MOTION COORDINATION FOR MOBILE ROBOTIC NETWORKS WITH VISIBILITY SENSORS

BY

ANURAG GANGULI

B.Tech, Indian Institute of Technology Bombay, 2002
M.S., University of Illinois at Urbana-Champaign, 2004

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

ABSTRACT

The subject of this dissertation is motion coordination for mobile robotic networks with visibility sensors. Such networks consist of robotic agents equipped with sensors that can measure distances to the environment boundary and to other agents within line of sight. We look at two fundamental coordination problems: (i) deploying over an unknown nonconvex environment to achieve complete visibility, and (ii) gathering all agents initially scattered over the environment at a single location.

As a special case of problem (i), we first address the problem of optimally locating a single robotic agent in a nonconvex environment. The agent is modeled as a point mass with continuous first-order dynamics. We propose a nonsmooth gradient algorithm for the problem of maximizing the area of the region visible to the observer in a non-self-intersecting nonconvex polygon. First, we show that the visible area is almost everywhere a locally Lipschitz function of the observer location. Second, we provide a novel version of the LaSalle Invariance Principle for discontinuous vector fields and for Lyapunov functions with a finite number of discontinuities. Finally, we establish the asymptotic convergence properties of the nonsmooth gradient algorithm and we illustrate numerically its performance.

Second, we address problem (i) by proposing a novel algorithm to the deploy a group of robotic agents in an unknown nonconvex environment to achieve complete visibility. The agents are point masses with discrete-time first-order dynamics. The agents operate asynchronously and two agents can communicate when mutually visible. We also address this deployment problem under the additional constraint that the visibility graph of the final agent locations is connected. We provide distributed algorithms that are guaranteed to solve the two deployment problems if a sufficient number of agents is available. Remarkably, this number is identical to the upper bound established in the famous Art Gallery Problem, i.e., the number of agents sufficient to achieve complete visibility in a known environment through centralized computation. We additionally provide time complexity

bounds for the proposed algorithms.

Third, we address problem (ii) by proposing a novel motion coordination algorithm for a group of robotic agents to achieve rendezvous, that is, to move to a common location inside a nonconvex environment. The robots move synchronously in discrete-time, they have a range-limited visibility sensor, and no communication ability is required. The algorithm is designed using the notions of robust visibility, connectivity-preserving constraint sets, and proximity graphs. We rigorously establish the correctness of the algorithm and we illustrate through simulations the algorithm's performance in asynchronous setups with sensor measurement and control errors.

To my family

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Francesco Bullo, for introducing me to the fascinating world of control theory, robotics and in particular to the beautiful distributed illumination problems. His tremendous support, guidance and encouragement, especially during those lowest moments, have been the driving force behind all this work. Next, I would also like to express my heartfelt gratitude towards Prof. Jorge Cortés for his feedback, help and for the countless hours that we have spent together over challenging proofs.

I would like to thank Prof. Seth Hutchinson for all his support. I would also like to thank all the other members of my doctoral committee, Prof. P R. Kumar, Prof. Mark Spong and Prof. Steven LaValle, for agreeing to serve on it and for their valuable feedback. A special thanks to Prof. Perkins, Prof. Spong, Prof. Franke and Prof. Hutchinson for helping me during the transition from General Engineering to the Electrical and Computer Engineering Department.

I would like to thank my roommates over the years, Sanket Dusad, Girish Varatkar, Jeff Dunbar, Ketan Savla, Shaunak Bopardikar and Deepesh Tated and to my friends Puneet Sharma, Nitin Agarwal, Gaurav Jain, Monica Awasthi, Soham Mazumdar, Nitin Gupta, Ashish Agarwal, Rajeev Jaiman, Amit Bhatia, Manoj Parmar, Amit Pathak, Anubhav Arora and Sorabh Gandhi for all the wonderful times outside of work. I would like to thank my friends at the Coordinated Science Laboratory, Saurabh Tavildar, Siddharth Mallik, Vikas Kawadia, Vivek Raghunathan, Nikhil Chopra and L. N. Rajan for making an already wonderful place a super environment to work in.

I would also like to thank Becky Lonberberger and Francie Bridges at the Coordinated Science Laboratory, Donna Eiskamp at the General Engineering Department and Laurie Fisher and Sherry Beck at the Electrical and Computer Engineering Department for being ever helpful.

In the end I would like to dedicate this thesis to my family for all the things for which I have no words to describe.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Recent years have seen an increasing amount of research in the field of robotic sensor networks [1, 2]. This is due in no small part to the remarkable advances which have been made in recent years in the development of small, agile, relatively inexpensive sensor nodes with mobile and networking capabilities, ultimately intended for a wide variety of purposes such as search and rescue, exploration, environmental monitoring, location-aware computing, and the maintaining of structures. The potential advantages of employing arrays of robotic sensors are numerous. For instance, certain tasks are difficult, if not impossible, when performed by a single agent. Further, a group of agents inherently provides robustness to failures of single agents or communication links.

The existence of such motion-enabled sensing devices and the anticipated development of still more advanced versions raise compelling questions: Can large numbers of such small autonomous devices be successfully deployed as a search team, an exploration group, or something similar, to cooperatively carry out a prescribed task and to respond as a group to high-level management commands?

These future scenarios motivate the study of algorithms for autonomy, adaptation and coordination of robotic sensor networks. So is born the emerging discipline of multi-agent systems. Although technology provides the physical components of such networks and although these systems would have a clear impact on numerous applications, these potential benefits are not yet being realized. As of today, the fundamental limitation is a lack of understanding of how to assemble and coordinate the individual devices into a coherent whole. In other words, there are no systematic methodologies to control large-scale, reliable, distributed systems such as a robotic sensor network performing complex tasks.

In this work, we address some of the above issues for mobile robotic agents equipped with omnidirectional visibility sensors. By visibility sensors, we refer to any device capable of measuring

distances to obstacles and to other agents within line of sight. Two agents can communicate with each other if they are mutually visible to one another, possibly with some time delay. We envision that such networks of mobile agents will be able to perform certain elementary tasks which will play the role of primitive building blocks. Examples of such elementary tasks include deployment and rendezvous. In the context of robotic agents described above, we address the following deployment and rendezvous problems in this dissertation.

Optimal deployment of a single agent We consider a robotic agent in an unknown nonconvex planar environment. The research objective is to design a continuous time, memoryless, feedback algorithm to drive the agent to a position of local optimum of the area visible to it.

Deployment of multiple agents We consider a group of robotic agents in an unknown nonconvex planar environment. The research objective is the design of a distributed algorithm for the following task: entering the environment at a single specified location, the agents move to final positions such that each point of the environment is visible to at least one agent.

Multirobot rendezvous We consider a group of robotic agents scattered in a nonconvex planar environment, possibly at the end of the completion of an earlier task. The objective is to design a provably correct discrete-time memoryless distributed algorithm to drive all the agents to a single location.

We begin by identifying the broad topics to various aspects of the problems described above belong and reviewing the relevant literature. More specific reviews of existing work related to deployment and rendezvous are included in the respective chapters.

Cooperative control

Recent years have witnessed a large research effort focused on motion planning and coordination problems for multi-vehicle systems. Topics include formation control [3, 4, 5, 6, 7], cooperative motion planning [8, 9, 10], cyclic pursuit [11, 12], swarm aggregation [13], and conflict avoidance [14, 15]. It is only recently, however, that truly[1] distributed control laws for dynamic networks are being

[1]Here, by truly distributed, we mean a situation where the communication topology is representative of an ad-hoc wireless or sensor network, and therefore the topology is state-dependent.

proposed. Examples include [16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. Related problems include the design of communication and consensus protocols [26, 27, 28, 29, 30]. The critical role of graph theoretical tools in cooperative control is highlighted in [31, 32]. Short surveys on these problems are given in [25, 33].

Illumination problems and geometric optimization

Illumination and art gallery problems are classic topics, e.g., see [34, 35, 36, 37, 38, 39]. Coverage algorithms (for systems with binary, limited-range sensors) are surveyed in [40]. Next-best-view problems are discussed in [41]. Geometric optimization is a vast and exciting avenue of current research, see for example [42, 43, 44]. Here, by geometric optimization, we mean an optimization problem induced by a collection of geometric objects. For example, in facility location problems service sites are spatially allocated to fulfill a specified request [45, 46]. These approaches mainly rely on centralized computation for a known static environment and are not applicable in a distributed, asynchronous, adaptive setting.

Distributed algorithms

The study of distributed algorithms is concerned with providing mathematical models, devising precise specifications for their behavior, and formally proving their correctness and complexity. Via an automata-theoretic approach, the reference [47] treats distributed consensus, resource allocation, communication, and data consistency problems. Numerical distributed asynchronous algorithms as networking algorithms, rate and flow control, and gradient descent flows are discussed in [48, 49, 50]. All these references do not typically address algorithms over ad-hoc dynamically changing networks. A model of distributed robotic network has been recently proposed in [23, 24].

Behavioral and reactive control

Heuristic approaches to the design of interaction rules and emerging behaviors have been throughly investigated within the literature on behavior-based robotics; see [51, 52, 53, 54, 55, 56, 57, 58, 59]. Along this line of research, few formal results are currently available on how to design reactive control laws, ensure their correctness, and guarantee their optimality or performance with respect

to an aggregate objective.

System theory and optimization methods

An approach to formalizing behavioral control has been pursued using tools from both control theory and formal methods in computer science. Hybrid models of motion control systems are introduced in [60], motion description languages in [61], and hybrid automata are described in [62, 63]. An alternative set of useful tools comes from the "dynamical system approach to algorithms" discussed in [64, 65]. Distributed hybrid and dynamical systems are to be designed as gradient flows of appropriate aggregate cost functions. These ideas can be combined with nonsmooth and convex optimization concepts; see [66, 67].

We now outline the organization of the remainder of the thesis describing in brief the contents of the chapters and our main results therein.

1.1 Outline

As described earlier, in this work we have address the fundamental coordination problems of deployment and rendezvous for networks of robotic agents equipped with visibility sensors. In all of the problems, the agents are assumed to be point masses with first order dynamics and are assumed to operate in an unknown nonconvex environment.

In Chapter 2, we address the problem of optimal deployment of a single agent. In other words, we look at the problem of maximizing the area of the region visible to the agent. Thus, this problem is related to Next Best View approaches to 2D map-building problems in robotics and to visibility and Art Gallery Problems in computational geometry. A heuristic for the Next Best View problem is proposed and simulated in [68] and in the early work [69]. In the context of computational geometry, existing literature deals with visibility optimization when the environment is known. A numerical approach is proposed in [70]. Our work aims to provide a provably correct algorithm to the single agent deployment problem. The distinct contributions in this direction are as follows. First, we rigorously characterize the smoothness properties of the area visible from a point in the environment. Second, we provide a generalized version of the LaSalle Invariance Principle for discontinuous vector fields available in the literature. Third and last, we use the previous novel

results to design a nonsmooth gradient algorithm that monotonically increases the area visible to a point observer. A preliminary version of the above work appeared in [71] and the complete sets of results have appeared in [72].

In Chapter 3, we address the problem of deploying a group of robotic agents over an environment to achieve complete visibility. The agents operate asynchronously and two agents can communicate when mutually visible. This problem is also related to map-building and exploration problems pertaining to unknown 2D environments and planar graphs. Though the above problems have received a lot of attention [73, 74], there have been few works on deployment [75]. Again, this problem can be thought of as a distributed sensor-based version of the classic Art Gallery Problem of computational geometry. Indeed, that is where we draw our biggest motivation from. The famous Art Gallery Theorem states that given a polygon with n vertices, $\lfloor n/3 \rfloor$ agents are always sufficient and occasionally necessary to achieve complete visibility of the environment [76]. However, it is assumed that the environment is known a priori. One of our main contributions is the design of a distributed algorithm that ensures that if all robots are initially collocated, then $\lfloor n/3 \rfloor$ agents are sufficient and occasionally necessary to complete the task. In other words, we obtain the same bounds on the number of agents required. We also consider the deployment problem under the additional constraint that the visibility graph of the final agent locations is connected. It has been shown [77, 78] that $(n-2)/2$ agents are sufficient and occasionally necessary to solve the second problem. Here also, it is assumed that the environment is known. We propose a second distributed algorithms that are guaranteed to solve the above problems if $(n-1)/2$ of agents are available. We, thus, obtain a bound that differs from the bound in the centralized case by at most one. We additionally provide time complexity bounds for the proposed algorithms. Preliminary versions of this work have appeared in [79, 80].

In Chapter 4, we present a motion coordination algorithm for a group of robotic agents to achieve rendezvous, that is, to move to a common location inside the environment. In this case, we also assume that the agents have access to sensory information alone and their ability to sense distances to other objects is present only up to a certain distance. The rendezvous problem has been studied for agents with limited range sensing in obstacle free unbounded environments [81]. The current formulation of the problem is, however, completely novel. A synchronous algorithm is

designed that is guaranteed to work if the range-limited visibility graph of the agents is connected at any time during the execution of the algorithm. Simulations illustrate the theoretical results on the correctness of the proposed algorithm, and its performance in asynchronous setups and with sensor measurement and control errors. A preliminary version of the above work appeared in [82] and the complete sets of results have appeared in [83].

Finally, in Chapter 5 we conclude by summarizing the contributions of this work and discussing some interesting directions of future research.

CHAPTER 2

Optimal deployment of a single robotic agent

2.1 Introduction

Consider a single-point mobile robot in a planar nonconvex environment modeled as a non-self-intersecting polygon: how should the robot move in order to monotonically increase the area of its visible region (i.e., the region within its line of sight)? This problem is the subject of this chapter. The following are the modeling assumptions in our method of approach. The dynamical model for the robot's motion is a first-order system of the form $\dot{p} = u$, where p refers to the position of the robot in the environment and u is the driving input. The robot is equipped with an omnidirectional line-of-sight range sensor; the range of the sensor is larger than the diameter of the environment. The robot does not know the entire environment nor its position in it, and its instantaneous motion depends only on what is within line of sight (this assumption restricts our attention to memoryless feedback laws).

In broad terms, this work is related to numerous references on optimal sensor location and motion planning coming from the computational geometry, geometric optimization, and robotics literature. The problem we consider is akin to the Next Best View problems in robotics for 2D map building. In these map-building problems the objective is to compute the next position of a robot in an environment with obstacles that maximizes the gain in visible area. A heuristic is proposed and simulated in [68] and in the early work [69]. Other relevant references can be found in computational geometry. For example, the classic Art Gallery Problem is to find the smallest number of such guards necessary for each point of the environment to be visible to at least one guard; see [42, 84]. Also studied in computational geometry is the problem of locating a guard in a non-self-intersecting polygon so as to maximize the visible area. This problem is still open to the best of our knowledge and is the subject of ongoing research; see [85, 86, 87]. Key differences exists

between the computational geometric approach to this problem and our sensor-based feedback approach. In the computational geometric version, the data about the entire polygon is available a priori, the difficulties are of combinatorial nature, and the solutions can be thought of as open loop. In the problem of interest in this chapter, we consider the feedback control problem for a mobile robot based upon only local knowledge of the environment and without recollection of past trajectories. The work that is perhaps closest in spirit to our work is the numerical approach proposed in [70].

A second set of relevant references are those on nonsmooth stability analysis. Indeed, our approach to maximizing visible area is to design a nonsmooth gradient flow. To define our proposed algorithm we rely on the notions of generalized gradient [66] and of Filippov solutions for differential inclusions [88]. To study our proposed algorithm we extend recent results on the stability and convergence properties of nonsmooth dynamical systems, as presented in [89, 90]. Finally, the present work has some interesting connections with the study of the behavior of certain territorial animals. A particularly relevant reference is the study of the effect of visibility on space use by red-capped cardinals [91]. These are birds that defend territories along shorelines of rivers and lakes and tend to spend the majority of their time near peninsulas (areas that offer greater amount of visibility of their respective territories) rather than bays.

The contributions in this chapter are threefold. First, we prove some basic properties of the area visible from a point observer in a nonconvex polygon Q, see Figure 2.1. Namely, we show that the area of the visibility polygon, as a function of the observer position, is a locally Lipschitz function almost everywhere, and that the finite point set of discontinuities consists of the reflex vertices of the polygon Q. Additionally, we compute the generalized gradient of the function and show that the function is not, in general, regular.

Second, we provide a generalized version of certain stability theorems for discontinuous vector fields available in the literature [89, 90]. Specifically, we provide a generalized nonsmooth LaSalle Invariance Principle for discontinuous vector fields, Filippov solutions, and Lyapunov functions that are locally Lipschitz almost everywhere (except for a finite set of discontinuities).

Third and last, we use these novel results to design a nonsmooth gradient algorithm that monotonically increases the area visible to a point observer. To the best of our knowledge, this is

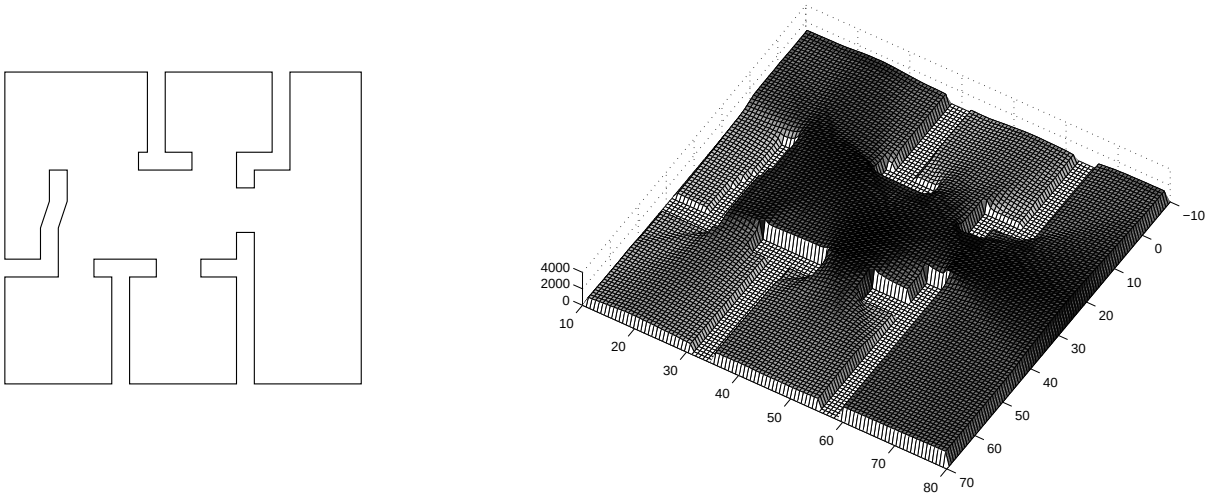


Figure 2.1: The figure on the left shows a nonconvex polygonal environment shaped like a typical floor-plan. The figure on the right shows the variation of the visible area in the environment as a function of the position of a point observer.

the first provably correct algorithm for this version of the Next Best View problem. We illustrate the performance of our algorithm via simulations for some interesting polygons.

Before proceeding with the technical content, we provide here a detailed comparison between our proposed local feedback method and a two-phase approach, where the guard first explores the entire environment and later finds the approximate location of the maximum. The approximate global maximum could be computed using, for example, the method in [87]. Let us refer to the latter approach as the *explore and optimize method*. Clearly in this latter method, the optimal location of the observer is a global maximum whereas in the former it is a local maximum. However, our approach does not require any memory for the observer. In the explore and optimize method, the observer needs to remember the environment as it is being explored. As the size of the environment increases, so does the amount of memory and run time required by the observer. Another problem that arises in the exploration of unknown environments in the absence of accurate global positioning is that of *simultaneous location and mapping (SLAM)*. To explain briefly, to build a map of the environment accurately, one needs an accurate estimate of the position of the observer. This is not available due to odometry errors and lack of global positioning ability. Therefore, to accurately localize the observer inside the environment, an accurate map of the environment is needed but which is again unavailable due to measurement sensor errors. Various approaches to solving the

SLAM problem have been studied but most of the accurate approaches again are intensive in terms of computation and memory. In contrast, the fact that our approach is local and requires no memory renders it more robust to errors in computation and sensor measurements. Another advantage of being memoryless is that it works for environments that may change quasi-statically where as any other method relying on memory is ineffective in this case. Finally, note that out local optimization approach might be used in conjunction with a global search method to find the globally optimum position. For example, after finding the position of a local optimum, the observer could perform a random walk in the environment. While executing the random walk, the observer must compare the magnitude of the visible area with the value at the previously discovered optimum. If the value is greater, find the local optimum using the approach in this chapter. This procedure is repeated and one can expect to find the global optimum after sufficient time has elapsed.

The rest of the chapter is organized as follows. Section 2.2 contains the analysis of the smoothness properties and of the generalized gradient of the function of interest. Section 2.3 contains the novel results on nonsmooth stability analysis. Section 2.4 presents the nonsmooth gradient algorithm and the properties of the resulting closed-loop system. Finally, the simulations in Section 2.5 illustrate the convergence properties of the algorithm.

2.2 The area visible from an observer

In this section we study the area of the region visible to a point observer equipped with an omnidirectional camera. We show that the visible area, as a function of the location of the observer, is locally Lipschitz, except at a finite point set. We prove that, for general nonconvex polygons, the function is not regular. We also provide expressions for the generalized gradient of the visible area function wherever it is locally Lipschitz. We have included the notions of locally Lipschitz functions and related concepts whenever they first appear in this chapter.

Let us start by introducing the set of lines on the plane $\mathbb{R}^2$. For $(a,b,c) \in \mathbb{R}^3 \backslash \{(0,0,c) \in \mathbb{R}^3 \mid c \in \mathbb{R}\}$, define the equivalence class $[(a,b,c)]$ by

$$[(a,b,c)] = \left\{(a',b',c') \in \mathbb{R}^3 \mid (a,b,c) = \lambda(a',b',c'),\ \lambda \in \mathbb{R}\right\}.$$

The *set of lines on* $\mathbb{R}^2$ is defined as

$$\mathbb{L} = \left\{ [(a,b,c)] \subset \mathbb{R}^3 \mid (a,b,c) \in \mathbb{R}^3,\ a^2 + b^2 \neq 0 \right\}.$$

It is possible to show that $\mathbb{L}$ is a 2-dimensional manifold, sometimes referred to as the affine Grassmannian of lines in $\mathbb{R}^2$; see [92].

Next, two useful functions are defined. Let $f_{\text{pl}} : \mathbb{R}^2 \times \mathbb{R}^2 \setminus \{(p,p) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid p \in \mathbb{R}^2\} \to \mathbb{L}$ map two distinct points in $\mathbb{R}^2$ to the line passing through them. Given distinct $(x_1, y_1), (x_2, y_2) \in \mathbb{R}^2$, we have that

$$f_{\text{pl}}\left((x_1,y_1),(x_2,y_2)\right) = [(y_2 - y_1, x_1 - x_2, y_1 x_2 - x_1 y_2)].$$

If $l_1 \parallel l_2$ denotes that two lines $l_1, l_2 \in \mathbb{L}$ are parallel, then let $f_{\text{lp}} : \mathbb{L}^2 \setminus \{(l_1,l_2) \in \mathbb{L}^2 \mid l_1 \parallel l_2\} \to \mathbb{R}^2$ map two non-parallel lines to their unique intersection point. Given two lines $[(a_1,b_1,c_1)]$ and $[(a_2,b_2,c_2)]$, we have that

$$f_{\text{lp}}\left([(a_1,b_1,c_1)],[(a_2,b_2,c_2)]\right) = \left(\frac{b_2 c_1 - b_1 c_2}{a_2 b_1 - a_1 b_2}, \frac{a_1 c_2 - a_2 c_1}{a_2 b_1 - a_1 b_2} \right).$$

Note that the maps f_{pl} and f_{lp} are class C^ω, i.e., they are analytic over their domains.

Now, let us turn our attention to the polygonal environment. Let Q be a non-self-intersecting polygon, possibly nonconvex. A polygon is said to be non-self-intersecting if the only points in the plane belonging to two polygon edges are the polygon vertices. Such a polygon has a well-defined interior and exterior. Note that a non-self-intersecting polygon can contain holes. Let $\mathring{Q}$ and ∂Q denote the interior and the boundary of Q, respectively. Let $\text{Ve}(Q) = (v_1, \ldots, v_n)$ be the list of vertices of Q ordered counterclockwise. The *interior angle of a vertex* v of Q is the angle formed inside Q by the two edges of the boundary of Q incident at v. The point $v \in \text{Ve}(Q)$ is a *reflex vertex* if its interior angle is strictly greater than π. Let $\text{Ve}_r(Q)$ be the list of reflex vertices of Q. If S is a finite set, then let $|S|$ denote its cardinality.

A point $q \in Q$ is *visible from* $p \in Q$ if the segment between q and p is contained in Q. The *visibility polygon* $S(p) \subset Q$ from a point $p \in Q$ is the set of points in Q visible from p. It is convenient to think of $p \mapsto S(p)$ as a map from Q to the set of polygons contained in Q. It must be noted that the visibility polygon is not necessarily a non-self-intersecting polygon.

