. 22, no. 1, pp. 64–75, 2015.

[133] F. Oleari, M. Magnani, D. Ronzoni, and L. Sabattini, "Industrial agvs: Toward a pervasive diffusion in modern factory warehouses," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2014, pp. 233–238.

[134] L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," *Encyclopedia of complexity and system science*, pp. 5783–5800, 2009.

[135] Y. Zhang and H. Mehrjerdi, "A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations," in *2013 International conference on unmanned aircraft systems (ICUAS)*. IEEE, 2013, pp. 1087–1096.

[136] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[137] A. Cámara, D. Silva, P. Henriques Abreu, and E. Oliveira, "Comparing a centralized and decentralized multi-agent approaches to air traffic control," in *Proceedings of the Proceedings of the 28th European Simulation and Modelling Conference, Porto, Portugal*, 10 2014.

[138] T. Siméon, S. Leroy, and J.-P. Lauumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002.

[139] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[140] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[141] R. Olmi, "Traffic management of automated guided vehicles in flexible manufacturing systems," *Ferrara (Italy): University of Ferrara*, 2011.

[142] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević, and S. Bogdan, "Decentralized control of multi-agv systems in autonomous warehousing applications," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1433–1447, 2016.

[143] L. Sabattini, V. Digani, C. Secchi, and C. Fantuzzi, "Hierarchical coordination strategy for multi-agv systems based on dynamic geodesic environment partitioning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, oct. 2016.

[144] C. Wei, K. V. Hindriks, and C. M. Jonker, "Multi-robot cooperative pathfinding: A decentralized approach," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2014, pp. 21–31.

[145] D. Herrero-Perez and H. Martinez-Barbera, "Decentralized coordination of automated guided vehicles," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, 2008, pp. 1195–1198.

[146] P. Yang, R. A. Freeman, and K. M. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, 2008.

[147] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[148] W. Zhang, M. Kamgarpour, D. Sun, and C. J. Tomlin, "A hierarchical flight planning framework for air traffic management," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 179–194, 2011.

[149] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 3. IEEE, 2001, pp. 1213–1219.

[150] R. Luna and K. E. Bekris, "Efficient and complete centralized multi-robot path planning," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3268–3275.

[151] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 2612–2619.

[152] I. F. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, 2006.

[153] H. Martínez-Barberá and D. Herrero-Pérez, "Autonomous navigation of an automated guided vehicle in industrial environments," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 4, pp. 296–311, 2010.

[154] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy," in *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2013, pp. 209–215.

[155] R. Olmi, C. Secchi, and C. Fantuzzi, "Coordination of industrial agvs," *International Journal of Vehicle Autonomous Systems*, vol. 9, no. 1-2, pp. 5–25, 2011.

[156] ——, "Coordination of multiple agvs in an industrial application," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1916–1921.

[157] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[158] J. P. Van Den Berg, D. Nieuwenhuisen, L. Jaillet, and M. H. Overmars, "Creating robust roadmaps for motion planning in changing environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 1053–1059.

[159] R. Geraerts and M. H. Overmars, "A comparative study of probabilistic roadmap planners," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 43–57.

[160] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[161] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[162] A. MANNUCCI, "Intra-logistics with integrated automatic deployment: from one to multi-mobile robot systems," *University of Pisa*, 2020.

[163] A. Mannucci, L. Pallottino, and F. Pecora, "Provably safe multi-robot coordination with unreliable communication," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3232–3239, 2019.

[164] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, "Time-dependent graphs: Definitions, applications, and algorithms," *Data Science and Engineering*, vol. 4, no. 4, pp. 352–366, 2019.

[165] E. G. Coffman, M. Elphick, and A. Shoshani, "System deadlocks," *ACM Computing Surveys (CSUR)*, vol. 3, no. 2, pp. 67–78, 1971.

[166] J. W. Havender, "Avoiding deadlock in multitasking systems," *IBM Systems Journal*, vol. 7, no. 2, pp. 74–84, 1968.

[167] C. Godsil and G. Royle, "Algebraic graph theory," *Graduate text in mathematics, Springer, New York*, 2001.

[168] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[169] S. Bandi and D. Thalmann, "Path finding for human motion in virtual environments," *Computational Geometry*, vol. 15, no. 1-3, pp. 103–127, 2000.

[170] F. Schulz, D. Wagner, and K. Weihe, "Dijkstra's algorithm on-line: An empirical case study from public railroad transport," *Journal of Experimental Algorithmics (JEA)*, vol. 5, pp. 12–es, 2000.

[171] R. V. Bostelman, T. H. Hong, and R. Madhavan, "Towards agv safety and navigation advancement obstacle detection using a tof range camera," in *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE, 2005, pp. 460–467.

[172] M. Boehning, "Improving safety and efficiency of agvs at warehouse black spots," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP).* IEEE, 2014, pp. 245–249.

[173] D. Chen, Z. Shi, P. Yuan, T. Wang, Y. Liu, M. Lin, and Z. Li, "Trajectory tracking control method and experiment of agv," in *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC).* IEEE, 2016, pp. 24–29.

[174] P. S. Pratama, A. V. Gulakari, Y. D. Setiawan, D. H. Kim, H. K. Kim, and S. B. Kim, "Trajectory tracking and fault detection algorithm for automatic guided vehicle based on multiple positioning modules," *International Journal of Control, Automation and Systems*, vol. 14, no. 2, pp. 400–410, 2016.

[175] L. Sabattini, V. Digani, M. Lucchi, C. Secchi, and C. Fantuzzi, "Mission assignment for multi-vehicle systems in industrial environments," in *Proceedings of the IFAC Symposium on Robot Control (SYROCO)*, Salvador, Brazil, aug. 2015.

[176] T. Shimomura and K. Ikeda, "Two types of deadlock detection: cyclic and acyclic," in *Intelligent Systems for Science and Information*. Springer, 2014, pp. 233–259.

[177] A. Hagberg, D. Schult, and P. Swart, "Networkx: Python software for the analysis of networks," *Mathematical Modeling and Analysis, Los Alamos National Laboratory*, 2005.

[178] M. Peasgood, C. M. Clark, and J. McPhee, "A complete and scalable strategy for coordinating multiple robots within roadmaps," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 283–292, 2008.

[179] S. MohaimenianPour, M. Behbooei, and S. S. Ghidary, "Adaptive multiagent path planning with dynamic heuristic," in *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 591–603.

[180] W. Xia, J. Goh, C. A. Cortes, Y. Lu, and X. Xu, "Decentralized coordination of autonomous agvs for flexible factory automation in the context of industry 4.0," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 488–493.

[181] K. Zheng, D. Tang, W. Gu, and M. Dai, "Distributed control of multi-agv system based on regional control model," *Production Engineering*, vol. 7, pp. 433–441, 2013.