Definition 2.2.1. *Let v be a reflex vertex of Q, and let $w \in \mathrm{Ve}(Q)$ be visible from v. The* (v, w)-generalized inflection segment $I(v, w)$ *is the set*

$$I(v, w) = \{q \in S(v) \mid q = \lambda v + (1 - \lambda)w, \lambda \geq 1\}.$$

A reflex vertex v of Q is an anchor of $p \in Q$ *if it is visible from p and if $\{q \in S(v) \mid q = \lambda v + (1 - \lambda)p, \lambda > 1\}$ is not empty.*

In other words, a reflex vertex is an anchor of p if it occludes a portion of the environment from p. Note that in the robotics path-planning literatre, when v and w are both reflex vertices and $I(v, w)$ is tangent to the boundary of Q at both v and w, the inflection segment is also known as a bitangent; see [93]. Figure 2.2 illustrates the various notions defined above. Given a point q and a line l, let $\mathrm{dist}(q, l)$ denote the distance between them.

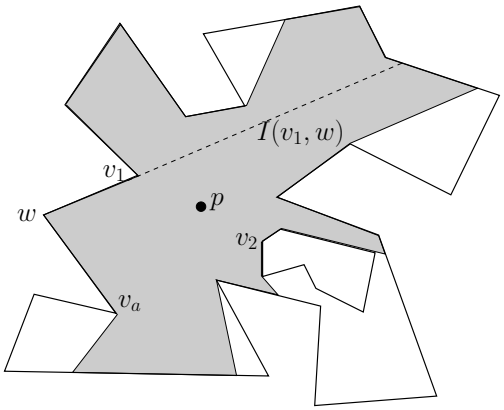


Figure 2.2: Reflex vertices v_1 and v_2, a generalized inflection segment $I(v_1, w)$, an anchor v_a of p and the visibility polygon (shaded region) from p. Note that the polygonal environment has a hole.

Note that any generalized inflection segment $I(v, w)$ splits the polygonal environment into two smaller polygons. The vertex w is visible from any point on $I(v, w)$ and from the interior of only *one* of the two smaller polygons. Intuitively, it then follows that if p belongs to the interior of a polygon and does not lie on a generalized inflection segment, then in a neighborhood of p the number of vertices of the visibility polygon does not change and their positions vary smoothly as a function of p. This is described formally in the following theorem.

Theorem 2.2.2. *Let $\{I_\alpha\}_{\alpha \in \mathcal{A}}$ be the set of generalized inflection segments of Q, and let P be*

a connected component of $Q \setminus \bigcup_{\alpha\in\mathcal{A}} I_\alpha$. *For all* $p \in P$, *the visibility polygon* $S(p)$ *is non-self-intersecting and has a constant number of vertices, say* $\mathrm{Ve}(S(p)) = \{u_1(p), \ldots, u_k(p)\}$. *For all* $i \in \{1, \ldots, k\}$, *the map* $P \ni p \mapsto u_i(p)$ *is* C^ω *and*

$$\mathrm{d}u_i(p) = \begin{cases} 0, & u_i(p) \in \mathrm{Ve}(Q), \\ \dfrac{\mathrm{dist}(v_a, l)}{(\mathrm{dist}(p, l) - \mathrm{dist}(v_a, l))^2\sqrt{a^2+b^2}} \begin{bmatrix} -b \\ a \end{bmatrix} \begin{bmatrix} y - y_a \\ x_a - x \end{bmatrix}^T, & u_i(p) = f_{\mathrm{lp}}(f_{\mathrm{pl}}(v_a, p), l), \end{cases}$$

where $v_a = (x_a, y_a)$ *is an anchor of* p *and where* $l = [(a, b, c)]$ *is a line defined by an edge of* Q.

Proof. The first part of the proof is by contradiction. Let $|\,\mathrm{Ve}(S(p'))| > |\,\mathrm{Ve}(S(p))|$ for some point $p' \in P$. This means that at least one additional vertex is visible from p' that was occluded by an anchor of p. Two cases may arise. First, when the additional vertex belongs to $\mathrm{Ve}(Q)$, then by our definition, p and p' must lie on opposite sides of a generalized inflection segment. This is a contradiction. Secondly, if the additional vertex does not belong to $\mathrm{Ve}(Q)$, it must be the projection of a reflex vertex (acting as an anchor). Here again two cases may arise: (1) the reflex vertex is visible from p, and (2) it is not. The first case is possible only if the reflex vertex is visible but does not act as an anchor. So, positive lengths of both sides adjoining the reflex vertex must also be visible from p and at least one of the sides is completely not visible from p' since there is a projection. This means that p and p' lie on opposite sides of a generalized inflection segment generated by the reflex vertex and one of its adjacent vertices. This is a contradiction. The second case is possible if the reflex vertex in question is occluded by another reflex vertex. But this means that p and p' lie on opposite sides of the generalized inflection segment from the reflex vertex to the anchor occluding the reflex vertex; again this is a contradiction. If, on the other hand, $|\,\mathrm{Ve}(S(p'))| < |\,\mathrm{Ve}(S(p))|$, then the above arguments hold by interchanging p and p'. Hence, p and p' lie on opposite sides of a generalized inflection segment which is a contradiction. This completes the proof that $|\,\mathrm{Ve}(S(p'))|$ is constant for all $p' \in P$.

Let $p \in P$. Since the visibility polygon $S(p)$ is star-shaped and since any ray emanating from p can intersect Q at most at two distinct points, then $S(p)$ is non-self-intersecting. (Indeed, if the ray emanating from p intersects the environment at three or more points inside $S(p)$, then p must

belong to a generalized inflection segment. See Figure 2.3)

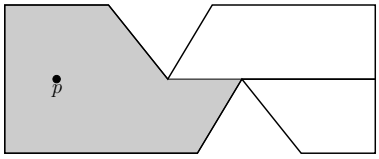


Figure 2.3: The visibility polygon of the point represented by p. Note that there exists a ray emanating from p which intersects the environment at three points and hence the corresponding visibility polygon is self-intersecting.

Regarding the second statement, it is clear that if $u_i(p)$ is a vertex of Q then it is independent of p. Instead, if $u_i(p) \notin \mathrm{Ve}(Q)$, then

$$u_i(p) = f_{\mathrm{lp}}(f_{\mathrm{pl}}((x, y), (x_a, y_a)), \ell)$$

where $p = (x, y)$, $v_a = (x_a, y_a)$ is an anchor of p, and ℓ is the line, determined by an edge of Q, that identifies u_i. Now, $p \in P$ implies $p \neq v_a$. It follows that $f_{\mathrm{pl}}(p, v_a)$ is C^ω for all $p \in P$. Also, from the definition of $u_i(p)$, it is clear that $f_{\mathrm{pl}}(p, v_a) \nparallel \ell$. Therefore, for all $p \in P$, $f_{\mathrm{lp}}(f_{\mathrm{pl}}(p, v_a), \ell)$ is C^ω; this implies that $p \mapsto u_i(p)$ is also C^ω. The formula for the derivative can be verified directly. $\square$

Next, the area of a visibility polygon as a function of the observer location is studied, see Figure 2.1. Recall that the area of a non-self-intersecting polygon Q with counterclockwise-ordered vertices $\mathrm{Ve}(Q) = ((x_1, y_1), \ldots, (x_n, y_n))$ is given by

$$A(Q) = \frac{1}{2} \sum_{i=1}^{n} x_i(y_{i-1} - y_{i+1}),$$

where $(x_0, y_0) = (x_n, y_n)$ and $(x_{n+1}, y_{n+1}) = (x_1, y_1)$. As in the previous theorem, let $\{I_\alpha\}_{\alpha \in \mathcal{A}}$ be the set of generalized inflection segments of Q and let P be a connected component of $Q \setminus \bigcup_{\alpha \in \mathcal{A}} I_\alpha$. Next, if $p \in P$, the visibility polygon from p has a constant number of vertices, say $k = |\mathrm{Ve}(S(p))|$, is non-self-intersecting, and satisfies $A \circ S(p) = \sum_{i=1}^{k} x_i(y_{i-1} - y_{i+1})$ where $\mathrm{Ve}(S(p)) = (u_1, \ldots, u_k)$ are ordered counterclockwise, $u_i(p) = (x_i, y_i)$, $u_0 = u_k$, and $u_{k+1} = u_1$. Therefore, $P \ni p \mapsto A \circ S(p)$

is also C^ω and

$$\mathrm{d}(A \circ S)(p) = \sum_{i=1}^{k} \frac{\partial A(u_1, \ldots, u_k)}{\partial u_i} \mathrm{d}u_i(p). \tag{2.1}$$

Remark 2.2.3. For any $u_i(p) \notin \mathrm{Ve}(Q)$, we have

$$\frac{\partial (A \circ S)}{\partial u_i} \mathrm{d}u_i(p) = \frac{\mathrm{dist}(v_a, l)}{2} \frac{\mathrm{dist}(u_{i+1}, l) - \mathrm{dist}(u_{i-1}, l)}{(\mathrm{dist}(p, l) - \mathrm{dist}(v_a, l))^2} \begin{bmatrix} y - y_a \\ x_a - x \end{bmatrix}^T. \tag{2.2}$$

Note here that $\frac{\partial (A \circ S)}{\partial u_i} \mathrm{d}u_i(p)$ is perpendicular to $p - v_a$.

To illustrate (2.1) and (2.2), it is convenient to introduce the *versor* operator defined by $\mathrm{vers}(X) = X/\|X\|$ if $X \in \mathbb{R}^2 \setminus \{0\}$ and by $\mathrm{vers}(0) = 0$. We depict the normalized gradient $\mathrm{vers}(\mathrm{d}(A \circ S))$ of the visible area function in Figure 2.4.

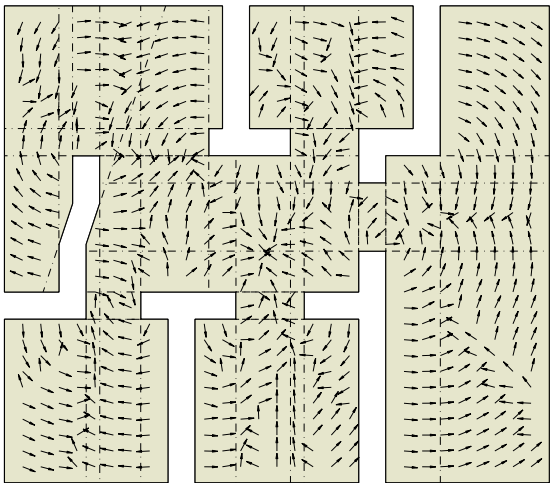


Figure 2.4: Normalized gradient of the visible area function over the nonconvex polygon depicted in Figure 2.1. The dashed lines represent some of the generalized inflection segments.

We will now characterize the smoothness properties of the map $A \circ S$ over a polygon Q excluding the set of reflex vertices. Before that, we present the following notion.

Definition 2.2.4. *A function $f : \mathbb{R}^N \to \mathbb{R}$ is said to be* locally Lipschitz near $x \in \mathbb{R}^N$ *if there exist positive constants L_x and ϵ such that $|f(y) - f(y')| \leq L_x \|y - y'\|$ for all $y, y' \in B(x, \epsilon)$, where $B(x, \epsilon)$ is a N dimensional open ball of radius ϵ and centered at x.*

Note that continuously differentiable functions at x are locally Lipschitz near x.

Theorem 2.2.5. *The map $A \circ S$ restricted to $Q \setminus \mathrm{Ve}_r(Q)$ is locally Lipschitz.*

Proof. By Theorem 2.2.2, it suffices to consider points lying on generalized inflection segments. Let p belong to multiple, say m, generalized inflection segments $\{I_\alpha\}_{\alpha\in\{1,\ldots,m\}}$. Let ϵ be small enough such that no generalized inflection segments intersect $B(p,\epsilon)$ other than $\{I_\alpha\}_{\alpha\in\{1,\ldots,m\}}$. For $\alpha \in \{1,\ldots,m\}$, let v_{k_α} be the anchor determining the generalized inflection segment I_α. Without loss of generality, it can be assumed that no anchor is visible from p other than $v_{k_1},\ldots,v_{k_m}$. For $\alpha \in \{1,\ldots,m\}$, lines $l_\alpha \perp f_{\mathrm{pl}}(p, v_{k_\alpha})$ can be constructed with the property that $l_\alpha \cap Q = \emptyset$ and the vector $v_{k_\alpha} - p$ points toward l_α. Let, h_α be the line parallel to l_α, tangent to $B(\epsilon, p)$, and intersecting the segment from p to v_{k_α}. Let p' and p'' belong to $B(p,\epsilon) \cap (Q \setminus \mathrm{Ve}_r(Q))$. Next, let $q'_\alpha = f_{\mathrm{lp}}(f_{\mathrm{pl}}(p', v_{k_\alpha}), l_\alpha)$ and $q''_\alpha = f_{\mathrm{lp}}(f_{\mathrm{pl}}(p'', v_{k_\alpha}), l_\alpha)$; see Figure 2.5. Let v'_α and v''_α be the

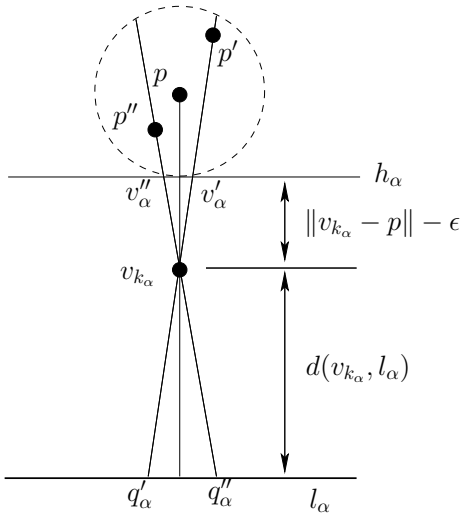


Figure 2.5: Definition of the lines l_α, h_α, and the points $q'_\alpha, q''_\alpha, v'_\alpha, v''_\alpha$.

intersections between h_α and the lines $f_{\mathrm{pl}}(p', v_{k_\alpha})$ and $f_{\mathrm{pl}}(p'', v_{k_\alpha})$, respectively.

Now, $|A(v_{k_\alpha}, q'_\alpha, q''_\alpha)| = \frac{1}{2}\|q'_\alpha - q''_\alpha\| \operatorname{dist}(v_{k_\alpha}, l_\alpha)$. But from Figure 2.5, it is easy to see that $\|q'_\alpha - q''_\alpha\| = \frac{\operatorname{dist}(v_{k_\alpha}, l_\alpha)}{\|v_{k_\alpha} - p\| - \epsilon}\|v'_\alpha - v''_\alpha\|$ and that $\|v'_\alpha - v''_\alpha\| < \|p' - p''\|$. For $K_\alpha(p) = \frac{1}{2}\frac{\operatorname{dist}(v_{k_\alpha}, l_\alpha)^2}{\|v_{k_\alpha} - p\| - \epsilon}$, the

following is true:

$$\begin{aligned}|A(S(p')) - A(S(p''))| &\leq \sum_{\alpha=1}^{m} |A(v_{k_\alpha}, q'_\alpha, q''_\alpha)| \\ &\leq \sum_{\alpha=1}^{m} K_\alpha(p)\|p' - p''\|.\end{aligned}$$

This fact is illustrated by Figure 2.6. This completes the proof that $Q \setminus \mathrm{Ve}_r(Q) \ni p \mapsto A \circ S(p)$ is

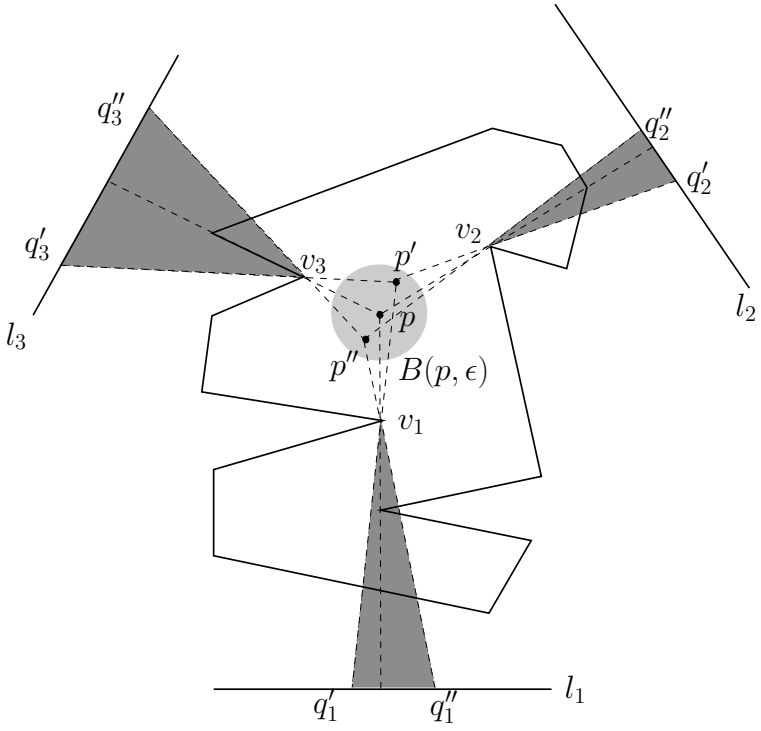


Figure 2.6: Upper bounds on the change in area. Here $m = 3$.

locally Lipschitz. □

It is clear that the map $A \circ S$ is not differentiable everywhere. However other notions of derivatives might still be defined for it. The usual *right directional derivative* and the *generalized directional derivative* of any function f at x in the direction of $v \in \mathbb{R}^N$ are defined, respectively, as

$$f'(x;v) = \lim_{t \to 0^+} \frac{f(x+tv) - f(x)}{t}, \qquad f^o(x;v) = \limsup_{\substack{y \to x \\ t \to 0^+}} \frac{f(y+tv) - f(y)}{t}.$$

For a locally Lipschitz function, the limit in the definition of $f'(x;v)$ does not always exist, whereas the limit in $f^o(x;v)$ is always well-defined. Also, from Rademacher's Theorem [66], we know that locally Lipschitz functions are continuously differentiable almost everywhere (in the sense of Lebesgue measure). If Ω_f denotes the set of points in $\mathbb{R}^N$ at which f fails to be differentiable, and S denotes any other set of measure zero, the *generalized gradient* of f is defined by

$$\partial f(x) = \operatorname{co}\left\{ \lim_{i \to +\infty} \mathrm{d}f(x_i) \mid x_i \to x\,,\ x_i \notin S \cup \Omega_f \right\}.$$

Note that this definition coincides with $\mathrm{d}f(x)$ if f is continuously differentiable at x. The generalized gradient and the generalized directional derivative (cf. Proposition 2.1.2 in [66]) are related by $f^o(x;v) = \max\{\zeta \cdot v \mid \zeta \in \partial f(x)\}$, for each $v \in \mathbb{R}^N$.

To obtain the expression for the generalized gradient of $A \circ S$, the polygon Q is partitioned as follows.

Lemma 2.2.6. *Let $\{I_\alpha\}_{\alpha \in \mathcal{A}}$ be the set of generalized inflection segments of Q. There exists a unique partition $\{\overline{P}_\beta\}_{\beta \in \mathcal{B}}$ of Q where P_β is a connected component of $Q \setminus \bigcup_{\alpha \in A} I_\alpha$ and $\overline{P}_\beta$ denotes its closure.*

Figure 2.7 illustrates this partition for the given nonconvex polygon. Note that this partition is similar to the View Space Partition in the computer vision and robotics literature; see [94]. For $\beta \in \mathcal{B}$, define $A_\beta : \overline{P}_\beta \to \mathbb{R}_+$ by

$$A_\beta(p) = A \circ S(p), \quad \text{for } p \in P_\beta,$$

and by continuity on the boundary of P_β. It turns out that the maps A_β, $\beta \in \mathcal{B}$, are continuously differentiable[1] on $\overline{P}_\beta$. Equation (2.1) gives the value of the gradient for $p \in P_\beta$. However, in general, for $p \in \overline{P}_{\beta_1} \bigcap \ldots \bigcap \overline{P}_{\beta_m} \setminus \mathrm{Ve}_r(Q)$, based on Theorem 2.2.5 and Lemma 2.2.6, we can write

$$\partial(A \circ S)(p) = \operatorname{co}\Big\{ \mathrm{d}A_{\beta_1}(p), \ldots, \mathrm{d}A_{\beta_m}(p) \Big\}. \tag{2.3}$$

[1] A function is continuously differentiable on a closed set if (1) it is continuously differentiable on the interior, and (2) the limit of the derivative at a point in the boundary does not depend on the direction from which the point is approached.

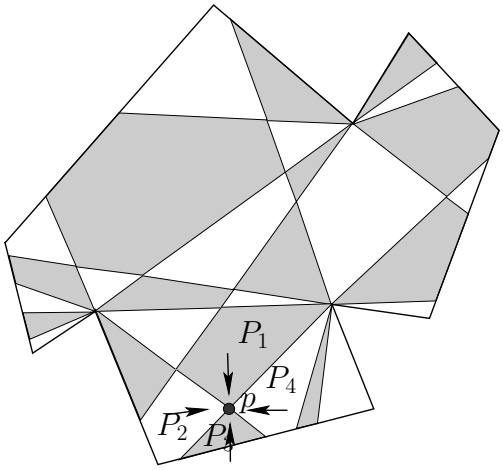


Figure 2.7: Partition of Q. The generalized gradient of the area function at p is the convex hull of the gradient of four functions $A_1, \ldots, A_4$ at p.

This completes our study of the generalized gradient of the locally Lipschitz function $A \circ S$. Apart from differentiability, another smoothness property of $A \circ S$ that we will characterize is the one of regularity. As will be clear later, this will be crucial in constructing a set-valued estimate of the rate at which $A \circ S$ changes as the observer moves. We define regularity first and then show how $A \circ S$ is not regular in many interesting situations.

Definition 2.2.7. *A function $f : \mathbb{R}^N \to \mathbb{R}$ is said to be* regular at $x \in \mathbb{R}^N$ *if for all $v \in \mathbb{R}^N$, $f'(x; v)$ exists and $f^o(x; v) = f'(x; v)$.*

Again, a continuously differentiable function at x is regular at x. Also, a locally Lipschitz function at x which is convex is also regular (cf. Proposition 2.3.6 in [66]).

Lemma 2.2.8. *There exists a nonconvex polygon Q such that the maps $A \circ S$ and $-A \circ S$ restricted to $Q \setminus \mathrm{Ve}_r(Q)$ are not regular.*

Proof. We present an example to justify the above statement. In Figure 2.8, $\partial(A \circ S)(p') = \mathrm{co}\{\mathrm{d}A_1, \mathrm{d}A_2\}$ where $\|\mathrm{d}A_1\| \gg \|\mathrm{d}A_2\|$. Take a vector η' perpendicular to the generalized inflection segment to which p' belongs (see Figure 2.8). It is clear that $(A \circ S)'(p; \eta') = \mathrm{d}A_2 \cdot \eta'$. However, $(A \circ S)^0(p'; \eta') = \max\{\zeta \cdot \eta' | \zeta \in \partial(A \circ S)(p')\} = \mathrm{d}A_1 \cdot \eta' > \mathrm{d}A_2 \cdot \eta'$. Again, in Figure 2.8, $\partial(-A \circ S)(p'') = \mathrm{co}\{-\mathrm{d}A_3, -\mathrm{d}A_4\}$, where $\| - \mathrm{d}A_4\| \gg \| - \mathrm{d}A_3\|$. Take a vector η'' perpendicular to the generalized inflection segment to which p'' belongs (see Figure 2.8). It is clear

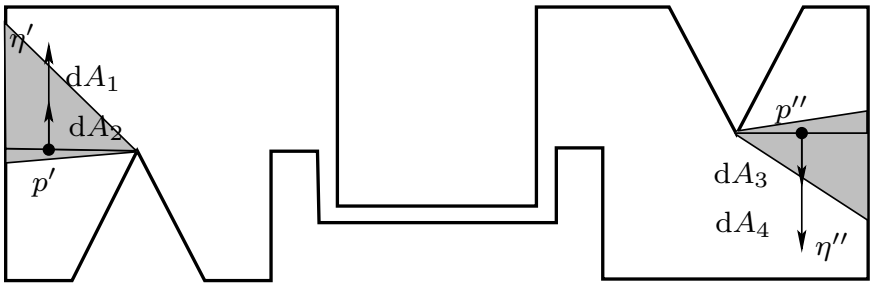


Figure 2.8: Example polygon for which $A \circ S$ and $-A \circ S$ restricted to $Q \setminus \mathrm{Ve}_r(Q)$ are not regular. Note here that $\mathrm{d}A_1$ and $\mathrm{d}A_2$ are not perfectly aligned with η'. Also, $\mathrm{d}A_3$ and $\mathrm{d}A_4$ are not perfectly aligned with η''.

that $-(A \circ S)'(p''; \eta'') = -\mathrm{d}A_4 \cdot \eta''$. However, $(A \circ S)^0(p''; \eta'') = \max\{\zeta \cdot \eta'' | \zeta \in \partial(A \circ S)(p'')\} = -\mathrm{d}A_3 \cdot \eta'' > -\mathrm{d}A_4 \cdot \eta''$. □

2.3 An invariance principle in nonsmooth stability analysis

This section presents results on stability analysis for discontinuous vector fields via nonsmooth Lyapunov functions. The results extend the work in [90] and will be useful in the next control design section, see also [95]. We refer the reader to [88] and to Appendix A for some useful nonsmooth analysis concepts that we have not included in the main body of this chapter.

In what follows we shall study differential equations of the form

$$\dot{x}(t) = X(x(t)), \tag{2.4}$$

where $X : \mathbb{R}^N \to \mathbb{R}^N$ is a measurable and essentially locally bounded, possibly discontinuous vector field. We understand the solution of this equation in the Filippov sense following [88]. For each $x \in \mathbb{R}^N$, consider the set

$$K[X](x) = \bigcap_{\delta > 0} \bigcap_{\mu(S)=0} \mathrm{co}\{X(B(x, \delta) \setminus S)\},$$

where μ denotes the usual Lebesgue measure in $\mathbb{R}^N$. Alternatively, one can show [96] that there

exists a set S_X of measure zero such that

$$K[X](x) = \operatorname{co}\left\{ \lim_{i\to+\infty} X(x_i) \mid x_i \to x\,,\ x_i \notin S \cup S_X \right\}\,,$$

where S is any set of measure zero. A Filippov solution (see A) of (2.4) on an interval $[t_0, t_1] \subset \mathbb{R}$ is defined as a solution of the differential inclusion

$$\dot{x} \in K[X](x)\,. \tag{2.5}$$

Since the set-valued map $K[X] : \mathbb{R}^N \to 2^{\mathbb{R}^N}$ is upper semicontinuous with nonempty, compact, convex values and locally bounded (cf. [88]), the existence of Filippov solutions of (2.4) is guaranteed by Lemma A.0.3. A set M is *weakly invariant* (respectively *strongly invariant*) for (2.4) if for each $x_0 \in M$, M contains a maximal solution (respectively all maximal solutions) of (2.4).

We now introduce another useful tool. Given a locally Lipschitz function $f : \mathbb{R}^N \to \mathbb{R}$, the *set-valued Lie derivative of f with respect to X* at x is defined as

$$\widetilde{\mathcal{L}}_X f(x) = \{a \in \mathbb{R} \mid \exists v \in K[X](x) \text{ such that } \zeta \cdot v = a\,,\ \forall \zeta \in \partial f(x)\}\,.$$

For each $x \in \mathbb{R}^N$, $\widetilde{\mathcal{L}}_X f(x)$ is a closed and bounded interval in $\mathbb{R}$, possibly empty. If f is continuously differentiable at x, then $\widetilde{\mathcal{L}}_X f(x) = \{\mathrm{d}f \cdot v \mid v \in K[X](x)\}$. If, in addition, X is continuous at x, then $\widetilde{\mathcal{L}}_X f(x)$ corresponds to the singleton $\{\mathcal{L}_X f(x)\}$, the usual Lie derivative of f in the direction of X at x.

We are now ready to state the first result in this section.

Lemma 2.3.1. *Let $X : \mathbb{R}^N \to \mathbb{R}^N$ be measurable and essentially locally bounded and let $f : \mathbb{R}^N \to \mathbb{R}$ be locally Lipschitz. Let $\gamma : [t_0, t_1] \to \mathbb{R}^N$ be a Filippov solution of X such that $f(\gamma(t))$ is regular for almost all $t \in [t_0, t_1]$. Then*

(i) $\frac{\mathrm{d}}{\mathrm{d}t}(f(\gamma(t)))$ *exists for almost all $t \in [t_0, t_1]$, and*

(ii) $\frac{\mathrm{d}}{\mathrm{d}t}(f(\gamma(t))) \in \widetilde{\mathcal{L}}_X f(\gamma(t))$ *for almost all $t \in [t_0, t_1]$.*

Proof. The result is an immediate consequence of Lemma 1 in [90]. □

The following result is a generalization of the classic LaSalle Invariance Principle for smooth vector fields and smooth Lyapunov functions to the setting of discontinuous vector fields and nonsmooth Lyapunov functions.

Theorem 2.3.2 (LaSalle Invariance Principle)**.** *Let $X : \mathbb{R}^N \to \mathbb{R}^N$ be measurable and essentially locally bounded and let $S \subset \mathbb{R}^N$ be compact and strongly invariant for X. Let $C \subset S$ consist of a finite number of points and let $f : S \to \mathbb{R}$ be locally Lipschitz on $S \setminus C$ and bounded from below on S. Assume the following properties hold:*

(A1) if $x \in S \setminus C$, then either $\max \widetilde{\mathcal{L}}_X f(x) \leq 0$ or $\widetilde{\mathcal{L}}_X f(x) = \emptyset$,

(A2) if $x \in C$ and if γ is a Filippov solution of X with $\gamma(0) = x$, then $\lim_{t\to 0^-} f(\gamma(t)) \geq \lim_{t\to 0^+} f(\gamma(t))$, and

(A3) if $\gamma : \overline{\mathbb{R}}_+ \to S$ is a Filippov solution of X, then $f \circ \gamma$ is regular almost everywhere.

Define $Z_{X,f} = \left\{x \in S \setminus C \mid 0 \in \widetilde{\mathcal{L}}_X f(x)\right\}$ and let M be the largest weakly invariant set contained in $(\overline{Z}_{X,f} \cup C)$. Then the following statements hold:

(i) if $\gamma : \overline{\mathbb{R}}_+ \to S$ is a Filippov solution of X, then $f \circ \gamma$ is monotonically nonincreasing;

(ii) each Filippov solution of X with initial condition in S approaches M as $t \to +\infty$;

(iii) if M consists of a finite number of points, then each Filippov solution of X with initial condition in S converges to a point of M as $t \to +\infty$.

Proof. Fact (i) is a consequence of Assumptions (A1), (A2) and (A3), and of Lemma 2.3.1.

In what follows we shall require the following notion. Given a curve $\gamma : \mathbb{R}_+ \to \mathbb{R}^N$, the positive limit set of γ, denoted by $\Omega(\gamma)$, is the set of $y \in \mathbb{R}^N$ for which there exists a sequence $\{t_k\}_{k\in\mathbb{N}} \subset \mathbb{R}$ such that $t_k < t_{k+1}$, for $k \in \mathbb{N}$, $\lim_{k\to+\infty} t_k = +\infty$, and $\lim_{k\to+\infty} \gamma(t_k) = y$. For $x \in S$, let γ_1 be a Filippov solution of X with $\gamma_1(0) = x$ and let $\Omega(\gamma_1)$ be the limit set of γ_1. Under this setting, $\Omega(\gamma_1)$ is nonempty, bounded, connected and weakly invariant, see [88]. Furthermore, $\Omega(\gamma_1) \subset S$ because S is strongly invariant and closed.

To prove fact (ii), it suffices to show that $\Omega(\gamma_1) \subset \overline{Z}_{X,f} \cup C$. Trivially, $\Omega(\gamma_1) \cap C \subset C$. Let $y \in \Omega(\gamma_1) \setminus C$ so that f is locally Lipschitz at y. There exists a sequence $\{t_k\}_{k\in\mathbb{N}}$ such

that $\lim_{k\to+\infty}\gamma_1(t_k) = y$. Because $f\circ\gamma_1$ is monotonically nonincreasing and f is bounded from below, $\lim_{t\to+\infty} f(\gamma_1(t))$ exists and is equal to, say, $a \in \mathbb{R}$. Now, by continuity of f, $a = \lim_{k\to+\infty} f\circ\gamma_1(t_k) = f(y)$. This proves that $f(y) = a$ for all $y \in \Omega(\gamma_1)\setminus C$. At this point we distinguish two cases. First, assume that y is an isolated point in $\Omega(\gamma_1)$. Then clearly, there exists a Filippov solution of X, say γ_2, such that $\gamma_2(t) = y$ for all $t \geq 0$. Hence $\frac{\mathrm{d}}{\mathrm{d}t} f(\gamma_2(t)) = 0$, and, by Lemma 2.3.1, $0 \in \widetilde{\mathcal{L}}_X f(\gamma_2(t))$ or in other words $y \in Z_{X,f}$. Second, assume that y is not isolated in $\Omega(\gamma_1)$, and let γ_2 be a Filippov solution of X with $\gamma_2(0) = y$. Since f is continuous at y and $\Omega(\gamma_1)$ contains a finite number of points of discontinuity of f, there exists $\delta > 0$ such that $f(y') = a$ for all $y' \in B(y,\delta)\cap\Omega(\gamma_1)$. Therefore, there exists $t' > 0$ such that $f(\gamma_2(t)) = a$ for all $t \in [0,t']$. Hence, we have $\frac{\mathrm{d}}{\mathrm{d}t} f(\gamma_2(t)) = 0$ for all $t \in [0,t']$. It follows from Lemma 2.3.1 that for all $t \in [0,t']$, we have $0 \in \widetilde{\mathcal{L}}_X f(\gamma_2(t))$ or in other words $\gamma_2(t) \in Z_{X,f}$. By continuity of γ_2 at $t = 0$, we have that $\gamma_2(0) = y \in \overline{Z}_{X,f}$. Since $\Omega(\gamma_1)$ is weakly invariant, we have $\Omega(\gamma_1) \subset M$ and hence γ_2 approaches M.

We now prove fact (iii). If M consists of a finite number of points, and since $\Omega(\gamma_1) \subset M$ is connected, $\Omega(\gamma_1)$ is a point. Hence, by the argument in the preceding paragraph, each Filippov solution of X approaches a point of M. In other words, it converges to a point of M. □

Corollary 2.3.3. *The LaSalle Invariance Principle is valid under the following relaxed assumption:*

(A3) if $\gamma : \overline{\mathbb{R}}_+ \to S$ is a Filippov solution of X, then almost everywhere either $f\circ\gamma$ or $-f\circ\gamma$ is regular.

Proof. The proof is a consequence of the fact that $\frac{\mathrm{d}}{\mathrm{d}t}(f(\gamma(t)))$ exists and belongs to $\widetilde{\mathcal{L}}_X f(\gamma(t))$ if and only if $\frac{\mathrm{d}}{\mathrm{d}t}(-f(\gamma(t)))$ exists and belongs to $\widetilde{\mathcal{L}}_X(-f)(\gamma(t))$. Thus result (ii) of Lemma 2.3.1 still holds and the proof of the LaSalle Invariance Principle remains unchanged. □

2.4 Maximizing the area visible from a mobile observer

In this section we build on the analysis results obtained thus far to design an algorithm that maximizes the area visible to a mobile observer. We aim to reach local maxima of the visible area $A\circ S$ by designing some appropriate form of a gradient flow for the discontinuous function $A\circ S$.

We now present an *introductory and incomplete* version of the algorithm: the objective is to steer the mobile observer along a path for which the visible area is guaranteed to be nondecreasing.

Name:	Increase visible area for Q
Goal:	Maximize the area visible to a mobile observer
Assumption:	Generalized inflection segments of Q do not intersect. Initial position does not belong to a generalized inflection segment.

Let $p(t)$ denote the observer position at time t inside the nonconvex polygon Q. The observer performs the following tasks at each time instant:

compute visibility polygon $S(p(t)) \subset Q$,

if $p(t)$ does not belong to any generalized inflection segment or to the boundary of Q **then**
move along the versor of the gradient $\mathrm{d}(A \circ S)$

else if $p(t)$ belongs to a generalized inflection segment but not to the boundary of Q **then**
depending on the generalized gradient $\partial(A \circ S)$, either slide along the segment or leave the segment in an appropriate direction

else if $p(t)$ belongs to the boundary of Q but not to a reflex vertex, **then**
depending on the projection of $\partial(A \circ S)$ along the boundary, either slide along the boundary or move in an appropriate direction toward the interior of Q

else
either follow a direction of ascent of $A \circ S$ or stop

end if

The remainder of this section is dedicated to formalizing this loose description.

2.4.1 A modified gradient vector field

Before describing the algorithm to maximize the area visible to the mobile observer, we introduce the following useful notions. Given a non-self-intersecting polygon Q with $\mathrm{Ve}(Q) = (v_1, \ldots, v_n)$ and $\epsilon > 0$, define the following quantities:

(i) let the *ϵ-expansion of Q* be $Q^\epsilon = \{p \mid ||p - q|| \leq \epsilon \text{ for some } q \in Q\}$,

(ii) for $i \in \{1, \ldots, n\}$, let P_i^ϵ be the open set delimited by the edge $\overline{v_i v_{i+1}}$, the bisectors of the external angles at v_i and v_{i+1} and the boundary of Q^ϵ,

(iii) for ϵ small enough and for any point p in Q^ϵ, let $\mathrm{prj}_Q(p)$ be uniquely equal to $\arg\min\{||p' - p|| \mid p' \in \partial Q\}$, and

(iv) for $p \in \cup_{i\in\{1,\ldots,n\}} P_i^\epsilon$, let the *outward normal* $n(\mathrm{prj}_Q(p))$ be the unit vector directed from $\mathrm{prj}_Q(p)$ to p.

We illustrate these notions in Figure 2.9. Note that $\mathrm{prj}_Q(p)$ can never be a reflex vertex. We

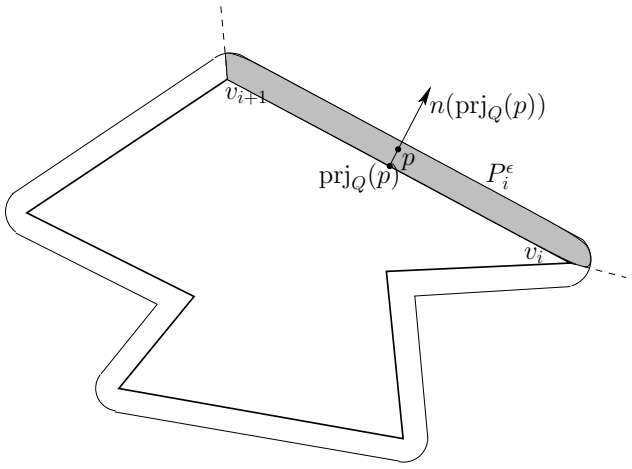


Figure 2.9: The ϵ-expansion Q^ϵ of the non-self-intersecting polygon Q, an open set P_i^ϵ and the corresponding outward normal $n(\mathrm{prj}_Q(p))$.

can now define a vector field on Q^ϵ as follows:

$$X_Q(p) = \begin{cases} \mathrm{vers}(\mathrm{d}(A \circ S)(p)), & \text{if } p \in \mathring{Q} \setminus \{I_\alpha\}_{\alpha\in\mathcal{A}}, \\ -n(\mathrm{prj}_Q(p)), & \text{if } p \in P_i^\epsilon, \\ 0, & \text{otherwise.} \end{cases}$$

(Recall that the versor operator is defined by $\mathrm{vers}(Y) = Y/\|Y\|$ if $Y \in \mathbb{R}^2 \setminus \{0\}$ and by $\mathrm{vers}(0) = 0$.) Note that X_Q is well-defined because at $p \in \mathring{Q} \setminus \{I_\alpha\}_{\alpha\in\mathcal{A}}$ the function $A \circ S$ is analytic. Clearly, X_Q is not continuous on Q^ϵ. However, the set of points where it is discontinuous is of measure zero. Almost everywhere in the interior of Q, the vector field X_Q is equal to the normalized gradient of

$A \circ S$ as depicted in Figure 2.4.

Remark 2.4.1. An important observation in this setting is that at all points p where $A \circ S$ is locally Lipschitz, we have $K[\mathrm{d}(A \circ S)](p) = \partial(A \circ S)(p)$. In such a case it is also true that for all $\eta \in \partial(A \circ S)(p)$, there exists at least one $\delta > 0$ such that $\delta\eta \in K[X_Q](p)$ and vice versa.

We now present the differential equation describing the motion of the observer:

$$\dot{p}(t) = X_Q(p(t)). \tag{2.6}$$

A Filippov solution of (2.6) on an interval $[t_0, t_1] \subset \mathbb{R}$ is defined as a solution of the differential inclusion

$$\dot{p}(t) \in K[X_Q](p(t)), \tag{2.7}$$

where $K[X_Q]$ is the usual Filippov differential inclusion associated with X_Q, see Appendix A. Since X_Q is measurable and bounded, the existence of a Filippov solution is guaranteed. We study uniqueness and completeness of Filippov solutions in the following lemma.

Lemma 2.4.2. *The following statements hold true:*

(i) there exists a non-self-intersecting polygon Q for which the corresponding vector field X_Q admits multiple Filippov solutions starting from the same initial condition;

(ii) any non-self-intersecting polygon Q is a strongly invariant set for the corresponding vector field X_Q and, therefore, any Filippov solution is defined over $\overline{\mathbb{R}}_+$.

Proof. We present an example to justify the statement (i). In Figure 2.10, at the point p_0 on the generalized inflection segment, both directions η_1 and η_2 belong to $\partial(A \circ S)(p_0)$. Three distinct Filippov solutions of equation (2.6) exist. Two of the solutions start from p_0 along the two directions η_1 and η_2 while the third solution is $p(t) = p_0$ for all $t \geq 0$. Statement (ii) is a consequence of the

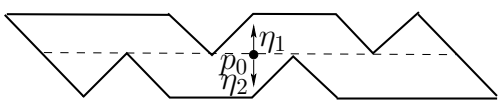


Figure 2.10: Three Filippov solutions exist starting from the point p_0.

definition of X_Q on P_i^ϵ for $i \in \{1, \ldots, n\}$. □

We now claim that any solution of the differential inclusion (2.7) has the property that the visible area increases monotonically. To prove these desirable properties, we first present the following results in nonsmooth analysis.

2.4.2 Properties of solutions and convergence analysis

To prove the convergence properties of the solution of (2.7) using the results presented in Section 2.3, we must first define a suitable Lyapunov function. Intuitively since our objective is to maximize the visible area, our Lyapunov function should be closely related to it. For $\epsilon > 0$, we now define the *extended area function* A_Q^ϵ at all points $p \in Q \bigcup \{\cup_i P_i^\epsilon\}$. The extended function coincides with the original function on the interior and on the boundary of Q and is defined appropriately outside:

$$A_Q^\epsilon(p) = \begin{cases} A \circ S(p), & \text{if } p \in Q, \\ A \circ S(\text{prj}_Q(p)) - ||p - \text{prj}_Q(p)||, & \text{if } p \in \cup_i P_i^\epsilon. \end{cases}$$

For all $p \in \partial Q \setminus \text{Ve}\, Q$, A_Q^ϵ satisfies (see Figure 2.11):

$$A_Q^\epsilon{}'(p; n(\text{prj}_Q(p))) = -1.$$

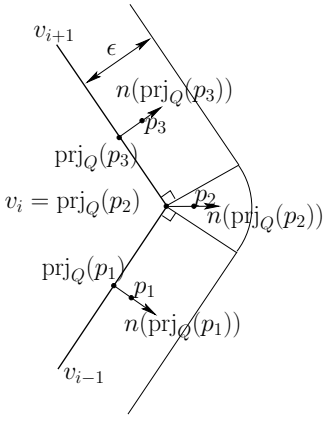


Figure 2.11: Extending the function $A \circ S$ to A_Q^ϵ. Note the direction of $n(\text{prj}_Q(p_i))$ at all points p_i.

Remark 2.4.3. The extended area function A^ϵ_Q is locally Lipschitz on $(Q \setminus \mathrm{Ve}_r(Q)) \bigcup \{\cup_i P^\epsilon_i\}$ and analytic almost everywhere on $Q \bigcup \{\cup_i P^\epsilon_i\}$.

The following theorem characterizes the regularity of the map $p \mapsto -A^\epsilon_Q(p)$ along a Filippov solution of X_Q. This is important to prove that the area of the visibility polygon is nondecreasing along any Filippov solution of X_Q

Theorem 2.4.4. *Let $G(Q)$ be the subset of Q where both maps $p \mapsto -A^\epsilon_Q(p)$ and $p \mapsto A^\epsilon_Q(p)$ are not regular. Then any Filippov solution $\gamma : \overline{\mathbb{R}}_+ \to Q$ of X_Q has the property that $\gamma(t) \notin G(Q)$ for almost all $t \in \overline{\mathbb{R}}_+$ unless γ reaches a critical point of $K[X_Q]$.*

Proof. Note that $G(Q)$ is a subset of $\cup_{\alpha \in \mathcal{A}} I_\alpha$. This is a consequence of Theorem 2.2.2 and the fact that functions are regular at points of differentiability. Given a generalized inflection segment I_α, let l_α be the line extending I_α and let t_α be one of the two unit tangent vectors to I_α. A Filippov solution γ of X_Q slides along I_α starting from $p_0 \in I_\alpha$ only if $\partial A^\epsilon_Q(p_0)$ contains either t_α or $-t_\alpha$. It then suffices to show that if $\partial A^\epsilon_Q(p_0)$ contains t_α or $-t_\alpha$, then either A^ϵ_Q or $-A^\epsilon_Q$ is regular at p_0. Let us also assume that p_0 does not belong to any other generalized inflection segment. If this were not the case, then either p_0 is a critical point or the Filippov solution does not belong to the point of intersection for almost all $t \in \overline{\mathbb{R}}_+$.

Let l_α divide $\mathbb{R}^2$ into two open half planes H_1 and H_2. There exists $\delta > 0$ such that A^ϵ_Q is analytic on $H_i \cap B(p_0, \delta)$, $i \in \{1, 2\}$, see Figure 2.12. On l_α, we have $(A^\epsilon_Q)_1 = (A^\epsilon_Q)_2$ where $(A^\epsilon_Q)_i$ is

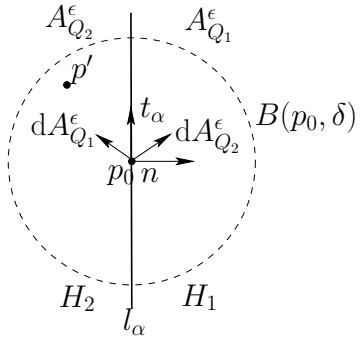


Figure 2.12: The point p_0 lies on the generalized inflection segment l. H_1 and H_2 are half planes on either side of l. n and t are normal and parallel to l respectively. The other arrows indicate the directions of $\mathrm{d}A^\epsilon{}_Q(p)$ on either side of l.

the function A^ϵ_Q restricted to H_i. Let $p' \in B(p_0, \delta)$ and, without loss of generality, let $p' \in H_2$. Let

n be the normal to I_α at p_0 pointing away from p'. Note that in terms of the notation introduced in Section 2.4.1, $n = -n(\mathrm{prj}_{l_\alpha}(p'))$ where $\mathrm{prj}_{l_\alpha}(p') = \arg\min\{||p'-p|| \mid p \in l_\alpha\}$. Now, $(A_Q^\epsilon)_1$ can be extended to $H_2 \cap B(p_0, \delta)$ by analyticity. Likewise, $(A_Q^\epsilon)_2$ can be extended to $H_1 \cap B(p_0, \delta)$. Since the functions $(A_Q^\epsilon)_i$, $i \in \{1, 2\}$, are analytic, they can be written as the expansions of their Taylor series:

$$(A_Q^\epsilon)_i(p') = (A_Q^\epsilon)_i(p_0) + \mathrm{d}((A_Q^\epsilon)_i)(p_0) \cdot (p' - p_0) + O(\|p' - p_0\|^2).$$

It follows from the above set of equations that:

$$(A_Q^\epsilon)_2(p') - (A_Q^\epsilon)_1(p') = \big(\mathrm{d}((A_Q^\epsilon)_2) - \mathrm{d}((A_Q^\epsilon)_1)\big) \cdot (p' - p_0) + O(\|p' - p_0\|^2).$$

Note that n is the same for all $p' \in H_2$. Now, $p' - p_0 = -c_1 n + c_2 t_\alpha$ such that $c_1 \geq 0$. Also, $\mathrm{d}(A_Q^\epsilon)_1(p_0) \cdot t_\alpha = \mathrm{d}(A_Q^\epsilon)_2(p_0) \cdot t_\alpha$ since $(A_Q^\epsilon)_1(p) = (A_Q^\epsilon)_2(p)$ for $p \in I_\alpha$. Therefore,

$$(A_Q^\epsilon)_2(p') - (A_Q^\epsilon)_1(p') = c_1 \left(\mathrm{d}(A_Q^\epsilon)_1(p_0) \cdot n - \mathrm{d}(A_Q^\epsilon)_2(p_0) \cdot n\right) + O(\|p' - p_0\|^2).$$

Now, either t_α or $-t_\alpha$ belongs to $\partial A_Q^\epsilon(p_0) = \mathrm{co}\{\mathrm{d}(A_Q^\epsilon)_1, \mathrm{d}(A_Q^\epsilon)_2\}$ if and only if the product of $\mathrm{d}(A_Q^\epsilon)_1(p_0) \cdot n$ and $\mathrm{d}(A_Q^\epsilon)_2(p_0) \cdot n$ is less than or equal to zero (see Figure 2.12). If $\mathrm{d}(A_Q^\epsilon)_1(p_0) \cdot n = 0$ and $\mathrm{d}(A_Q^\epsilon)_2(p_0) \cdot n = 0$, then clearly A_Q^ϵ is C^1 at p_0 and hence regular. Otherwise, let us assume, without loss of generality, that $\mathrm{d}(A_Q^\epsilon)_1(p_0) \cdot n - \mathrm{d}(A_Q^\epsilon)_2(p_0) \cdot n < 0$. Therefore, there exists $\eta_2 > 0$ such that $(A_Q^\epsilon)_2(p') - (A_Q^\epsilon)_1(p') \leq 0$ for $p' \in H_2 \cap B(p_0, \eta_2)$. Similarly, there exists $\eta_1 > 0$ such that for $p' \in H_1 \cap B(p_0, \eta_1)$, we have $(A_Q^\epsilon)_1(p') - (A_Q^\epsilon)_2(p') \leq 0$. Thus, there exists a neighborhood around p_0 where $A_Q^\epsilon(p) = \min\{(A_Q^\epsilon)_1(p), (A_Q^\epsilon)_2(p)\}$ or $-A_Q^\epsilon(p) = \max\{-(A_Q^\epsilon)_1(p), -(A_Q^\epsilon)_2(p)\}$. Since $(A_Q^\epsilon)_i$, $i \in \{1, 2\}$, are smooth functions, it follows from Proposition 2.3.12 in [66] that $-A_Q^\epsilon$ is regular at p_0. On the other hand, if we assume that $\mathrm{d}(A_Q^\epsilon)_1(p_0) \cdot n - \mathrm{d}(A_Q^\epsilon)_2(p_0) \cdot n > 0$, then we get that A_Q^ϵ is regular at p_0. □

In the following theorem, the functions A_Q^ϵ and $-A_Q^\epsilon$ are used as candidate Lyapunov functions to show the convergence properties of Filippov solutions of X_Q.

Theorem 2.4.5. *Any Filippov solution $\gamma : \overline{\mathbb{R}}_+ \to Q$ of X_Q has the following properties:*

(i) $t \mapsto A \circ S(\gamma(t))$ is continuous and monotonically nondecreasing,

(ii) γ approaches the set of critical points of $K[X_Q]$.

Proof. Let us start by showing that, if γ is a Filippov solution of X_Q, then $A \circ S \circ \gamma$ is continuous. The reader is referred to Figure 2.13 for an introduction of notations used. Let X_Q^r and X_Q^θ be the components of X_Q parallel and perpendicular to $p - v$ respectively. Similarly, let $\mathrm{d}(A \circ S(p))^r$ and $\mathrm{d}(A \circ S(p))^\theta$ be the components of $\mathrm{d}(A \circ S(p))$ parallel and perpendicular to $p - v$ respectively. Note that if $\|\mathrm{d}(A \circ S(p))\| \neq 0$, then $\|X_Q^r\| = \frac{\|\mathrm{d}(A \circ S(p))^r\|}{(\|\mathrm{d}(A \circ S(p))^r\|^2 + \|\mathrm{d}(A \circ S(p))^\theta\|^2)^{1/2}}$ and $\|X_Q^\theta\| = \frac{\|\mathrm{d}(A \circ S(p))^\theta\|}{(\|\mathrm{d}(A \circ S(p))^r\|^2 + \|\mathrm{d}(A \circ S(p))^\theta\|^2)^{1/2}}$. Let $\epsilon > 0$ be such that $\|\mathrm{d}(A \circ S(p))\| \neq 0$ for all $p \in B(v, \epsilon) \cap D$. For now, let us also assume that $\{\cup_{\alpha \in \mathcal{A}} I_\alpha\} \cap B(v, \epsilon) \cap D = \emptyset$. We now claim that in $B(v, \epsilon) \cap D$, $\mathrm{d}(A \circ S(p))^\theta = \Omega(1/\|p - v\|)$ and $\mathrm{d}(A \circ S(p))^r = O(1)$. In other words there exist constants $k_\theta > 0$ and $k_r > 0$ such that $\|\mathrm{d}(A \circ S(p))^\theta\| \geq \frac{k_\theta}{\|p - v\|}$ and $\|\mathrm{d}(A \circ S(p))^r\| \leq k_r$. Notice that $\mathrm{d}(A \circ S(p)) = \mathrm{d}(A \circ S(p))^r + \mathrm{d}(A \circ S(p))^\theta = \sum_i \frac{\partial (A \circ S)}{\partial u_i} \mathrm{d}u_i(p)$. Let $u_1 = u$. From (2.2), it is clear that $\frac{\partial (A \circ S)}{\partial u} \mathrm{d}u(p)$ is perpendicular to $p - v$ and hence contributes only to $\mathrm{d}(A \circ S(p))^\theta$. Also $\| \sum_{i \geq 2} \frac{\partial (A \circ S)}{\partial u_i} \mathrm{d}u_i(p)\|$ is bounded for all $p \in B(v, \epsilon) \cap D$. Therefore, $\mathrm{d}(A \circ S(p))^\theta = \frac{\partial (A \circ S)}{\partial u} \mathrm{d}u(p) + \Omega(1) = \Omega(\| \frac{\partial (A \circ S)}{\partial u} \mathrm{d}u(p)\|)$ and $\mathrm{d}(A \circ S(p))^r = O(1)$. Again from (2.2), we have

$$\| \frac{\partial (A \circ S)}{\partial u} \mathrm{d}u(p)\| = \frac{\mathrm{dist}(v, l)}{2} \frac{\|p - v\| \, |\, \mathrm{dist}(u_2, l) - \mathrm{dist}(u_n, l)|}{(\mathrm{dist}(p, l) - \mathrm{dist}(v, l))^2}.$$

Now, $|\, \mathrm{dist}(p, l) - \mathrm{dist}(v, l)| \leq \|p - v\|$. Therefore,

$$\| \frac{\partial (A \circ S)}{\partial u} \mathrm{d}u(p)\| = \Omega \left(\frac{|\, \mathrm{dist}(u_2, l) - \mathrm{dist}(u_n, l)|}{\|p - v\|} \right).$$

Since p does not lie on a generalized inflection segment, either $u_n = v$ or $u_2 = v$. Without loss of generality, let $u_n = v$. Since u belongs to l, clearly u_2 must belong to l. Hence $|\, \mathrm{dist}(u_2, l) - \mathrm{dist}(u_n, l)| = \mathrm{dist}(v, l)$ and is a constant for all $p \in B(v, \epsilon) \cap D$. Thus

$$\| \frac{\partial (A \circ S)}{\partial u} \mathrm{d}u(p)\| = \Omega \left(\frac{1}{\|p - v\|} \right).$$

Hence, $\mathrm{d}(A\circ S(p))^\theta = \Omega(\frac{1}{\|p-v\|})$. Therefore $\frac{\|\mathrm{d}(A\circ S(p))^\theta\|}{\|\mathrm{d}(A\circ S(p))^r\|} \geq \frac{k_\theta}{k_r\|p-v\|}$. It follows that

$$\begin{aligned}\|X_Q^r\| &= \frac{1}{(1+\frac{\|\mathrm{d}(A\circ S(p))^\theta\|^2}{\|\mathrm{d}(A\circ S(p))^r\|^2})^{1/2}} \leq \frac{1}{(1+\frac{k_\theta^2}{k_r^2\|p-v\|^2})^{1/2}} = \frac{k_r\|p-v\|}{(k_\theta^2+k_r^2\|p-v\|^2)^{1/2}} \\ &\leq \frac{k_r\|p-v\|}{k_\theta}.\end{aligned}$$

Note that a convex combination of finitely many X_Q^r will also admit a similar inequality and so the assumption that $\{\cup_{\alpha\in\mathcal{A}} I_\alpha\}\cap B(v,\epsilon)\cap D = \emptyset$ is not limiting. Now let $\gamma(t)$ be a solution of X_Q such that $\gamma(0) = v$. Let T be any time such that $\|\gamma(T)-v\| = R$ and for all $t\in[0,T]$, $\gamma(t)\in B(v,\epsilon)\cap D$ and $X_Q^r(\gamma(t))$ is directed away from v. Then clearly, $R = \int_0^T X_Q^r \mathrm{d}t \leq R\frac{k_r}{k_\theta}T$. In other words the time T taken for a trajectory to travel any distance R is greater than $\frac{k_\theta}{k_r}$. This is a contradiction. Therefore, our assumption that for all $t\in[0,T]$, $\gamma(t)\in B(v,\epsilon)\cap D$ is false. So, the trajectory must belong to C for some finite time interval contained in $[0,T]$. We can choose R as small as possible and this implies that there exists a finite time interval $[0,T_C]$ for which $\gamma(t)\in C$. It follows trivially that $t\mapsto A\circ S(\gamma(t))$ is right continuous at t where $\gamma(t) = v$. We can prove similarly that $t\mapsto A\circ S(\gamma(t))$ is left continuous at t where $\gamma(t) = v$ by considering the vector field $-X_Q$ in place of X_Q. This completes the proof that $t\mapsto A\circ S(\gamma(t))$ is continuous.

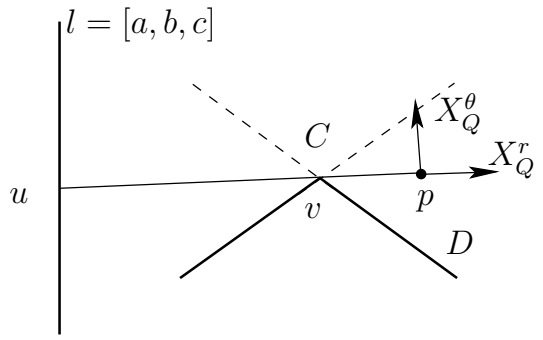


Figure 2.13: Illustration of various notions used in Theorem 2.4.5. The dashed lines represent generalized inflection segments generated by the reflex vertex v and vertices adjacent to it. These divide the region around v that is inside Q into three subregions C , D and E. $u\in \mathrm{Ve}(S(p))$ lies on the line l. The generalized inflection segments including the vertex v are assumed to belong to region C. Note that $D\cap C = \emptyset$.

Next we show that Assumptions (A1), (A2) and (A3) in Theorem 2.3.2 hold. Let $p\in Q\setminus \mathrm{Ve}_r(Q)$ and take $a\in \widetilde{\mathcal{L}}_{X_Q}(-A_Q^\epsilon)(p)$. By definition, there exists $k\in K[X_Q](p)$ such that $a = k\cdot\zeta$ for all

$\zeta \in -\partial A_Q^\epsilon(p)$. In particular, it is true for $\zeta = -\delta k$, for some $\delta > 0$, see Remark 2.4.1. Therefore, $a = -\delta\|k\|^2 \leq 0$. This proves that either $\max \widetilde{\mathcal{L}}_{X_Q}(-A_Q^\epsilon)(p) \leq 0$ or $\widetilde{\mathcal{L}}_{X_Q}(-A_Q^\epsilon)(p) = \emptyset$, i.e., Assumption (A1) is satisfied. Assumption (A2) is a consequence of the continuity of $A \circ S \circ \gamma$. Finally, Assumption (A3) is a consequence of Theorem 2.4.4. Applying now Theorem 2.3.2 and its corollary, we conclude that fact (i) holds. Moreover, we also deduce that any Filippov solution of X_Q converges to the largest weakly invariant set M contained in $\overline{Z}_{X_Q,-A_Q^\epsilon} \cup \mathrm{Ve}_r(Q)$.

To prove fact (ii), let us show that $M = \{p \in Q \mid 0 \in K[X_Q](p)\} \cap (\overline{Z}_{X_Q,-A_Q^\epsilon} \cup \mathrm{Ve}_r(Q))$. Based on Theorem 2.4.4, Theorem 2.3.2 and Corollary 2.3.3, it suffices to show that M is contained in $\{p \in Q \mid 0 \in K[X_Q](p)\}$. Let us note that the set $\{p \in Q \mid 0 \in K[X_Q](p)\}$ is weakly invariant and can be established to be closed following the same reasoning as in Proposition 2.1.1 in [95]. Let $x \in Z_{X_Q,-A_Q^\epsilon}$. Then, $0 \in \widetilde{\mathcal{L}}_{X_Q}(-A_Q^\epsilon)(x)$, i.e., there exists $k \in K[X_Q](x)$ such that $\zeta \cdot k = 0$ for all $\zeta \in -\partial A_Q^\epsilon(x)$. But, $k \in K[X_Q](x)$ implies that there exists $\delta > 0$ such that $\delta k \in -\partial A_Q^\epsilon(x)$, see Remark 2.4.1. Thus, for $\zeta = \delta k$, we get $\delta\|k\|^2 = 0$, that is, $0 \in K[X_Q](x)$. This shows that $Z_{X_Q,-A_Q^\epsilon} \subset \{p \in Q \mid 0 \in K[X_Q](x)\}$. Next, let $x \in \mathrm{Ve}_r(Q) \cap M$. If the set $\{x\}$ is weakly invariant, then by definition $0 \in K[X_Q](x)$. If on the other hand x is not isolated in M, then there exists a sequence of points $\{x_m\}_{m\in\mathbb{N}}$ converging to x such that $x_m \in Z_{X_Q,-A_Q^\epsilon}$ or, alternatively, $0 \in K[X_Q](x_m)$. Because $\{p \in Q \mid 0 \in K[X_Q](p)\}$ is closed, it follows that $0 \in K[X_Q](x)$. Thus we proved that any weakly invariant set contained in $Z_{X_Q,-A_Q^\epsilon} \cup \mathrm{Ve}_r(Q)$ is a subset of $\{p \in Q \mid 0 \in K[X_Q](p)\}$. Again, as in Proposition 2.1.1 in [95], it can be shown that $Z_{X_Q,-A_Q^\epsilon}$ is a closed set and hence the claim that $M \subset \{p \in Q \mid 0 \in K[X_Q](p)\}$ follows. □

Theorem 2.4.5 implies that the single observer converges to a critical point of $A \circ S$ or to a reflex vertex of Q. However, as shown in Figure 2.15, the presence of noise or computational inaccuracies actually works to drive the observer away from a reflex vertex that is not a local maximum. This will also be true for other critical points that are not local maxima.

2.5 Simulation results

To conduct experiments, a simulation environment has been developed in `Matlab`®. There are two levels of the code. The lower level consists of a library containing routines to answer queries such

as whether two points in a two-dimensional polygonal environment are visible to each other. The higher level utilizes these routines and consists of two major portions. In the first, the vertices of the visibility polygon are obtained by means of an $O(n^2)$ algorithm, where n is the number of vertices of the polygonal environment. These are then sorted in counterclockwise order to compute the visibility polygon. The second consists of the controller which decides the direction and the step size of the observer motion at each time instant. The main task of the controller is the calculation of the generalized gradient of the visible area function which is a natural outcome of (2.1) and (2.3). *Such a framework gives the flexibility to easily implement other visibility-based algorithms for single or multiple observers in a polygonal environment. This can be done by extracting the appropriate information using the low level functions and implementing the desired controller.*

Figures 2.15 and 2.17 illustrate the performance of the gradient algorithm in equation (2.7). Computational inaccuracies in the implementation of the algorithm to calculate the visibility polygon have been noticed in some configurations; see the plot of the evolution of visible area with time in Figure 2.15. See Figure 2.16 for the phase portrait of the vector field X_Q for the polygon in Figure 2.14. Simulation results for an observer in a similar polygonal environment containing a hole is shown in Figure 2.18. Our experiments suggest that the observer reaches a local maximum of the visible area in finite time, however this can be shown not to be true in general.

2.6 Conclusions

In this chapter, we have introduced a gradient-based algorithm to optimally locate a mobile observer in a nonconvex environment. We have presented nonsmooth analysis and control design results. The simulation results illustrate that, in the presence of noise, the observer reaches a local maximum of the visible area. In a "highly nonconvex" environment, a single observer may not be able to see a large fraction of the environment. In such a case, a team of observers can be deployed to achieve the same task. We therefore plan to investigate this same visibility objective for teams of observers. This is the subject of the following chapter. Other directions of future research include practical robotic implementation issues as well as other combined mobility and visibility problems.

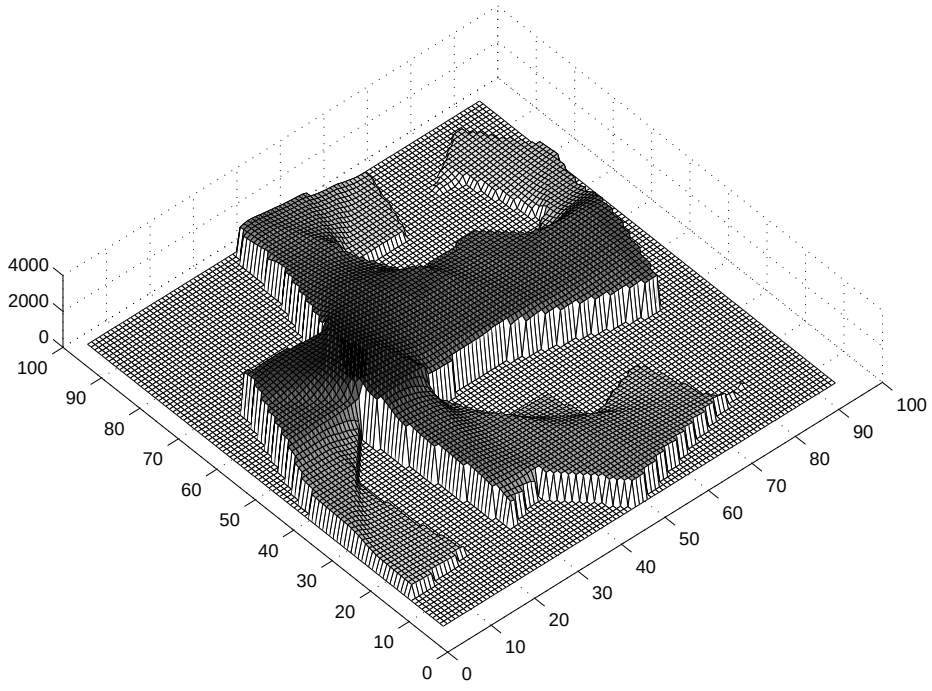


Figure 2.14: Example of visible area function over a typical nonconvex polygon.

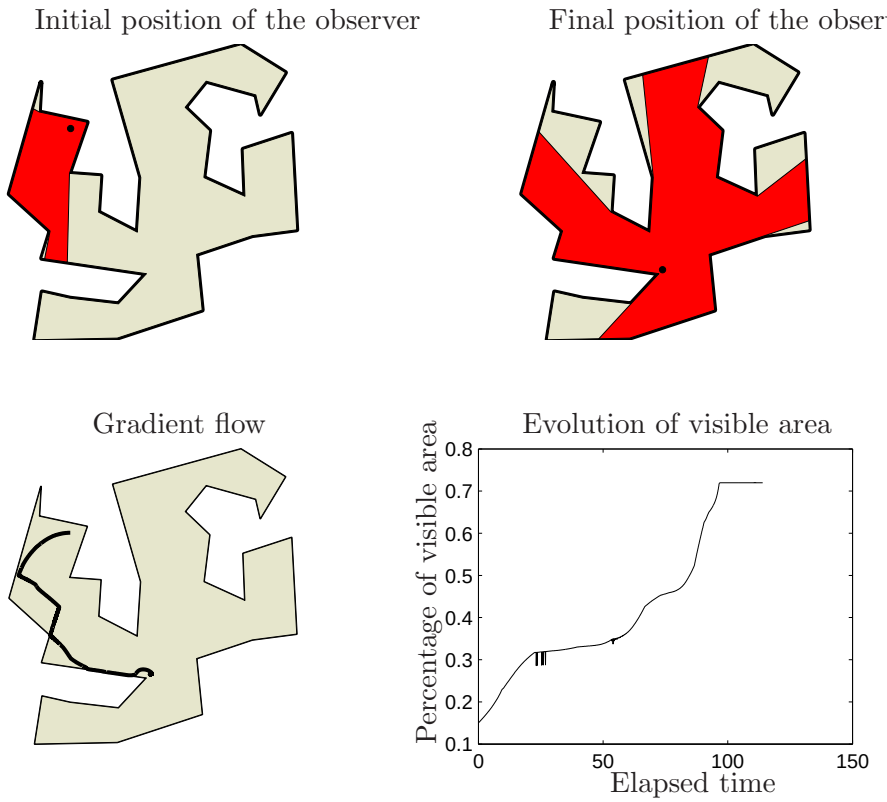


Figure 2.15: Simulation results of the gradient algorithm for the nonconvex polygon depicted in Figure 2.14. The observer arrives, in finite time, at a local maximum. Note here that the observer visits a reflex vertex at some point in its trajectory but comes out of it due to computational inaccuracies because it is not a local maximum.

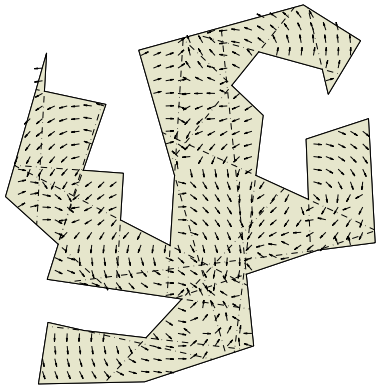


Figure 2.16: Example of vector field over the nonconvex polygon in Figure 2.14.

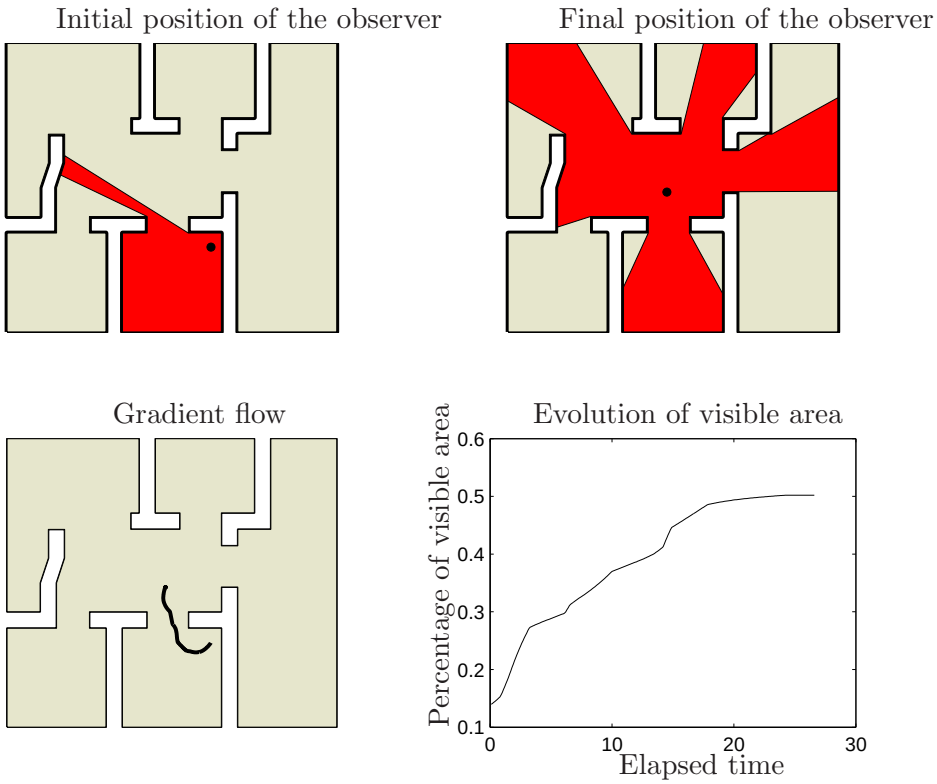


Figure 2.17: Simulation results of the gradient algorithm for the nonconvex polygon in Figure 2.1. The observer arrives, in finite time, at a local maximum.

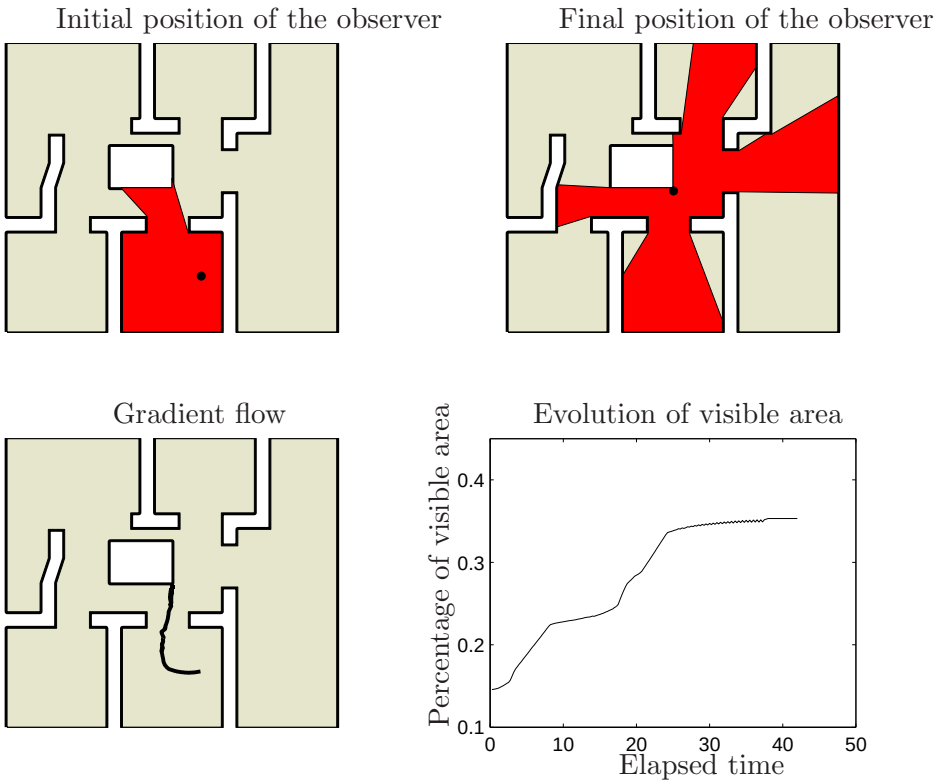


Figure 2.18: Simulation results of the gradient algorithm for an observer in a nonconvex environment with a hole. The observer arrives, in finite time, at a reflex vertex.

CHAPTER 3

Deployment of multiple robotic agents

3.1 Introduction

Recently, much research has focused on the use of unmanned robots for the purpose of surveillance and search. This chapter provides algorithms to deploy robotic agents with limited capabilities to monitor an unknown environment. The environment is assumed to be simply connected, i.e., without holes, and polygonal. The agents are modeled as point masses with first-order dynamics. The agents are all identical except with distinct unique identifiers (UID). No assumption is made about the UIDs except that they are distinct. The agents are assumed to operate asynchronously and to have limited communication and sensing capabilities: they can communicate only with agents within line-of-sight and they can sense the distance to the environment boundary or to any other agent within line of sight. The first objective is to deploy the agents starting from a single location so that all points of the environment are visible to at least one agent. We present a distributed algorithm to solve the above problem requiring no more than $\lfloor n/3 \rfloor$ agents, where n is the number of vertices in the environment. A second objective is to deploy the agents so that the visibility graph of the final configuration of the agents is connected. We also present a distributed algorithm to solve this problem requiring no more than $(n-1)/2$ agents. The time complexity of the proposed algorithms is also investigated.

Deployment of robotic sensors have been studied in centralized and decentralized contexts, centralized referring to the fact that the environment is known a priori and decentralized otherwise. In the former setting, this problem becomes the classical Art Gallery Problem in the computational geometry literature, which aims to find both the minimum number of "guards" required and the locations of these guards to achieve complete visibility of a given polygonal environment. This problem was first analyzed by Chvátal, see [76], in the famous Art Gallery Theorem stating that

$\lfloor n/3 \rfloor$ guards are sufficient and sometimes necessary to guard any simply connected polygon with n vertices. In [77], Hernández-Peñalver gives an induction argument to prove that $(n-2)/2$ connected set of guards are always sufficient and occasionally necessary in a simply connected orthogonal environment. A constructive proof based on a coloring argument was later given by Pinciu [78].

Relevant works in the decentralized setting include [75], where an incremental heuristic for deployment is proposed, and [20] where distributed algorithms for coverage control based on Voronoi partitions are designed. Coordinated deployment of multiple heterogeneous robots has also been studied in [97]. Deployment locations are user-specified after an initial map of the unknown environment has been built.

Another related body of work is that of robotic exploration of unknown environments, since a natural two-stage strategy to solve the deployment problem is to first explore the environment and then solve the centralized problem. The most relevant literature to the current problem include topological exploration of graph-like environments by single and multiple robots [73, 74, 98, 99]. In these problems, it is either assumed that agents can synchronize their motions to fuse their data, or read and write to the nodes of the graph. These assumptions are stronger compared to the assumptions in the present treatment. Synchronizing motions and fusing data are additional complications, especially in the presence of limited communication bandwidth. Also, writing to nodes in a graph is not possible in the case of exploration of unknown environments. Finally, the problem of deployment is very different from the problem of exploration. Assuming that exploration is possible without accumulating errors, in the absence of a central processor, the robots would have to allocate tasks amongst themselves. Our strategy, on the other hand, is a simple one-step strategy for deployment, without the need for synchronization, achieving the worst-case optimal bounds in terms of number of robots required, and under limited communication constraints.

Notation Given any finite set X, let $|X|$ denote the cardinality of X. Given two points x, $y \in \mathbb{R}^2$, let $[x, y]$ denote the closed line segment joining the two points. Let $f : X \rightrightarrows Y$ denote a function mapping elements of X to subsets of Y.

3.2 Preliminaries

3.2.1 Visibility and graph theoretic concepts

We look at polygonal environments, denoted by Q, that are *simply connected*, i.e., they are connected and do not contain any holes. Unless otherwise stated, all environments in this chapter will be assumed to be simply connected and polygonal. Let $\mathrm{Ve}(Q)$ and $\mathrm{Ver}(Q)$ be the list of vertices and reflex[1] vertices of Q, respectively. A *diagonal* of Q is a line segment $[v_1, v_2]$ where v_1, $v_2 \in \mathrm{Ve}(Q)$ and $[v_1, v_2] \setminus \{v_1, v_2\}$ belongs to the interior of Q. We now describe some useful notions of visibility. A point $q \in Q$ is *visible from* $p \in Q$ if $[p, q] \subset Q$. The *visibility set* $\mathcal{V}(p, Q) \subseteq Q$ from a point $p \in Q$ is the set of points in Q visible from p. Figure 3.1 (left) shows a graphical illustration of the geometric notions introduced so far. A subset of Q is star-shaped if it contains a point $p \in Q$ with the property that $\mathcal{V}(p, Q) = Q$, i.e., the whole polygon is visible from p. The set of all such points in Q is referred to as the *kernel* of Q, and denoted by $\ker(Q)$. It follows that, if a polygon Q is not star-shaped, then $\ker(Q) = \emptyset$. Let $\ker^*(Q)$ denote the set of all subsets P of Q such that $\cup_{p \in P} \mathcal{V}(p, Q) = Q$, i.e., any point of the polygon is visible from at least a point in P. Notice that if Q is a polygon, then there exist elements in $\ker^*(Q)$ with finite cardinality. According to the Art Gallery Theorem [76], if Q is simply connected with n vertices, then there exists $P \in \ker^*(Q)$ with $|P| \leq \lfloor n/3 \rfloor$. With a slight abuse of terminology, henceforth, we shall use the term *kernel* of a polygon Q to refer to $\ker^*(Q)$. Given Q, it is always possible to triangulate it, i.e., partition into triangles with disjoint interiors, by means of diagonals; see [84] and Figure 3.1 (right). Let $\mathcal{T}_Q$ represent a triangulation of Q.

Let us now introduce some graph theoretic concepts. A connected graph with no simple cycles is a tree. In a rooted tree, a node is designated as the root, and a natural direction is induced on the edges of the tree, which in this chapter is taken to be away from the root. A node, $\mathcal{N}_j$, is a *child* of another node, $\mathcal{N}_i$, if there exists an edge directed from $\mathcal{N}_i$ to $\mathcal{N}_j$. In turn, $\mathcal{N}_i$ is the *parent* of $\mathcal{N}_j$. Two nodes that share the same parent are *siblings*. A *predecessor* of $\mathcal{N}_i$ is any node from which a directed path exists to $\mathcal{N}_i$. A *successor* of $\mathcal{N}_i$ is any node to which a directed path exists from $\mathcal{N}_i$. Let $\mathrm{pre}(\mathcal{N}_i)$ and $\mathrm{suc}(\mathcal{N}_i)$ denote the set of predecessors and successors of $\mathcal{N}_i$ respectively. The *depth* of a node is the number of edges in the path to the node from the root. Consequently

[1] A vertex is a reflex vertex if its interior angle is strictly greater than π radians.

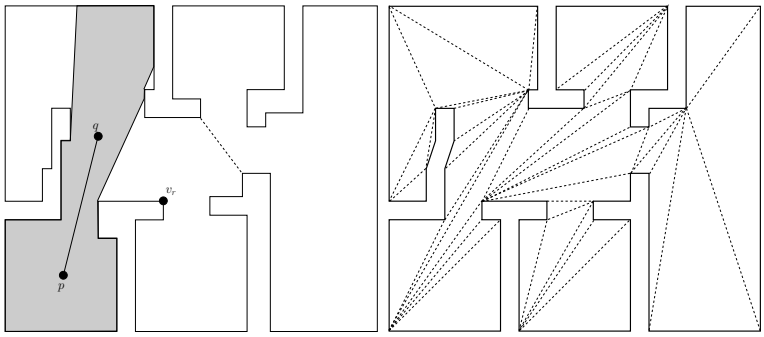


Figure 3.1: Left: Illustration of some preliminary notions. A simply connected environment Q (outer polygon), a reflex vertex v_r, a diagonal (dashed line segment) and two points p, $q \in Q$ mutually visible to each other (the line segment $[p, q]$ belongs to Q) and the visibility set $\mathcal{V}(p, Q)$ (shaded region). Right: A simply connected environment, Q, in the shape of a typical floor plan and its triangulation by means of diagonals.

the depth of a tree is the maximum depth of all nodes. The *visibility graph* $\mathcal{G}_{\text{vis},Q}(\{p_1, \ldots, p_n\})$ of a set of points $\{p_1, \ldots, p_n\}$ in Q is a graph with the node set equal to $\{p_1, \ldots, p_n\}$ and with (p_i, p_j) being an edge if and only if $[p_i, p_j] \subseteq Q$ and vice versa.

3.2.2 Robotic network model

We consider a group of robotic agents modeled as point masses, moving in a simple nonconvex polygonal environment, Q. Each agent has a unique identifier (UID), say i. Let a_i refer to the position of agent i. Each agent is equipped with an omnidirectional line-of-sight range sensor. Thus, the agent can sense its star-shaped visibility set $\mathcal{V}(a_i, Q)$. It can communicate with any other agent within line-of-sight. Each agent also stores in its memory an information vector $\mathcal{I}_i$ which it can broadcast to all agents in its communication region. Such a broadcast is denoted by `BROADCAST`$(i, \mathcal{I}_i)$ and also includes the agent's UID. It can also receive broadcasts from other agents. We also assume that there is a time delay between a broadcast and the corresponding reception; the time delay is arbitrary but upper bounded by $\delta > 0$. An agent has access to some memory $\mathcal{M}_i$ which uses to store information not necessarily available via local sensing and communication. We shall describe in detail the structures of the information vector and the memory in Sections 3.4.3 and 3.5.3.

Every agent i repeatedly performs the following sequence of actions beginning at a time instant, say T_l^i:

(i) send repeated BROADCAST$(i, \mathcal{I}_i)$ every δ seconds, until it starts moving;

(ii) LISTEN for at least 2δ seconds before processing the information;

(iii) PROCESS the necessary information. Also continue to LISTEN during this interval. Assume that the total LISTEN interval is less than λ seconds;

(iv) MOVE to a desired point.

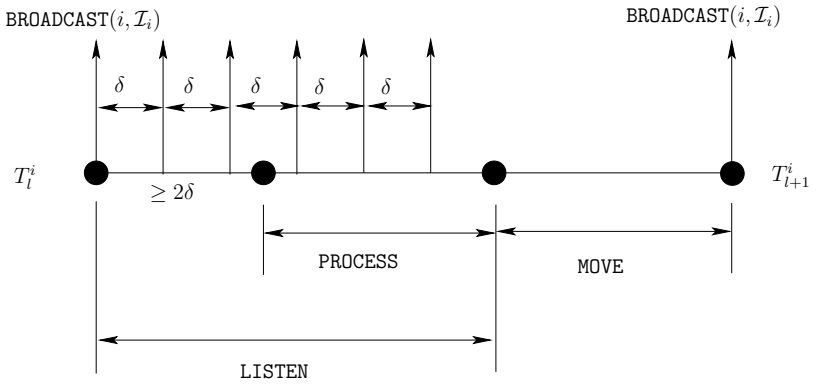


Figure 3.2: Sequence of actions for agent i beginning at time T_l^i. Instantaneous BROADCAST$(i, \mathcal{M}_i)$ events are represented by vertical pulses. The MOVE interval might be empty if the agent does not move.The subsequent instant T_{l+1}^i is the time when the agent stops performing the MOVE action and it is not predetermined.

Agent i, in the MOVE state, is capable of moving at any time t according to the following discrete-time control system:

$$a_i(t + \Delta t) = a_i(t) + u_i,$$

where $\|u_i\| \leq 1$. The control action depends on time, on the memory $\mathcal{M}_i(t)$, and on the information obtained from communication and sensing. The subsequent wake-up instant T_{l+1}^i is the time when the agent stops performing the MOVE action and it is not predetermined. This model of robotic agents is similar in spirit to the *partially asynchronous network model* described in [48].

Remark 3.2.1 (Higher-order dynamics)**.** *For the deployment algorithm presented later, we need only the assumption that an agent can traverse between two points in a simply connected polygon if they are mutually visible to one another. Thus, the algorithm will also work for agents with more complex dynamics provided the above assumption is true. However, the analysis of the time*

complexity of the algorithm does depend on the time taken to move between two points and is thus specific to the first-order dynamics model considered above. □

3.2.3 Problem statement

We consider the following two problems for the robotic network model introduced in the previous section. We assume that robots enter an environment from the same location.

Our first objective is to design a provably correct distributed algorithm to deploy a group of agents on locations such that each point of the environment is visible to at least one agent. We aim to achieve this objective with a sufficiently small number of agents, i.e., $\lfloor n/3 \rfloor$ agents in an environment with n vertices. This is the *visibility-based nonconvex deployment problem.*

Our second objective is to design a provably correct distributed algorithm to deploy a group of agents on locations such that each point of the environment is visible to at least one agent under the additional constraint that the final configuration of agents is connected. We aim to achieve this objective with a sufficiently small number of agents, i.e., $\lfloor n/2 \rfloor$ agents in an environment with n vertices. This is the *visibility-based nonconvex deployment problem with connectivity.*

In what follows, we present analogous solutions to the two problems. We first treat the second problem and we then present a parallel, more sophisticated treatment of the first problem. This chapter organization simplifies the presentation of the two algorithms.

3.3 An incremental partition algorithm and the resulting triangulation

In this section, we describe a procedure to incrementally partition a simply connected environment Q into star-shaped sets. Given Q and one of its vertices s, the procedure, referred to as the `Incremental Partition Algorithm`, results in the computation of a finite ordered set of star-shaped polygonal sets $\mathcal{R}$. The procedure also returns a finite ordered set of points $\mathcal{P}$ with $|\mathcal{P}| = |\mathcal{R}|$ and with $\mathcal{R}_i$ visible from p_i, where $\mathcal{R}_i$ and p_i are the ith elements of $\mathcal{R}$ and $\mathcal{P}$, respectively. Thus, $\mathcal{P} \in \ker^*(Q)$.

The `Incremental Partition Algorithm` consists of two components. The first component is

an `Initialization` routine and the second one is a `Star-shaped Set Computation` algorithm. The `Initialization` routine is as follows:

Algorithm: `Initialization`

Input: Simply-connected polygon Q and $s \in \mathrm{Ve}(Q)$

(1) Compute the set of vertices of Q visible from s.

(2) Arrange the vertices in clockwise order, with s being the first. Let $\mathcal{R}_{\mathrm{diag}}$ be the polygon represented by the ordered set.

Return: $\mathcal{R}_{\mathrm{diag}}$; s

At the end of the `Initialization` routine, we obtain a polygon $\mathcal{R}_{\mathrm{diag}}$ as illustrated in Figure 3.3. The `Star-shaped Set Computation` algorithm is as follows:

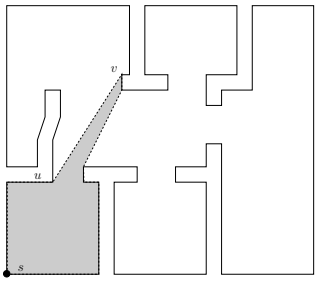


Figure 3.3: Illustration of the `Initialization` routine. The shaded region represents $\mathcal{R}_{\mathrm{diag}}$. The dashed edges of $\mathcal{R}_{\mathrm{diag}}$ are diagonals of Q.

Algorithm: `Star-shaped Set Computation`

Input: Simply-connected polygon Q, $s \in \mathrm{Ve}(Q)$, and diagonal (u, v) not containing s

(1) Compute the set of vertices visible from v on the opposite side of (u, v) as s.

(2) Arrange the vertices consecutively, with v being the first and u being the last. Let $\mathcal{R}_{\mathrm{diag}}$ be the polygon represented by the ordered set.

Return $\mathcal{R}_{\mathrm{diag}}$; v

Lemma 3.3.1 (Properties of $\mathcal{R}_{\mathrm{diag}}$)**.** *Given a simply connected polygon Q, $s \in \mathrm{Ve}(Q)$ and a diagonal (u, v) of Q, the polygon $\mathcal{R}_{\mathrm{diag}}$ has the following properties:*

(i) $|\mathrm{Ve}(\mathcal{R}_{\mathrm{diag}})| \geq 3$*;*

(ii) $\mathcal{R}_{\text{diag}}$ *is star-shaped and is visible from* v*;*

Proof. Since (u, v) is a diagonal of Q, it partitions Q into two smaller polygons, one containing s and the other not. Each of the polygons can be triangulated by means of diagonals of Q. Let T be the triangle containing (u, v) as an edge in the polygon not containing s. Let w be the third vertex of this triangle. Clearly, w and u are visible from v and, therefore, both these vertices will be vertices of $\mathcal{R}_{\text{diag}}$. Thus, $|\operatorname{Ve}(\mathcal{R}_{\text{diag}})| \geq 3$. Statement (ii) is trivial and follows from the construction of $\mathcal{R}_{\text{diag}}$. □

Beginning with the `Initialization` routine, the `Star-shaped Set Computation` algorithm is executed for every diagonal generated until there are no diagonals. However, for every diagonal the choice of which vertex serves as v is important. This is done as follows.

> `Odd-numbered placement scheme:` During the execution of the `Initialization` or the `Star-shaped Set Computation` algorithms, for any edge of $\mathcal{R}_{\text{diag}}$ that is a diagonal of Q, the vertex that is odd numbered in the ordering of the vertices is chosen as the vertex v; see Figure 3.5.

The `Initialization` routine together with the `Star-shaped Set Computation` algorithm and the `Odd-numbered placement scheme` for choosing the vertex v constitute the `Incremental Partition Algorithm`. Figure 3.5 illustrates an execution of the `Incremental Partition Algorithm` algorithm for a given polygon Q and $s \in \operatorname{Ve}(Q)$. Let $\mathcal{R}$ denote the list of $\mathcal{R}_{\text{diag}}$ thus generated and let $\mathcal{P}$ be the corresponding list of kernel points such that $\mathcal{R}_i$ is visible from p_i, where recall $\mathcal{R}_i$ and p_i are the ith elements of $\mathcal{R}$ and $\mathcal{P}$, respectively.

Remark 3.3.2 (Modification of $\mathcal{P}$)**.** *Notice that each invocation of* `Star-shaped Set Computation` *algorithm operates on a unique diagonal. The resultant star-shaped set can possibly contain two adjacent edges for both of which the* `Star-shaped Set Computation` *algorithm might have to be invoked. In the case that these two edges (which are diagonals of* Q*) share the same odd-numbered vertex,* $\mathcal{P}$ *will contain two elements, say* p_i *and* p_j *both being the same vertex but with distinct* $\mathcal{R}_i$ *and* $\mathcal{R}_j$*. In this case, we modify* $\mathcal{P}$ *as follows. The elements* p_i *and* p_j *are replaced by a single element* p_k *and* $\mathcal{R}_i$ *and* $\mathcal{R}_j$ *are replaced by* $\mathcal{R}_k = \mathcal{R}_i \cup \mathcal{R}_j$*. In what follows, without loss of generality, we will assume that for any* k*, we can write* $\mathcal{R}_k$ *as the union of* $\mathcal{R}'_k$ *and* $\mathcal{R}''_k$*. This is because even if*

p_i is not a point of the type described above, we can still write $\mathcal{R}_i$ to be union of $\mathcal{R}'_i$ and $\mathcal{R}''_i$ where $\mathcal{R}'_i = \mathcal{R}''_i$. Notice that the elements of $\mathcal{P}$ might still contain multiple copies of the same point. □

Lemma 3.3.3 (Properties of $\mathcal{P}$)**.** *Given a simply connected polygon Q and $s \in \mathrm{Ve}(Q)$, let $\mathcal{R}$ and $\mathcal{P}$ be the outputs of the* `Incremental Partition Algorithm` *with the modification in Remark 3.3.2. Then the following properties hold:*

(i) $\mathcal{P} \in \ker^*(Q)$*;*

(ii) $|\mathcal{P}| \leq n-2$*, where* $n = |\mathrm{Ve}(Q)|$*; and*

(iii) there exist Q and $s \in \mathrm{Ve}(Q)$ such that $|\mathcal{P}| = n-2$.

Proof. Statement (i) follows directly from the construction. Now, from Lemma 3.3.1 , we know that $|\mathrm{Ve}(\mathcal{R}_i)| \geq 3$ for any i. Thus, by drawing diagonals to the vertices of $\mathcal{R}_i$ from the corresponding p_i, we can triangulate $\mathcal{R}_i$ and any $\mathcal{R}_i$ will contain at least 1 triangle. Since the total number of triangles obtained by triangulation of polygon by diagonals is $n-2$, we have that $|\mathcal{R}| \leq n-2$. But $|\mathcal{P}| = |\mathcal{R}|$ and statement (ii) follows. For the proof of statement (iii), please refer to Figure 3.4.

□

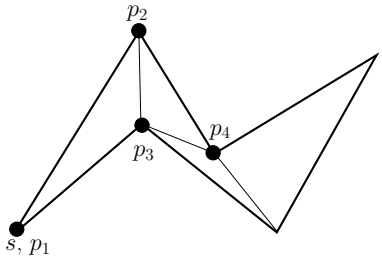


Figure 3.4: Illustration of a polygon Q and $s \in \mathrm{Ve}(Q)$ such that $|\mathcal{P}| = n-2$. Here $n = 6$ and $|\mathcal{P}| = 4$. The point-set represented by the black discs is $\mathcal{P}$. The diagonals represent the boundary of the sets $\mathcal{R}$.

Remark 3.3.4 (Triangulation using $\mathcal{R}$ and $\mathcal{P}$)**.** *First let us notice that given $\mathcal{R}_j \in \mathcal{R}$ and $p_j \in \mathcal{P}$, we can triangulate $\mathcal{R}_j$ by drawing diagonals to each of the vertices of $\mathcal{R}_j$ from p_j. This is possible since $\mathcal{R}_j$ is star-shaped and is visible from p_j. Following this procedure for each of the elements of $\mathcal{R}$, we end up with a triangulation of the environment. Let us denote this triangulation by $\mathcal{T}_Q(\mathcal{R}, \mathcal{P})$.*

□

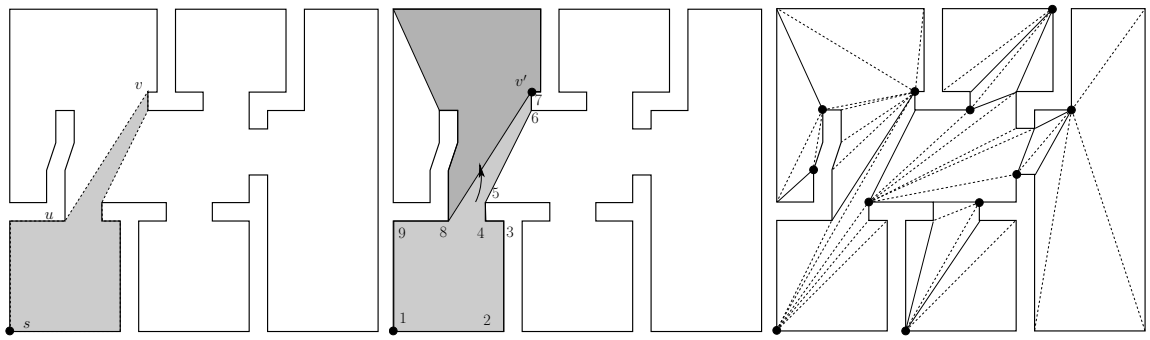


Figure 3.5: Illustration of the `Incremental Partition Algorithm`. The left plot shows the execution of the `Initialization` algorithm. The center plot shows how to pick the vertex v and the execution of `Star-shaped Set Computation` algorithm. The right plot shows the partition $\mathcal{R}$ and the corresponding kernel points (black discs).

3.4 Connected deployment of agents

In this section, we present a distributed deployment algorithm that guarantees that the visibility graph of the final configuration of the agents has one connected component. We begin by identifying a set P^*_{con} which is the desired point-set for deployment. This point-set should have the property that $P^*_{\text{con}} \in \ker^*(Q)$ and $\mathcal{G}_{\text{vis-}Q}(P^*_{\text{con}})$ has a single connected component. It can be easily checked that the set $\mathcal{P}$ computed according to the `Incremental Partition Algorithm` in Section 3.3 possesses these properties. However, we are interested in finding a set possessing the same properties with smaller cardinality. In [77], it is proved that there exists a point set satisfying the above properties with cardinality at most $(n-2)/2$. In [78], a constructive algorithm based on coloring is given to compute the set. However, the algorithm in [78] requires centralized computation and is not amenable to a distributed implementation. In this section, we present a novel yet simple `Connected Kernel Computation Algorithm` to compute P^*_{con} whose cardinality is at most $(n-1)/2$, i.e., it differs from the bound available in the literature by at most 1. Moreover, our algorithm is well-suited for a distributed implementation. Indeed, we present the `Connected Depth-first Deployment` algorithm based on the `Connected Kernel Computation Algorithm` to solved the connected deployment problem.

3.4.1 Navigation graph for connected deployment

We begin by defining a graph that is used in the `Connected Kernel Computation Algorithm` and for navigation in the `Connected Depth-first Deployment`.

Definition 3.4.1 (Navigation graph)**.** *Given a simply connected polygonal environment, Q, $s \in \mathrm{Ve}(Q)$, let $\mathcal{R}$ and $\mathcal{P}$ be the outputs of the `Incremental Partition Algorithm`. The* navigation graph*, $\mathcal{G}_{\text{nav-con}}(s, Q)$ is the graph with $\mathcal{P}$ as the node set and an edge between p_i and p_j if and only if $\mathcal{R}_i$ and $\mathcal{R}_j$ share a common edge and vice versa.*

The following lemma summarizes the properties of $\mathcal{G}_{\text{nav-con}}(s, Q)$.

Lemma 3.4.2 (Properties of the navigation graph)**.** *The graph, $\mathcal{G}_{\text{nav-con}}(s, Q)$, satisfies the following properties:*

(i) $\mathcal{G}_{\text{nav-con}}(s, Q)$ is a tree;

(ii) adjacent nodes of the graph are mutually visible to each other.

Proof. Statement (i) is a consequence of the fact that Q is simply connected and that $\mathcal{R}_i \cap \mathcal{R}_j$ is a diagonal of Q for any i and j. Statement (ii) follows from the construction of $\mathcal{R}$. □

Note that $s \in \mathcal{P}$ by construction. Let us designate s as the root of $\mathcal{G}_{\text{nav-con}}(s, Q)$. Thus, $\mathcal{G}_{\text{nav-con}}(s, Q)$ is now a rooted tree. Figure 3.6 shows an example of the navigation graph. We are now in a position to present the `Connected Kernel Computation Algorithm`.

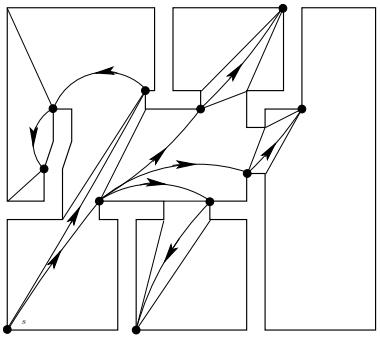


Figure 3.6: Illustration of the navigation graph, $\mathcal{G}_{\text{nav-con}}(s, Q)$. The vertex s is the root; the black discs represent the nodes and the directed arcs represent the edges.

3.4.2 A connected kernel point-set

Let $\mathcal{R}$ and $\mathcal{P}$ be the outputs of the `Incremental Partition Algorithm`. The `Connected Kernel Computation Algorithm` to find a set $P^*_{\text{con}} \subset \mathcal{P}$ such that $P^*_{\text{con}} \in \ker^*(Q)$ and $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$ has a single connected component is as follows:

Algorithm: `Connected Kernel Computation Algorithm`

Input: The navigation graph $\mathcal{G}_{\text{nav-con}}(s, Q)$ with simply connected polygon Q and $s \in \text{Ve}(Q)$

Initialization: $P^*_{\text{con}} := \emptyset$, $P_{\text{mark}} := \emptyset$

While $P_{\text{mark}} \neq \mathcal{P}$ **do**

(1) Take any node, say p_i, of $\mathcal{G}_{\text{nav-con}}(s, Q)$ which has no children or whose children all belong to P_{mark}.

(2) Let $\mathcal{R}_i := \mathcal{R}'_i \cup \mathcal{R}''_i$ where $\mathcal{R}'_i$ and $\mathcal{R}''_i$ are as described in Remark 3.3.2.

(3) **If** ($|\,\text{Ve}(\mathcal{R}'_i)| = 3$ **AND** number of children of p_i in $\mathcal{R}'_i$ belonging to P^*_{con} is equal to one) **AND** ($|\,\text{Ve}(\mathcal{R}''_i)| = 3$ **AND** number of children of p_i in $\mathcal{R}''_i$ belonging to P^*_{con} is equal to one) **then**

(4) Update $P_{\text{mark}} := P_{\text{mark}} \cup \{p_i\}$.

(5) **else**

(6) Update $P_{\text{mark}} := P_{\text{mark}} \cup \{p_i\}$ and $P^*_{\text{con}} := P^*_{\text{con}} \cup \{p_i\}$.

Return: P^*_{con}

Please refer to Figure 3.9 for an example of a point-set P^*_{con} obtained via the `Connected Kernel Computation Algorithm`.

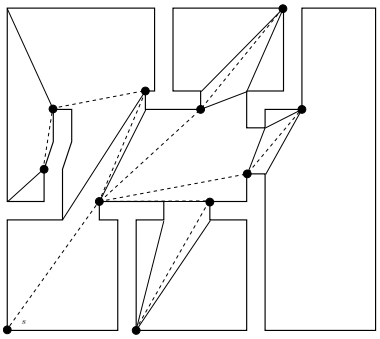


Figure 3.7: An illustration of a polygon Q and $s \in \text{Ve}(Q)$ the corresponding point-set P^*_{con}. The dashed segments show some of the edges of the visibility graph, $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$.

The following lemma states that the `Connected Kernel Computation Algorithm` terminates after a finite number of steps.

Lemma 3.4.3 (Execution time of the `Connected Kernel Computation Algorithm`). *Given Q, $s \in \mathrm{Ve}(Q)$ and the navigation graph $\mathcal{G}_{\text{nav-con}}(s, Q)$, the `Connected Kernel Computation Algorithm` terminates in exactly $|\mathcal{P}|$ iterations.*

Proof. The proof follows from the observation that as long as $P_{\text{mark}} \neq \mathcal{P}$, there always exists a node which has no children or whose all children belong to P_{mark}. Thus p_i in step (1) exists as long as $P_{\text{mark}} \neq \mathcal{P}$. It then follows that, as long as $P_{\text{mark}} \neq \mathcal{P}$, the cardinality of P_{mark} increases by one at every iteration. The result follows. □

The properties of the set P^*_{con} as computed by the `Connected Kernel Computation Algorithm` are as follows.

Proposition 3.4.4 (Properties of P^*_{con}). *Given Q, $s \in \mathrm{Ve}(Q)$ and the navigation graph $\mathcal{G}_{\text{nav-con}}(s, Q)$, let P^*_{con} be the output of the `Connected Kernel Computation Algorithm`. Then the following hold true:*

(i) $P^*_{\text{con}} \in \ker^*(Q)$;

(ii) $|P^*_{\text{con}}| \leq \dfrac{n-1}{2}$;

(iii) there exist Q and $s \in \mathrm{Ve}(Q)$ such that $|P^*_{\text{con}}| = \dfrac{n-1}{2}$; *and*

(iv) $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$ *has a single connected component.*

Proof. From the `Connected Kernel Computation Algorithm`, note that an element $p_i \in \mathcal{P}$ is not included in P^*_{con} if the conditions in step (3) are satisfied. But this means that $\mathcal{R}'_i$ and $\mathcal{R}''_i$ are triangles and there exists exactly one child of p_i belonging to P^*_{con} in each of $\mathcal{R}'_i$ and $\mathcal{R}''_i$. But since $\mathcal{R}'_i$ and $\mathcal{R}''_i$ are both triangles, they must be completely visible from the two children. Thus $\mathcal{R}_i$ is completely visible by elements in P^*_{con}. Statement (i) follows.

We now prove statement (ii). Let $p_j \in P^*_{\text{con}}$. We show that (a) if $p_j \neq s$, then we can always assign uniquely at least two triangles from $\mathcal{T}_Q(\mathcal{R}, \mathcal{P})$ to p_j; and (b) if $p_j = s$, then we can assign at least one triangle $\mathcal{T}_Q(\mathcal{R}, \mathcal{P})$ to p_j. Since the total number of triangles is $n - 2$, we get that

$2(|P^*_{\text{con}}| - 1) + 1 \leq n - 2$; thus $|P^*_{\text{con}}| \leq \frac{n-1}{2}$. We prove claim (a). Since $p_j \neq s$, p_j has a parent in $\mathcal{G}_{\text{nav-con}}(s, Q)$, say p_i. Let us first assign the triangle in $\mathcal{R}_i$ with $\mathcal{R}_i \cap \mathcal{R}_j$ as an edge to p_j. Next, we will assign a triangle in $\mathcal{R}_j$ to p_j. Let $\mathcal{R}_j = \mathcal{R}'_j \cup \mathcal{R}''_j$. Let us assume without loss of generality that $\mathcal{R}'_j$ and $\mathcal{R}''_j$ are distinct. Since $p_j \in P^*_{\text{con}}$, we have that either ($|\operatorname{Ve}(\mathcal{R}'_j)| > 3$ or number of children of p_j in $\mathcal{R}'_j$ belonging to P^*_{con} is not equal to one) or ($|\operatorname{Ve}(\mathcal{R}''_j)| > 3$ or number of children of p_j in $\mathcal{R}''_j$ belonging to P^*_{con} is not equal to one). Again without loss of generality assume that $|\operatorname{Ve}(\mathcal{R}'_j)| > 3$ or number of children of p_i in $\mathcal{R}'_i$ belonging to P^*_{con} is not equal to one. If $|\operatorname{Ve}(\mathcal{R}'_j)| > 3$, then let $|\operatorname{Ve}(\mathcal{R}'_j)| = 2m+1$ where $m \geq 2$ is an integer. The number of odd numbered vertices in $\mathcal{R}'_j$ is $m+1$. Therefore, the number of children of p_j is at most m according to the `Odd-numbered placement scheme` (since the vertex numbered 1 is p_j itself). Each of these m children are assigned m triangles in $\mathcal{R}'_j$. The total number of triangles in $\mathcal{R}'_j$ is equal to $2m + 1 - 2 = 2m - 1$. Thus, the number of triangles that are not assigned to any child is $2m - 1 - m = m - 1$. Since $m \geq 2$, we have that at least one triangle in $\mathcal{R}_j$ is not assigned to any child. Thus, we can assign at least one triangle in $\mathcal{R}_j$ to p_j. The case when $|\operatorname{Ve}(\mathcal{R}'_j)| = 2m$ and $m \geq 2$ can be argued similarly. Note that since $\mathcal{R}'_j$ is a polygon, $m \neq 1$. Now, if $|\operatorname{Ve}(\mathcal{R}'_j)| = 3$ and number of children of p_j in $\mathcal{R}'_j$ belonging to P^*_{con} is not equal to one, then the number of children of p_j in $\mathcal{R}'_j$ belonging to P^*_{con} must be zero. This is because $\mathcal{R}'_j$ is a triangle and therefore can have at most one child. Since p_j has no child in $\mathcal{R}'_j$, then the triangle in $\mathcal{R}'_j$ can be assigned to p_j. Thus, we have proved claim (a). To prove claim (b), notice that p_j does not have a parent. Therefore, we can only assign at least one triangle to it belonging to $\mathcal{R}_j$ as shown in case (a). This completes our proof of statement (ii).

For the proof of statement (iii), please refer to Figure 3.8.

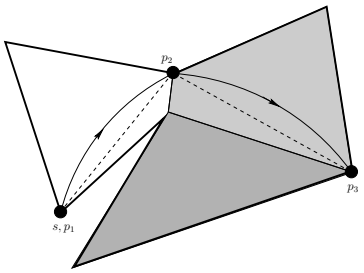


Figure 3.8: An illustration of a polygon Q and $s \in \operatorname{Ve}(Q)$ where $|P^*_{\text{con}}| = (n-1)/2$. Here $n = 7$. The outer polygon is Q. The black discs represent $\mathcal{P}$. The unshaded region is $\mathcal{R}_1$, the lightly shaded region is $\mathcal{R}_2$ and the darkly shaded region is $\mathcal{R}_3$. Directed arcs represent the edges of $\mathcal{G}_{\text{nav-con}}(s, Q)$. The set P^*_{con} is equal to $\{p_1, p_2, p_3\}$.

Finally, we prove statement (iv). First, we show that if (p_i, p_j) is an edge of $\mathcal{G}_{\text{nav-con}}(s, Q)$, then either p_i or p_j belongs to P^*_{con}. Without loss of generality, let p_j be the parent of p_i in $\mathcal{G}_{\text{nav-con}}(s, Q)$. If p_i is a leaf of $\mathcal{G}_{\text{nav-con}}(s, Q)$, then $p_i \in P^*_{\text{con}}$. This is because p_i is a node that satisfies the requirement in step (1) of the `Connected Kernel Computation Algorithm`. Also, since it cannot have any children, the condition in step (3) is also not satisfied. Thus, p_i is included in P^*_{con} in step (6). If now p_i is not a leaf, then let p_h be a child of p_i. If $p_i \notin P^*_{\text{con}}$, then p_j has at least one child location which does not belong to P^*_{con}. Then the condition in step (3) of the `Connected Kernel Computation Algorithm` algorithm cannot hold true for p_j. This is because if $|\mathcal{V}(R'_j)| = 3$, then p_j can have at most one child in $\mathcal{R}'_j$. Similarly, if $|\mathcal{V}(R''_j)| = 3$, then p_j can have at most one child in $\mathcal{R}''_j$. But at least one of the locations is p_i which does not belong to P^*_{con}. Thus, $p_j \in P^*_{\text{con}}$. This proves that for any edge (p_i, p_j) of $\mathcal{G}_{\text{nav-con}}(s, Q)$, at least one of p_i or p_j must belong to P^*_{con}. Thus, if p_j is the parent of p_i which in turn is the parent of p_h and $p_i \notin P^*_{\text{con}}$, then p_j and p_h belong to P^*_{con}. We now claim that p_j and p_h are also mutually visible. This follows easily from Figure 3.9. Since $\mathcal{R}'_i$ is a triangle with exactly one child, the location of the child, p_h must be as shown in Figure 3.9. This is because the edge shared between $\mathcal{R}_i$ and $\mathcal{R}_h$ must be the diagonal $[v, p_h]$. The segment $[v, p_i]$ cannot be a diagonal because then at least one vertex of Q on the opposite side of $[v, p_i]$ as p_j must be visible by p_i; but from the `Star-shaped Set Computation` algorithm, this vertex must be a part of $\mathcal{R}'_i$, which is a contradiction. Out of the two vertices of the diagonal $[v, p_h]$, the vertex p_h is chosen as the location of the child of p_i by the `Odd-numbered placement scheme`. Notice that $[p_i, p_h]$ is the shared edge between $\mathcal{R}_j$ and $\mathcal{R}_i$. Now it is easy to see that p_j and p_h are mutually visible.

Thus, any node $p_h \in P^*_{\text{con}}$ is connected to any node $p_k \in P^*_{\text{con}}$ in $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$ if p_k is a predecessor of p_h in $\mathcal{G}_{\text{nav-con}}(s, Q)$. If, on the other hand, p_k is not a predecessor of p_h and vice versa, then p_h and p_k must share a common predecessor in $\mathcal{G}_{\text{nav-con}}(s, Q)$, say p_l. If $p_l \in P^*_{\text{con}}$, then we are done, because p_k and p_h are both connected to p_l in $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$. If $p_l \notin P^*_{\text{con}}$, and p_l is not the root of $\mathcal{G}_{\text{nav-con}}(s, Q)$, then let p_m be the parent of p_l. Then $p_m \in P^*_{\text{con}}$ because the edge (p_l, p_m) of $\mathcal{G}_{\text{nav-con}}(s, Q)$ must contain one point belonging to P^*_{con}. Thus, p_h and p_k are both connected to p_m in $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$ since p_m is a common predecessor. On the other hand, if p_l is the root, then it cannot have a parent. If $p_l \notin P^*_{\text{con}}$, then this implies that p_l has exactly one child in $\mathcal{G}_{\text{nav-con}}(s, Q)$.

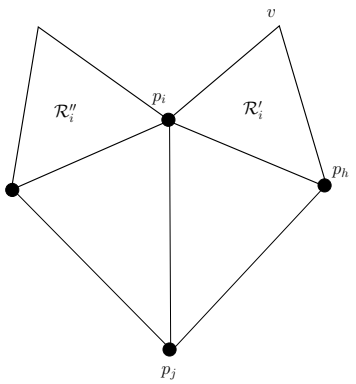


Figure 3.9: Illustration of the case when p_j is the parent of p_i and p_i is the parent of p_h and $p_i \notin P^*_{\text{con}}$.

This is because $\mathcal{R}'_l = \mathcal{R}''_l$ when p_l is the root and according to the condition in step (3) of the algorithm $|\operatorname{Ve}(\mathcal{R}'_l)| = 3$, which implies that p_l can have exactly one child. But this child must belong to P^*_{con} and this child is a common predecessor of p_h and p_k and hence connected to them in $\mathcal{G}_{\text{vis},Q}(P^*_{\text{con}})$. This completes the proof of statement (iv). □

3.4.3 Incremental algorithm for connected deployment

From the construction in the previous section, it is clear that if we can deploy the agents over the kernel point-set, P^*_{con}, then we will have solved the connected deployment problem requiring $\frac{n-1}{2}$ agents in the worst case. Our strategy is to incrementally construct the navigation graph $\mathcal{G}_{\text{nav-con}}(s, Q)$ and emulate the `Connected Kernel Computation Algorithm` via a *distributed* implementation.

We now describe a high-level motion coordination algorithm to implement the `Connected Kernel Computation Algorithm` in a multi-agent network. An agent is capable of identifying the children of the node of $\mathcal{G}_{\text{nav-con}}(s, Q)$ that it occupies by virtue of its sensing capabilities. We also assume that the agent can then order the children based on an *ordering scheme* described in the following section. We also assume that each agent is capable of moving along straight lines connecting neighboring vertices of the navigation graph $\mathcal{G}_{\text{nav-con}}(s, Q)$. We describe these abilities via two motion primitives `Move-to-Child Algorithm` and `Move-to-Parent Algorithm`. We postpone to the following section the description of the content of the memory of each agent and of the communication being exchanged between agents in order to allow each agent to execute

`Move-to-Child Algorithm` and `Move-to-Parent Algorithm`. For now, it suffices to say that each agent maintains in memory a variable `status`, with possible values {`UNASSIGNED`, `ASSIGNED`}, initialized to `UNASSIGNED`. The agent sets its `status` to `ASSIGNED` when it identifies a vertex of the environment as a node of $\mathcal{G}_{\text{nav-con}}(s, Q)$ that belongs to P^*_{con}. This vertex is the position where the agent will tentatively position itself. Agents will communicate and process information only when placed at the nodes of the navigation graph. The agent also stores the location of the parent of the node that it occupies and also the last node it has visited, p_{last}.

The `Connected Depth-first Deployment` algorithm is as follows:

Algorithm: `Connected Depth-first Deployment`
Input: Simply connected polygon Q, $s \in \text{Ve}(Q)$, k agents located at s
For every `PROCESS` interval for agent j located at node p_i of $\mathcal{G}_{\text{nav-con}}(s, Q)$ **do**

(1) **If** `ASSIGNED` message received from another agent at the same node with same parent and with UID lower than j, **then**

(2) `status` := `ASSIGNED`; `Stay` at current node; **Return**

(3) **If** `ASSIGNED` message received from another agent at the same node with same parent and with UID greater than j, **then**

(4) `status` := `UNASSIGNED`; `Move-to-Parent Algorithm`; **Return**
/* Therefore, no `ASSIGNED` message received from another agent at the same node with same parent */

(5) **If** `status` = `ASSIGNED` **then** `Stay` at current node; **Return**

(6) Compute $\mathcal{R}_i$ and the locations of all the children of p_i and order them.

(7) **If** ($|\text{Ve}(\mathcal{R}'_i)| = 3$) **AND** ($|\text{Ve}(\mathcal{R}''_i)| = 3$) **AND** (there exists exactly one child of p_i in each of $\mathcal{R}'_i$ and $\mathcal{R}''_i$) **AND** (both the children are occupied by at least one agent with `ASSIGNED` status and with parent p_i) **then**

(8) `status` := `UNASSIGNED`; `Move-to-Parent Algorithm`; **Return**

(9) **else if** (all children of p_i are occupied by agents with `ASSIGNED` status with parent p_i) **OR** (p_{last} is either the last child or the last child such that no agent with `ASSIGNED` status and with parent p_i occupies it) **OR** (no children exist) **then**

(10) `status` := `ASSIGNED`; `Stay` at current node; **Return**

(11) **else**

(12) `status` := `UNASSIGNED`;

(13) `Move-to-Child Algorithm` towards next unassigned child in ordering; **Return**

We are now ready to state the main result of this section.

Theorem 3.4.5 (Convergence and Run Time Analysis)**.** *Given a simply connected polygon Q with n vertices, assume that k agents are initially collocated at $s \in \mathrm{Ve}(Q)$. Then the following statements hold:*

(i) there exists a finite time t^*, such that for all times greater than t^* there is at least one agent on $\min\{|P^*_{\mathrm{con}}|, k\}$ points of P^*_{con};

(ii) if $k \geq \frac{n-1}{2}$, then the connected deployment problem is solved in finite time by the algorithm;

(iii) assuming unit speed for any agent and bounded diameter of Q, the time taken for task completion is $t^* \in O(n)$.

Proof. We begin by proving statement (i). We prove that in finite time there exists at least one agent with `ASSIGNED` status on $\min\{|P^*_{\mathrm{con}}|, k\}$ points of P^*_{con}. From the `Connected Depth-first Deployment`, it follows that once a node of the navigation graph is occupied by an agent with `ASSIGNED` status, it continues to be occupied by at least one agent with `ASSIGNED` status for all future times. To prove our claim, we first show that the `status` of an agent on a node, say p_i, can be `ASSIGNED` only if the node belongs to P^*_{con}. Let us assume that at some time t, all the agents with `ASSIGNED` status are located on nodes belonging to P^*_{con}. We show that the next agent that acquires the `ASSIGNED` must be located on a node belonging to P^*_{con}. An agent can acquire an `ASSIGNED` status either from another agent with `ASSIGNED` status or at step (10) of the algorithm. If it is the former, then we are done because by assumption all agents with `ASSIGNED` status occupy nodes belonging to P^*_{con}. For the latter case, we have to show that if the execution of the algorithm reaches step (10), then the agent must be on a point in P^*_{con}. For step (10) to be executed, the conditions in step (7) must not hold and at least one of the conditions in step (9) must be true. If all children contain agents with `ASSIGNED` status and with parent p_i, then all children belong to P^*_{con}. In the `Connected Kernel Computation Algorithm`, at any stage of the execution of the algorithm, the point-set P^*_{con} is a subset of P^*_{mark}. Thus, in step (1) of the `Connected Kernel`

`Computation Algorithm`, we can also work with a node whose all children belong to P^*_{con}. Thus, p_i is one such node. Since the condition in step (7) is not true, it implies that the condition in step (3) of the `Connected Kernel Computation Algorithm` is also not true. This proves that $p_i \in P^*_{\text{con}}$. Now, if p_{last} is the last child or the last child with no agent occupying it whose status is `ASSIGNED` and whose parent is p_i, then it implies that the agent at node p_i has visited all the children which do not have any agent with `ASSIGNED` status on them. Now, this implies that for all such children the condition in step (7) must have been true for the agent to execute `Move-to-Parent Algorithm` and move to p_i. Hence that child cannot belong to P^*_{con}. But from the proof of Proposition 3.4.4 (iii), every edge of the navigation graph contains at least one element of P^*_{con}. Since at least one child of p_i does not belong to P^*_{con}, we have that $p_i \in P^*_{\text{con}}$. Finally, if p_i does not have any children, then p_i is a leaf of the navigation graph and clearly belongs to P^*_{con}. In the end to prove our claim, we must show that the first agent acquiring the `ASSIGNED` status must be at a node belonging to P^*_{con}. But, this can be easily checked by noticing the for this agent the condition in step (10) is true only if there are no children of the node. Again, a node having no children belongs to P^*_{con}. We have thus proven that an agent can acquire the `ASSIGNED` status only on a node belonging to P^*_{con}.

Now, steps 1-5 of `Connected Depth-first Deployment` imply that an agent with `ASSIGNED` status stays at a node unless there is an agent with a higher UID collocated at the same node. It also follows that once such a node is occupied by an agent, it continues to be occupied by at least one agent for all future times. Therefore, the number of nodes containing agents with `ASSIGNED` status is non-decreasing. Since the number of nodes are finite, there exists a finite time after which the number of nodes containing agents with `ASSIGNED` status is constant. If this constant is equal to $|P^*_{\text{con}}|$, then we are done. On the other hand if this constant is equal to k, then again we are done. But, if this constant is less than both k and $|P^*_{\text{con}}|$, then this implies that there exists an agent that has `UNASSIGNED` status for all future times. Also notice that at any node occupied by an agent with `ASSIGNED` status, the UID of the agent that occupies it is nondecreasing. Since the number of agents are finite, there exists a finite time after which the maximum UID of an agent with `ASSIGNED` status at a node is constant. Therefore, after a finite time there will exist an agent that has `UNASSIGNED` status for all future times and any node that it will visit either does not have an agent with `ASSIGNED` status on it, or the UID of the agent with `ASSIGNED` status will be greater

than its own. If this agent executes only `Move-to-Parent Algorithm` for future times, then it will reach the root in finite time. Notice that the nodes of the navigation graph occupied by agents with `ASSIGNED` status are filled in a bottom up fashion, i.e., if an agent with `ASSIGNED` status occupies a node, then all nodes belonging to the subtree of that node and to P^*_{con} must already be occupied by agents with `ASSIGNED` status. Thus, if an agent reaches the root and executes only `Move-to-Parent Algorithm`, then all nodes of the navigation graph belonging to P^*_{con} must be occupied by agents with `ASSIGNED` status; this is a contradiction. On the other hand, if the agent executes a `Move-to-Child Algorithm` from a parent to a child, then both the parent and the child must not be occupied by agents with `ASSIGNED` status. But again one of these nodes must belong to P^*_{con}. Therefore, either the agent does not return to the parent, in which case it implies that it has acquired the `ASSIGNED` status which is a contradiction, or it executed the `Move-to-Parent Algorithm` from the child to the parent and since no agent is located on the child with status `ASSIGNED`, we have that the agent will acquire the `ASSIGNED` status at the parent node which is again a contradiction.

Statement (ii) is a consequence of statement (i) and Proposition 3.4.4 (ii).

The worst case time is obtained when all agents move together along the same depth-first path in the navigation graph till all the agents are deployed. Since the diameter of Q is bounded, the time taken to move between adjacent nodes of the navigation graph is bounded. Also, the time taken to process the information and the communication delay are bounded according to our model in Section 3.2. The navigation graph has at most $n-2$ nodes; thus visiting all the nodes along a depth-first search takes $O(n)$ time. This proves statement (iii). □

3.4.4 Distributed information processing

In this section, we describe the various ingredients of the `Connected Depth-first Deployment` algorithm described in the previous section.

Communication

The information vector $\mathcal{I}$ of each agent to be broadcast consists of the logic variable {`status`}, assuming values in the sets {`ASSIGNED`, `UNASSIGNED`} and the relative location of the parent node.

This is broadcast along with the UID of the agent as described in Section 3.2.

Motion primitives

From Lemma 3.4.2 (ii), we know that adjacent nodes of the navigation graph are mutually visible. Therefore, moving between adjacent nodes consists of moving along a straight line from one point to another, possible due to the first order dynamics of the agents described in Section 3.2. This constitutes the `Move-to-Parent Algorithm` and `Move-to-Child Algorithm` where the difference in name is only an aid to understand whether the agent is moving from a node to its parent or to one of its children.

Ordering the children

Let m be any positive integer. Then the children of any node, say p_i, of $\mathcal{G}_{\text{nav-con}}(s, Q)$, can be ordered in the following way:

> **Ordering scheme:** Let c_i be the number of children of p_i. The children are first ordered in clockwise manner as seen from p_i. Then the children are re-ordered by performing a cyclic shift with the $(m \mod c_i) + 1$th child as the first.

Memory management

At any node, say p_i of the navigation graph, to execute the `Move-to-Parent Algorithm`, the agent must know the location of the parent. Since the partition and hence the navigation graph is constructed in a top-down manner, the location of the parent cannot be computed from p_i. Hence, the relative location of the parent must be stored in the memory. Next, to compute the set $\mathcal{R}_i$ and hence the children of p_i, an agent needs to remember the diagonals that will separate the set $\mathcal{R}_i$ from the $\mathcal{R}_j$ where p_j is the parent of p_i. To order the children, the agent needs to have access to the parameter, m, used for the ordering as described in the previous section. As described in the next paragraph, this parameter depends on the UID and also on the topology of the navigation graph. The agent needs to remember the last node it has visited to enable it to visit the nodes of the navigation graph in an ordered way. Finally, the agents needs to have access to

the variable `status`, since it is at times not obtained as a result of local computation but via local communication.

The above geographic and other required information is gathered and managed by the agents via the following state transition laws and communication protocols. At this time, we make full use of the computation, communication, and sensing abilities of the agents as described in Section 3.2.

(i) The memory content $\mathcal{M}$ of each agent is a $6-$tuple $(p_{\text{parent}}, p_{\text{last}}, g_1, g_2, \texttt{status}, \text{ID}^*)$, where p_{parent} is an ordered list of points in $\mathbb{R}^2$, p_{last} is a point in $\mathbb{R}^2$, g_1 and g_2 are ordered list of elements belonging to $\mathbb{R}^2 \times \mathbb{R}^2$, and ID^* is an ordered list of positive integers denoting the parameter which dictates the ordering of the children of the corresponding nodes in p_{parent}. For any agent i, at time $t = 0$, $\mathcal{M}_i(0) = \{(a_i(0)), a_i(0), (a_i(0), a_i(0)), (a_i(0), a_i(0)), \texttt{UNASSIGNED}, i\}$.

During run time, $\mathcal{M}$ is updated to acquire and maintain the following meaning: p_{parent} is the list of *relative locations* of the predecessor nodes with respect agent's current node, p_{last} is the relative location of the last *node* visited by the agent, and g_1 and g_2 are the diagonals that are shared between the agents current cell(s) and the parent cell. This is accomplished as follows:

(ii) After an agent moves from a node p_i to a child node p_j located on diagonals described by vertices v_1', v_1'' and v_2', v_2'' via `Move-to-Child Algorithm`, its memory $\mathcal{M}$ is updated as follows: $p_i - p_j$ is added to the beginning of the list p_{parent}, $p_{\text{last}} := p_i - p_j$, $(v_1' - p_j, v_1'' - p_j)$ and $(v_2' - p_j, v_2'' - p_j)$ are added to the beginning of the lists g_1 and g_2 respectively, and $\lceil \text{ID}_1^*/c_i \rceil$ is added to the beginning of the list ID^*. Here ID_1^* refers to the first element of ID^*.

(iii) After an agent moves from a node p_j to the parent node p_i via `Move-to-Parent Algorithm`, its memory $\mathcal{M}$ is updated as follows: the first elements of p_{parent}, g_1, g_2 and ID^* are deleted and $p_{\text{last}} := p_i = p_j$.

(iv) `status` is managed according to the `Connected Depth-first Deployment` algorithm described in Section 3.4.3.

The following lemma quantifies the memory and communication complexities involved in the distributed information processing scheme described above.

Lemma 3.4.6 (Memory and communication complexity)**.** *Let k be the total number of agents and the UIDs of the agents belong to $\{1, \ldots, k\}$ and let the diameter of Q be bounded. For any agent i, at any time t, the following properties hold:*

(i) The number of bits required to store $\mathcal{M}_i$ is $O(d \log k)$, where d is the depth of $\mathcal{G}_{\text{nav-con}}(s, Q)$; and

(ii) the number of bits in any broadcast message is $O(\log k)$.

Proof. The memory is a 6-tuple; the objects p_{parent}, g_1, g_2 and ID* are lists of length at most d and the objects p_{last} and `status` are of length one. Since the diameter of Q is bounded, the number of bits required to store the relative locations of the elements of p_{parent} with finite resolution is $O(1)$. The value of an element of ID* is bounded by the UID of the agent which is bounded by k. It, therefore, takes at most $\log k$ bits to store an element of ID*. Statement (i) follows directly. A broadcast message consists of the UID, `status` and p_{parent_1}. The former is bounded by k and the latter two are of constant size. Statement (ii) follows. □

3.4.5 Simulation results

In this section we present simulation results for the `Connected Depth-first Deployment` algorithm described in Section 3.4.3. The algorithms have been implemented in `MATLAB`. The environment, Q, the root s and the navigation graph $\mathcal{G}_{\text{nav-con}}(s, Q)$ are as shown in Figure 3.6. Note that Q is chosen to represent a typical floor plan. Figure 3.10 shows, from left to right, the evolution of the simulation. At the end of the simulation, the agents are located at the points P^*_{con}, which in this case coincides with nodes of the navigation graph. The yellow sets denote the star-shaped sets owned by the agents that are already `ASSIGNED`.

3.5 General deployment of agents

In this section, we present a distributed algorithm for deployment of agents *without constraining the visibility graph of the final configuration of the agents to have one connected component.* As in the previous section, we begin by identifying a set $P^* \in \ker^*(Q)$ which is the desired set of points for deployment. The classic Art Gallery Theorem [76] states that there exists a point set

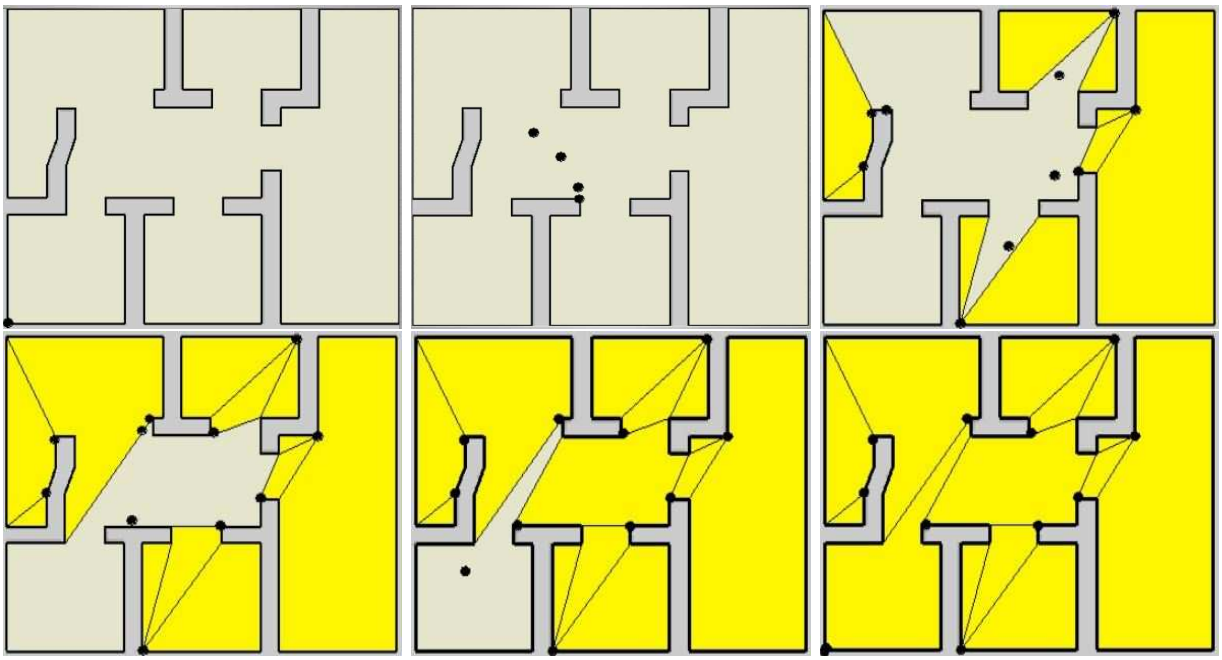


Figure 3.10: Execution of the `Connected Depth-first Deployment` algorithm over a prototypical floor plan. As forecasted by our analysis, the final agent configuration is connected in the visibility graph, and any point in the environment is visible to at least one agent.

satisfying the above properties with cardinality at most $\lfloor n/3 \rfloor$. In [100], a constructive algorithm based on coloring is given to compute the set. However, the algorithm in [100] requires centralized computation and is not amenable to a distributed implementation. In this section, we present a `Kernel Computation Algorithm` to compute P^* whose cardinality is at most $\lfloor n/3 \rfloor$, but which more amenable for a distributed implementation. We then present the `Depth-first Deployment` algorithm based on the `Kernel Computation Algorithm` to solve the deployment problem.

3.5.1 Navigation graph for deployment

In this section we first define the navigation graph for deployment. Unlike the navigation graph for connected deployment, the navigation graph in this section is not used in the `Kernel Computation Algorithm` algorithm but it will be an important ingredient of the *distributed implementation* of the `Kernel Computation Algorithm` via the `Depth-first Deployment` algorithm. We maintain the same organization of the section for the sake of consistency.

Definition 3.5.1 (Navigation graph)**.** *Given a simply connected polygonal environment, Q, $s \in$*

$\mathrm{Ve}(Q)$, *let* $\mathcal{R}$ *and* $\mathcal{P}$ *be the outputs of the* `Incremental Partition Algorithm`. *The* navigation graph, $\mathcal{G}_{\mathrm{nav}}(s, Q)$ *is the graph with* $\{v \in \mathrm{Ve}(Q) \cap \mathcal{R}_i \cap \mathcal{R}_j \mid \mathcal{R}_i,\ \mathcal{R}_j \in \mathcal{R},\ i \neq j\} \cup \{s\}$ *as the node set and an edge between two nodes* v' *and* v'' *if and only if* $v' = p_i$ *for some* i *and* $v'' \in \mathcal{R}_i \cap \mathcal{R}_j$ *for some* $\mathcal{R}_j$ *with* $v' \notin \mathcal{R}_j$.

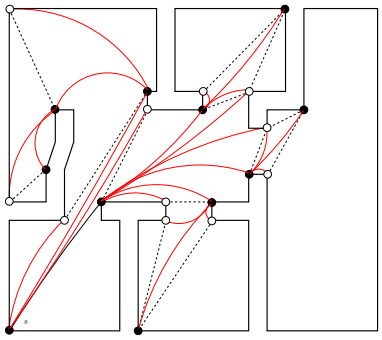


Figure 3.11: Illustration of the navigation graph, $\mathcal{G}_{\mathrm{nav}}(s, Q)$. The set of points represented by the black and white discs are the nodes of the graph; the red arcs represent the edges. The black discs represent the set of points in $\mathcal{P}$.

Figure 3.11 is an illustration of an example navigation graph. The following lemma summarizes the properties of $\mathcal{G}_{\mathrm{nav}}(s, Q)$.

Lemma 3.5.2 (Properties of the navigation graph)**.** *The graph,* $\mathcal{G}_{\mathrm{nav}}(s, Q)$*, satisfies the following properties:*

(i) $\mathcal{G}_{\mathrm{nav}}(s, Q)$ *is not necessarily a tree, i.e., it can contain cycles;*

(ii) adjacent nodes of the graph are mutually visible to each other.

Proof. Please refer to Figure 3.11 for an example of a scenario when $\mathcal{G}_{\mathrm{nav}}(s, Q)$ contains a cycle. Statement (ii) follows from the construction of $\mathcal{R}$. □

3.5.2 A sparse kernel point-set

We begin by introducing some additional useful notions. Given a triangulation $\mathcal{T}_Q$ of an environment Q, it is possible to define a graph where a node represents a triangle and an edge represents the fact that the corresponding triangles share a common diagonal; see [84] and Figure 3.12. For simply connected polygons, this graph is a tree. Let us now designate a node r that has at most

two neighbors as the root of the tree; this choice is arbitrary. We thus obtain a directed tree that is binary[2] and we denote it by $\mathcal{G}_{\triangle}(\mathcal{T}_Q, r)$. The node set and edge set are denoted by $\mathcal{N}(\mathcal{T}_Q)$ and $\mathcal{E}(\mathcal{T}_Q)$ respectively. A *fork* of $\mathcal{G}_{\triangle}(\mathcal{T}_Q, r)$ is any node having two children. A *chain* of $\mathcal{G}_{\triangle}(\mathcal{T}_Q, r)$ is a connected subgraph of $\mathcal{G}_{\triangle}(\mathcal{T}_Q, r)$ with the property that every node has at most one child in the subgraph. Before presenting the `Kernel Computation Algorithm`, we introduce three useful

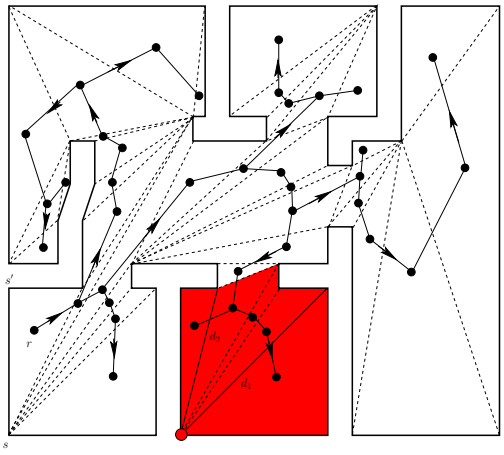


Figure 3.12: A simply connected environment, Q, in the shape of a typical floor plan and its triangulation by means of diagonals. A triangle r and the corresponding directed tree $\mathcal{G}_{\triangle}(\mathcal{T}_Q, r)$. The edges of $\mathcal{G}_{\triangle}(\mathcal{T}_Q, r)$ are directed away from the root. The shaded region on the right shows the set $\mathcal{N}_{\text{cvr}}(v, \text{ad}(v))$, where v is depicted by the red ball and $\text{ad}(v) = \{d_1, d_2\}$ where d_1, d_2 are the diagonals shown by the solid lines.

notions. First, we associate to each vertex in Q a set of diagonals of Q. Formally, given a vertex v, let $\text{ad}(v)$ be the set of *diagonals of Q associated with v*. To begin with, we set $\text{ad}(v) := \emptyset$ for all $v \in \text{Ve}(Q)$. As the `Kernel Computation Algorithm` populates the associated sets of diagonals, we will require that each diagonal to be added to a set $\text{ad}(v)$ must (1) contain v and (2) be the edge of a triangle in $\mathcal{T}_Q$. Second, we provide an algorithmic way of computing certain triangles visible by a vertex and containing the diagonals associated with that vertex. We call these triangles "covered."

Definition 3.5.3 (Covered triangles)**.** *Given a vertex v with a non-empty associated diagonal set* $\text{ad}(v)$*, the* set of triangles covered by v using $\text{ad}(v)$*, denoted by* $\mathcal{N}_{\text{cvr}}(v, \text{ad}(v))$*, is the set of triangles $\mathcal{N} \in \mathcal{N}(\mathcal{T}_Q)$ with the two following properties: (1) $\mathcal{N}$ contains v, and (2) either $\mathcal{N}$ or one of its successors contains a diagonal in* $\text{ad}(v)$*.*

[2] A directed tree where every node has at most two outgoing edges.

Third, given a set of vertices with non-empty associated diagonal set, we characterize some sets of triangles that are not covered by the vertices.

Definition 3.5.4 (Removable sets of triangles)**.** *Given Q and a triangulation $\mathcal{T}_Q$ of it, let $\mathcal{G}_\triangle(\mathcal{T}_Q, r)$ be the corresponding directed tree with r as the root. Let P be a set of vertices with non-empty associated diagonal sets* $\mathrm{ad}(p)$*, for $p \in P$.*

(i) A removable fork of P *is a fork $\mathcal{N}$ with the property that, in each of its two branches, the successors of $\mathcal{N}$ that do not belong to $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$ together with the fork form one of the two structures in Figure 3.13.*

(a) (b)

Figure 3.13: Illustration of possible cases of an occurrence of a removable fork. In case (a), all successors of $\mathcal{N}$ in the left branch belong to $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$ and only the child of $\mathcal{N}$ in the right branch does not belong to $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$. In case (b), in both branches of $\mathcal{N}$, only the children of $\mathcal{N}$ do not belong to $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$.

(ii) A removable chain of P *is a chain of length greater than or equal to* 3 *satisfying the following properties: (a) all triangles in the chain share a common vertex; (b) a fork may be present only at the beginning or at the end of the chain; (c) none of the triangles of the chain belong to the set $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$, except possibly for the triangle at the beginning of the chain; (d) if the triangle at the beginning of the chain is not a fork, then it must not belong to the set $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$; and (e) all triangles that are successors of the last triangle of the chain belong to the set $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$.*

(iii) A removable stub of P *is a chain of length less than or equal to* 2 *satisfying the following properties: (a) the first triangle is the root r; (b) none of the triangles of the chain belong to the set $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$; (c) all triangles that are successors of the last triangle of the chain belong to the set $\cup_{p\in P}\mathcal{N}_{\mathrm{cvr}}(p, \mathrm{ad}(p))$; and (d) if the root is a fork, then the length of the chain must be* 1*.*

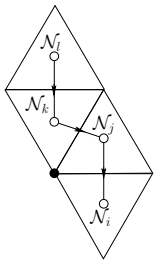


Figure 3.14: Illustration of an example of a removable chain formed by the triangles $\mathcal{N}_i$, $\mathcal{N}_j$ and $\mathcal{N}_k$: (a) The three triangles share a common vertex (black disc.); (b) $\mathcal{N}_j$ is not a fork; (c) none of the triangles belong to $\cup_{p\in P}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$; (d) $\mathcal{N}_k$ is not a fork and it does not belong to $\cup_{p\in P}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$; and (e) all triangles that are successors of $\mathcal{N}_i$ belong to $\cup_{p\in P}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$.

After these three preliminary definitions, we are now ready to design a procedure to compute a set of points P^* and associated diagonal sets by covering all triangles in $\mathcal{T}_Q$.

Algorithm: `Kernel Computation Algorithm`

Input: Directed tree $\mathcal{G}_\triangle(\mathcal{T}_Q, r)$ dual to the triangulation $\mathcal{T}_Q$ of polygon Q with root r

Initialization: $P^* := \emptyset$ and $\text{ad}(p) := \emptyset$ for all $p \in \text{Ve}(Q)$

While a removable fork, a removable chain or a removable stub for P^* exists **do**

`R1`: For a removable fork, define p and e as described in Figure 3.15.

Update $P^* := P^* \cup \{p\}$ and $\text{ad}(p) := \text{ad}(p) \cup e$.

`R2`: For a removable chain, let p be the vertex common to all triangles in the chain.

Set e to be the set containing the diagonal separating the last triangle in the chain from its predecessor.

Update $P^* := P^* \cup \{p\}$ and $\text{ad}(p) := \text{ad}(p) \cup e$.

`R3`: For a removable stub, place p on a vertex common to r and its child.

Set e to be the set containing the diagonal which the root shares with its child.

Update $P^* := P^* \cup \{p\}$ and $\text{ad}(p) := \text{ad}(p) \cup e$.

Return: P^*

Please refer to Figure 3.16 for an example of a point-set P^* obtained via the `Kernel Computation Algorithm`.

Remarks 3.5.5. *(i) The rules can be applied to different portions of the $\mathcal{G}_\triangle(\mathcal{T}_Q, r)$ simultaneously. This property will be useful in designing an* asynchronous *distributed implementation of the above algorithm later.*

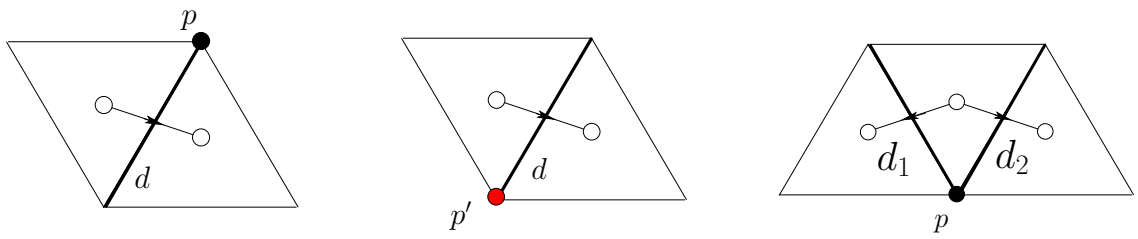


Figure 3.15: Illustration of placement rules for a removable fork. Case (a) illustrates the scenario when the point of $\mathcal{N}$ common to both its children does not belong to P^*. Let p be as shown in the figure and $e := \{d\}$. Case (b) depicts the situation when there exists a point $p' \in P^*$. Let $p = p'$ set $e := \{d\}$. In case (c), let point p be as shown and let $e := \{d_1, d_2\}$.

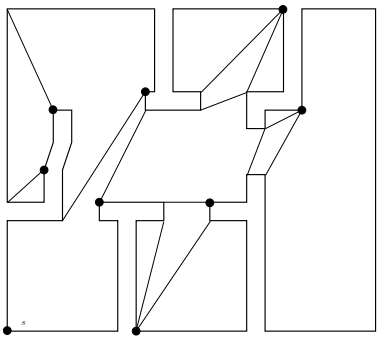


Figure 3.16: An illustration of a polygon Q and $s \in \text{Ve}(Q)$ the corresponding point-set P^*.

(ii) Rule `R3` *can be invoked at most once since there is only one root. Also, when invoked, it is the last rule to be done so.*

□

The following important theorem characterizes the set P^* obtained as a result of the above procedure.

Theorem 3.5.6 (Properties of the `Kernel Computation Algorithm`)**.** *Given a simply connected polygon Q with n vertices, a triangulation $\mathcal{T}_Q$ of Q, and a triangle $r \in \mathcal{T}_Q$ with at most two adjacent triangles, compute the point set P^* according to the* `Kernel Computation Algorithm` *invoked with input $\mathcal{G}_\triangle(\mathcal{T}_Q, r)$. Then the following properties hold:*

(i) The `Kernel Computation Algorithm` *invokes at most $(n-2)$ applications of the rules* `R1`*,* `R2` *and* `R3` *and its output P^* is unique up to the invocation of rule* `R3` *regardless of the sequence in which the earlier rules are applied;*

(ii) $P^* \in \ker^*(Q)$*;*

(iii) $|P^*| \leq n/3$.

Proof. We begin by proving statement (i). Let P denote P^* at any intermediate stage of the execution of the algorithm. To apply any of the rules `R1`, `R2` or `R3`, there must exist a removable fork, chain or stub respectively. In each of the three cases, there must exist at least a triangle, say $\mathcal{N}_j$, that does not belong to $\cup_{p\in P} \mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$. After the application of any rule, the triangle $\mathcal{N}_j$ belongs to $\cup_{p\in P} \mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$, where P is updated as in the `Kernel Computation Algorithm`. Thus, $|\cup_{p\in P^*} \mathcal{N}_{\text{cvr}}(p, \text{ad}(p))|$ increases strictly after the application of any rule. Therefore, the number of times the rules `R1`, `R2` and `R3` can be applied is bounded by the total number of triangles, which is $n - 2$.

To prove statement (ii), we show that $\cup_{p\in P^*} \mathcal{N}_{\text{cvr}}(p, \text{ad}(p)) = \mathcal{N}(\mathcal{T}_Q)$. We argue by contradiction. Let $\mathcal{N}_j \in \mathcal{N}(\mathcal{T}_Q)$ be the triangle at the greatest depth which does not belong to the set $\cup_{p\in P^*} \mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$. If $\mathcal{N}_j$ does not have a parent, then it is the root. Thus, it is a removable stub and hence `R3` can be applied which is a contradiction. Now let $\mathcal{N}_j$ have a parent, say $\mathcal{N}_i$. Now if $\mathcal{N}_i$ is a fork, then all grand-children of $\mathcal{N}_i$ belong to $\cup_{p\in P^*} \mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$ because by assumption

$\mathcal{N}_j$ is the triangle at the greatest depth that does not belong to the set. Then, $\mathcal{N}_i$ with $\mathcal{N}_j$ must form one of the structures shown in Figure 3.13. Thus, rule `R1` can be applied and again we have a contradiction. If on the other hand, $\mathcal{N}_i$ is not a fork, then we claim that it does not belong to $\cup_{p\in P^*}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$. If it did, then it was part of either a removable fork, removable chain or a removable stub. But, in all the three cases, it is a requirement that all triangles that are successors of the lower most triangles of the three structures belong to the set $\cup_{p\in P}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$, where P is an intermediate value of P^*. But since $\mathcal{N}_j$ does not belong to $\cup_{p\in P}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$, this is a contradiction. Thus, $\mathcal{N}_i$ also does not belong to $\cup_{p\in P^*}\mathcal{N}_{\text{cvr}}(p, \text{ad}(p))$. If now $\mathcal{N}_i$ does not have a parent, then $\mathcal{N}_i$ is the root and again $(\mathcal{N}_i, \mathcal{N}_j)$ forms a removable stub and `R3` can be applied which is a contradiction. If $\mathcal{N}_i$ has a parent, say $\mathcal{N}_h$, then $(\mathcal{N}_h, \mathcal{N}_i, \mathcal{N}_j)$ is a chain of length 3, and any 3 triangles in a chain share a common vertex. Thus, $(\mathcal{N}_h, \mathcal{N}_i, \mathcal{N}_j)$ is a removable chain and hence `R2` can be applied, which is a contradiction.

To prove statement (iii), let us introduce some useful notation. Given Q, a triangulation $\mathcal{T}_Q$ and a root r, let $P^*(Q, \mathcal{T}_Q, r)$ denote the kernel configuration obtained upon execution of the `Kernel Computation Algorithm`. If Q' is any simply connected polygonal subset of Q with every edge either an edge of Q or an edge of one of the triangles in the triangulation, $\mathcal{T}_Q$, then let $\mathcal{T}_{Q|Q'}$ denote the triangulation $\mathcal{T}_Q$ restricted to Q'.

The proof is by induction. We first show that the result holds true for any polygon with 5 or less vertices. If $n = 3$, then Q is a triangle. We apply `R3` and $|P^*(Q, \mathcal{T}_Q, r)| = 1 = n/3$ and we are done. If $n = 4$, then $\mathcal{G}_\triangle(\mathcal{T}_Q, r)$ is a chain of length 2 and also a removable stub of length 2. Thus, $|P^*(Q, \mathcal{T}_Q, r)| = 1 \leq n/3$. If $n = 5$, then $|\mathcal{T}_Q| = n - 2 = 3$. If the root triangle r has exactly one child, then the 3 triangles form a removable chain and $P^*(Q, \mathcal{T}_Q, r)$ consists of a single point placed at the common vertex of the triangles according to rule `R2`. Thus $|P^*(Q, \mathcal{T}_Q, r)| = 1 \leq n/3$. If r is a fork with two children, then the three triangles form a removable fork as shown in Figure 3.13 (c). In this case, $P^*(Q, \mathcal{T}_Q, r)$ consists of a single point placed according to rule `R1` in Figure 3.15 (c). Again, $|P^*(Q, \mathcal{T}_Q, r)| = 1 \leq n/3$. This proves the result for all polygons with 5 or less vertices.

Now, let us assume that the statement is true for all polygons with less than or equal to m vertices, where $m \geq 5$. Let Q be a polygon with $m + 1$ vertices and let $\mathcal{T}_Q$ be a triangulation of it. We now show that statement (iii) is also true for this polygon. Note that from statement (i)

of this lemma, we know that $P^*(Q, \mathcal{T}_Q, r)$ is unique and thus we can begin to apply the rules of the `Kernel Computation Algorithm` from any suitable portion of $\mathcal{G}_\triangle(\mathcal{T}_Q, r)$. If there are no forks present, let $\mathcal{N}_i$ be a leaf at the largest depth. If a fork is present, let us choose one with the largest depth. Now, let $\mathcal{N}_i$ be a successor of this fork which is at the largest depth. Let $\mathcal{N}_j$ be the parent of $\mathcal{N}_i$.

If $\mathcal{N}_j$ is a fork, then let $\mathcal{N}_k$ be the other child of $\mathcal{N}_j$. Now, $\mathcal{N}_k$ cannot have any successor since otherwise $\mathcal{N}_i$ will not be the leaf at the greatest depth. Thus, $\mathcal{N}_i$, $\mathcal{N}_j$ and $\mathcal{N}_k$ form a removable fork as in Figure 3.17. According to rule `R1` of the `Kernel Computation Algorithm`, let p be defined as in Figure 3.15 (c). Thus, $\mathcal{N}_{\text{cvr}}(p, \text{ad}(p)) = \{\mathcal{N}_i, \mathcal{N}_j, \mathcal{N}_k\}$. Let Q' be the polygon obtained after deleting the 3 triangles having $m-2$ vertices. Let $\mathcal{N}_l$ be the parent of $\mathcal{N}_j$. It must exist because Q consists of at least 4 triangles. But, then $P^*(Q, \mathcal{T}_Q, r) = P^*(Q', \mathcal{T}_{Q|Q'}, r) \cup \{p\}$. This implies that $|P^*(Q, \mathcal{T}_Q, r)| = |P^*(Q', \mathcal{T}_{Q|Q'}, r)| + 1$. From the induction hypothesis, we know that $|P^*(Q', \mathcal{T}_{Q|Q'}, r)| \leq (m-2)/3$. It follows that $|P^*(Q, \mathcal{T}_Q, r)| \leq (m-2)/3 + 1 = (m+1)/3$.

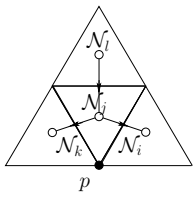
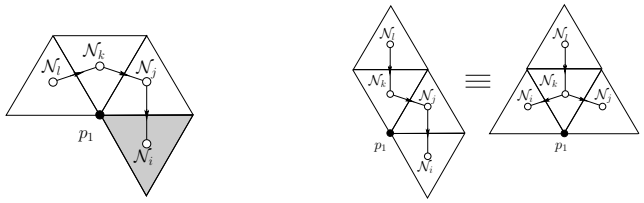


Figure 3.17: Illustration of the case when $\mathcal{N}_j$ is a fork.

If $\mathcal{N}_j$ is not a fork, then since there are at least 4 triangles, $\mathcal{N}_j$ has a parent, say $\mathcal{N}_k$. Then, $\mathcal{N}_i$, $\mathcal{N}_j$ and $\mathcal{N}_k$ form a removable chain and there exists $p_1 \in P^*(Q, \mathcal{T}_Q, r)$ at the vertex common to the three triangles. If $\mathcal{N}_k$ is not a fork, then let $\mathcal{N}_l$ be its parent. Again $\mathcal{N}_l$ exists because there are at least 4 triangles in Q. Then the cases illustrated in Figure 3.18 (a) and Figure 3.18 (b) (left) are possible. For the case in Figure 3.18 (a), let Q' be the polygon obtained after deleting $\mathcal{N}_i$. Then $P^*(Q, \mathcal{T}_Q, r) = P^*(Q', \mathcal{T}_{Q|Q'}, r)$. This is because $\mathcal{N}_j$, $\mathcal{N}_k$, $\mathcal{N}_l$ form a removable chain of Q' again; so there exists $p_1 \in P^*$ at the vertex common to the three triangles; the placement of points in the rest of the polygon is therefore unaffected. But $|\,\text{Ve}(Q')| = |\,\text{Ve}(Q)| - 1 = m$; thus by the induction hypothesis $|P^*(Q', \mathcal{T}_{Q|Q'}, r)| \leq m/3$. Therefore, $|P^*(Q, \mathcal{T}_Q, r)| \leq m/3 \leq (m+1)/3$ and

we are done. Also, Figure 3.18 (b) (left) is equivalent to Figure 3.18 (b) (right) upon rearranging as shown. But, this is in turn equivalent to the case illustrated in Figure 3.17.

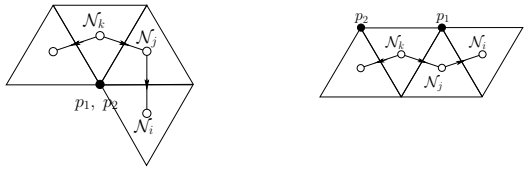


(a) (b)

Figure 3.18: Illustration of the possible cases when $\mathcal{N}_j$ and $\mathcal{N}_k$ are both not forks.

If $\mathcal{N}_k$ is a fork, then the other branch of $\mathcal{N}_k$ can have either one or two children. This is because by assumption, $\mathcal{N}_k$ is the fork at the greatest depth and $\mathcal{N}_i$ is its child at the greatest depth.

Let us first look at the case when the other branch of $\mathcal{N}_k$ has one triangle. Then the cases depicted in Figure 3.19 are possible. Notice that $(\mathcal{N}_k, \mathcal{N}_j, \mathcal{N}_i)$ is a removable chain; hence there exists $p_1 \in P^*$ at the vertex common to the three triangles. Now, $\mathcal{N}_k$ together with its child in the other branch is a removable fork; therefore there exists $p_2 \in P^*$ at a vertex governed by the placement rule `R1` in the `Kernel Computation Algorithm`. In case (a), p_1 and p_2 coincide. In case (a), delete $\mathcal{N}_i$ to obtain Q'. Then the result follows by arguing along the lines for the case in Figure 3.18 (a). For the case in Figure 3.19 (b), if $\mathcal{N}_k$ does not have a parent, then we have $m + 1 = 6$ and $|P^*(Q, \mathcal{T}_Q, r)| = 2 = 6/3$ and we are done. Let us now assume that $\mathcal{N}_k$ has a grand-parent as well as a sibling. The other remaining cases can easily be verified by the reader. Let $\mathcal{N}_l$ be the parent of $\mathcal{N}_k$ as shown in Figure 3.20. Then, two possible cases might arise: (i) the



(a) (b)

Figure 3.19: Illustration of all possible cases when $\mathcal{N}_j$ is not a fork, $\mathcal{N}_k$ is a fork and has exactly one triangle in the second branch.

triangle $\mathcal{N}_l$ together with the sibling of $\mathcal{N}_k$ forms a removable fork; and (ii) the triangle $\mathcal{N}_l$ is part of a removable chain.

Let us first look at case (i). Then, as per rule `R1` of the `Kernel Computation Algorithm`, a point will be placed to coincide with p_2. Thus, the third vertex of the $\mathcal{N}_l$ will not contain any point belonging to $P^*(Q, \mathcal{T}_Q, r)$. Therefore, we can decompose Q as shown in Figure 3.20 (top). The decomposed figure consists of four components as shown. The top left component is the polygon Q containing $\mathcal{N}_l$ and all its successors, say Q'. The bottom left component consists of two triangles as shown. The top right component consists of $\mathcal{N}_l$ and $\mathcal{N}_k$ and all successors of $\mathcal{N}_l$, say Q''. The bottom right component consists of the two triangles as shown. If we apply the `Kernel Computation Algorithm` to Q', then we claim that $P^*(Q', \mathcal{T}_{Q|Q'}, \mathcal{N}_l) = P^*(Q, \mathcal{T}_Q, r) \cap Q'$. This can be easily seen by noticing that the rules applied to the removable forks and chains in the part of Q that consists of successors of the sibling of $\mathcal{N}_k$ is not affected by the decomposition. What is affected is the fact that $\mathcal{N}_l$ together with the sibling of $\mathcal{N}_k$ forms a removable stub in Q' whereas in Q it is removable fork. However, even in the case of the removable, as per rule `R3` of the `Kernel Computation Algorithm`, we can place a point on one of the vertices of the diagonal that the root shares with its child; in this case we choose the point to coincide with p_2. Similarly, if we apply the `Kernel Computation Algorithm` to Q'', then we claim that $P^*(Q'', \mathcal{T}_{Q|Q''}, r) = P^*(Q', \mathcal{T}_Q, r) \cap Q''$. To see this notice that the vertex of $\mathcal{N}_l$ different from p_1 and p_2 does not contain a point belonging to $P^*(Q, \mathcal{T}_Q, r)$. Thus, the rules applied to the removable forks and chains in the part of Q that consists of predecessors of the sibling of $\mathcal{N}_l$ is affected only by p_1. But in Q'', the triangles $\mathcal{N}_k$, $\mathcal{N}_l$ and its predecessor form a removable chain. Thus, we place a point to coincide with p_1. The placement of points in the rest of Q'' is then identical to that in Q. Therefore we have that $|P^*(Q, \mathcal{T}_Q, r)| = |P^*(Q', \mathcal{T}_{Q|Q'}, \mathcal{N}_l)| + |P^*(Q'', \mathcal{T}_{Q|Q''}, r)|$. But, by our induction hypothesis, we have that $|P^*(Q', \mathcal{T}_{Q|Q'}, \mathcal{N}_l)| \leq |\operatorname{Ve}(Q')|/3$ and $|P^*(Q'', \mathcal{T}_{Q|Q''}, r)| \leq |\operatorname{Ve}(Q'')|/3$. But, $|\operatorname{Ve}(Q') + |\operatorname{Ve}(Q'')|| = |\operatorname{Ve}(Q)|$ since the three vertices of $\mathcal{N}_l$ are repeated in Q' and Q'' but also three vertices belonging to the bottom left and bottom right polygons are removed. This proves the result for case (i).

Let us now look at case (ii). Since $\mathcal{N}_l$ and the sibling of $\mathcal{N}_k$ do not form a removable fork, the point p_2 does not influence the placement of the rest of the points in Q. Two further cases are

possible. In the first case, point p_2 is placed according to rules `R1` or `R2` applied to removable forks or chains that in the part of Q consisting of $\mathcal{N}_l$ and its successors. In this case, we can decompose Q as shown in the Figure 3.20 (bottom) into three polygons. The left polygon is a triangle, the center polygon consists of $\mathcal{N}_j$ and all its predecessors together with the successors of $\mathcal{N}_l$, the right polygon again consists of one triangle as shown. Let the center polygon be Q'. If we apply the `Kernel Computation Algorithm` to Q', we obtain that $P^*(Q, \mathcal{T}_Q, r) = P^*(Q', \mathcal{T}_{Q|Q'}, r)$. This is because point p_2 is present by assumption. Point p_1 will be present since $\mathcal{N}_j$, $\mathcal{N}_k$ and $\mathcal{N}_l$ form a removable chain of Q'. But by the induction hypothesis, $|P^*(Q', \mathcal{T}_{Q|Q'}, r)| \leq |\operatorname{Ve}(Q')|/3 \leq |\operatorname{Ve}(Q)|/3$ and thus $|P^*(Q, \mathcal{T}_Q, r)| \leq |\operatorname{Ve}(Q)|/3$. In the second case, there are no removable forks or chains in the part of Q consisting of $\mathcal{N}_l$and its successors such that a point is placed at p_2. In this case, we may decompose Q into three polygons. The left polygon is a triangle, the center polygon consists of $\mathcal{N}_k$ and its predecessors together with the successors of $\mathcal{N}_l$ and the right polygon consists of two triangles as shown. Let Q' be the center polygon. Since point p_2 does not belong to $P^*(Q', \mathcal{T}_{Q|Q'}, r)$ by assumption, we have that point p_1 must belong to $P^*(Q', \mathcal{T}_{Q|Q'}, r)$ because of the fact that $\mathcal{N}_l$ and $\mathcal{N}_k$ will form a removable fork. Therefore, $|P^*(Q', \mathcal{T}_{Q|Q'}, r)| = |P^*(Q, \mathcal{T}_Q, r)| - 1$ since p_2 is not a part of $P^*(Q', \mathcal{T}_{Q|Q'}, r)$. But, by the induction hypothesis $|P^*(Q', \mathcal{T}_{Q|Q'}, r)| \leq |\operatorname{Ve}(Q')|/3 \leq (|\operatorname{Ve}(Q)| - 3)/3 = |\operatorname{Ve}(Q)|/3 - 1$. Thus, $|P^*(Q, \mathcal{T}_Q, r)| \leq |\operatorname{Ve}(Q)|/3$. This completes the proof for the case when the other branch of $\mathcal{N}_k$ has one triangle.

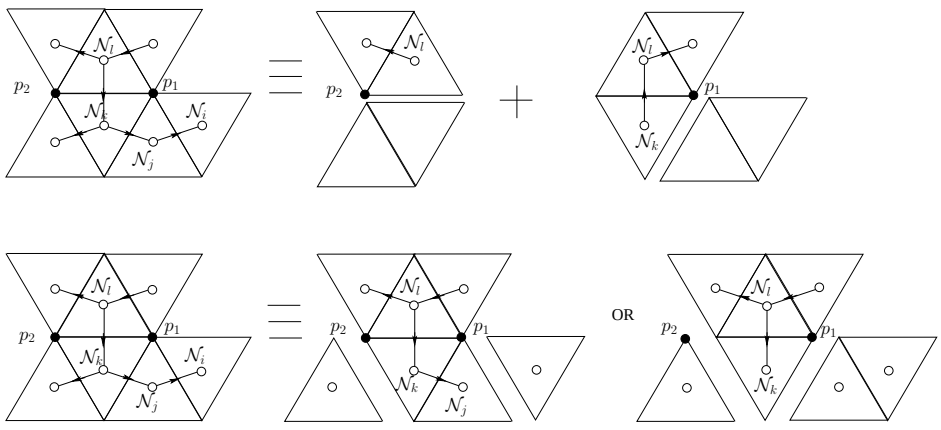
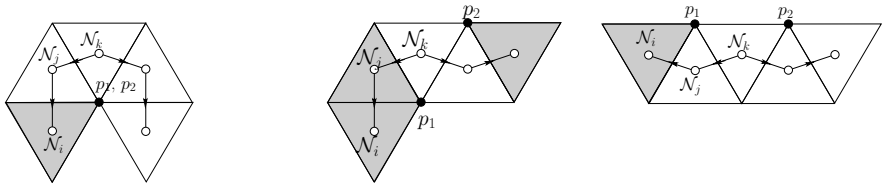


Figure 3.20: Illustration of the ideas used in the proof related to the case shown in Figure 3.19 (b).

Secondly, let us look at the case when the other branch of $\mathcal{N}_k$ has two triangles. Then the cases illustrated in Figure 3.21 are possible. Notice that $(\mathcal{N}_k, \mathcal{N}_j, \mathcal{N}_i)$ is a removable chain; hence

there exists $p_1 \in P^*$ at the vertex common to the three triangles. Similarly, the other branch of $\mathcal{N}_k$ also is a removable chain; again there exists $p_2 \in P^*$ at the common vertex. In case (a), p_1 and p_2 coincide. Again in case (a), delete $\mathcal{N}_i$ to obtain Q' as in Figure 3.19 (a). If it is case (b),



(a) (b) (c)

Figure 3.21: Illustration of all possible cases when $\mathcal{N}_j$ is not a fork, $\mathcal{N}_k$ is a fork and has exactly two triangles in the second branch.

delete $\mathcal{N}_i$, $\mathcal{N}_j$ and the other grand child of $\mathcal{N}_k$ to obtain Q'. Also, delete p_1. Notice that p_2 will be present in $P^*(Q', \mathcal{T}_{Q|Q'}, r)$ since the sibling of $\mathcal{N}_j$, $\mathcal{N}_k$ and its parent form a removable chain and p_2 is located at the common vertex. Since $|P^*(Q, \mathcal{T}_Q, r)| = |P^*(Q', \mathcal{T}_{Q|Q'}, r)| - 1$ and by our induction hypothesis, we have that $|P^*(Q', \mathcal{T}_{Q|Q'}, r)| \leq |\operatorname{Ve}(Q')|/3 = |\operatorname{Ve}(Q)|/3 - 1$, we obtain the desired result. If it is case (c) delete $\mathcal{N}_i$ to obtain Q' as in Figure 3.19 (b). This completes the proof of statement (iii). □

Remark 3.5.7. *By construction, we know that $s \in \mathcal{P}$. Let $p_i = s$ and now let us look at the set $\mathcal{R}_i$. Let s' be a vertex of Q that is adjacent to s in the clockwise direction. Clearly, s' belongs to $\mathcal{R}_i$. Now, let r be the triangle in $\mathcal{T}_Q(\mathcal{R}, \mathcal{P})$ containing the edge $[s, s']$. Thus, we can define the directed tree $\mathcal{G}_\triangle(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}), r)$, where the triangle r serves as the root; see Section 3.5.2.*

We now further characterize the point set P^* when $\mathcal{G}_\triangle(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}), r)$ is the input to the `Kernel Computation Algorithm`, where $\mathcal{R}$ and $\mathcal{P}$ are the outputs of the `Incremental Partition Algorithm`. But first, notice that from rule `R3` of the `Kernel Computation Algorithm`, we know that if a removable stub exists then a point p is added to P^*, where p belongs to a vertex common to r and its child. However, the exact vertex is not specified. Before characterizing P^*, let is resolve this ambiguity for the case when $\mathcal{G}_\triangle(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}))$ is the input to the `Kernel Computation Algorithm`.

The following lemma states that the root triangle r in $\mathcal{G}_\triangle(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}), r)$ always has s as one of the vertices that it shares with its children.

Lemma 3.5.8. *Let Q be a simply connected polygon and let $s \in \mathrm{Ve}(Q)$ be given. Let $\mathcal{R}$ be a partition constructed according the* `Incremental Partition Algorithm` *and let $\mathcal{P}$ be the corresponding set of kernel points. Let $\mathcal{G}_{\triangle}(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$ be the input to the* `Kernel Computation Algorithm` *algorithm. Then s belongs to the common edge between r and one of its children.*

Proof. s is a vertex of r; see Remark 3.5.7. Let s' and s'' be the other two vertices. Let $[s, s']$ be an edge of Q; again see Remark 3.5.7. We now claim that $[s, s'']$ is always a diagonal of Q. We know that $s \in \mathcal{P}$. Without loss of generality assume $p_1 = s$. Then $r \subseteq \mathcal{R}_1$. By construction every triangle in $\mathcal{R}_1$ that is a part of the triangulation $\mathcal{T}_Q(\mathcal{R},\mathcal{P})$ must contain a diagonal with $p_1 = s$ as one of the vertices of the diagonal. But $[s, s']$ is not a diagonal. Then $[s, s'']$ must be diagonal of Q. The lemma follows directly. □

Based on the above lemma we can now modify Rule `R3` of the `Kernel Computation Algorithm` as follows.

Algorithm: Modification of `R3` of `Kernel Computation Algorithm`
Input: $\mathcal{G}_{\triangle}(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$

`R3`: For a removable stub, place p on s
Set e to be the set containing the diagonal in r with s as one of the vertices.
Update $P^* := P^* \cup \{p\}$ and $\mathrm{ad}(p) := \mathrm{ad}(p) \cup e$.

In what follows, the set $P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$ will denote the output of the `Kernel Computation Algorithm` with the modified rule `R3`, given $\mathcal{G}_{\triangle}(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$ as the input. Finally, we are in a position to present the result that further characterizes the location of the points $P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$. The following lemma states that any element of $P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$ is a vertex of Q that belongs to the set of nodes of the navigation graph, $\mathcal{G}_{\mathrm{nav}}(s, Q)$. Thus, the deployment problem now reduces to deploying agents on a subset of the nodes of the navigation graph.

Lemma 3.5.9. *The set $P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$ is a subset of the vertex set of the navigation graph $\mathcal{G}_{\mathrm{nav}}(s, Q)$.*

Proof. We reason by contradiction. Let $u \in P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}), r)$ and let $u \notin \{v \in \mathrm{Ve}(Q) \cap \mathcal{R}_i \cap \mathcal{R}_j \mid \mathcal{R}_i,\ \mathcal{R}_j \in \mathcal{R},\ i \neq j\} \cup \{s\}$. Then $u \neq s$. Also, u belongs to exactly one element of $\mathcal{R}$, say $\mathcal{R}_i$. Since u belongs

only to $\mathcal{R}_i$, u can belong to at most two triangles in $\mathcal{T}_Q(\mathcal{R}, \mathcal{P})$. Those triangles cannot be forks. Thus, u must be placed according to rule `R2` in the `Kernel Computation Algorithm`. But then u must be common to at least three triangles which are part of a removable chain. This is a contradiction. □

3.5.3 Incremental algorithm for deployment

From the construction in the previous section, it is clear that if we can deploy the agents over the kernel point, P^*, then we will have solved the deployment problem requiring $\frac{n}{3}$ agents in the worst case. Our strategy is to incrementally construct the navigation graph $\mathcal{G}_{\text{nav}}(s, Q)$ and emulate the `Kernel Computation Algorithm` via a *distributed* implementation.

We now describe a high-level motion coordination algorithm to implement the `Kernel Computation Algorithm` in a multi-agent network. The agents move along straight lines between nodes of the navigation graph, $\mathcal{G}_{\text{nav}}(s, Q)$, and communicate and process information only when placed at the nodes. Notice that since $\mathcal{G}_{\text{nav}}(s, Q)$ is not a tree, there is no natural meaning of children and parent of a node of the graph. *For an agent located at a node v of $\mathcal{G}_{\text{nav}}(s, Q)$, the parent node is the last node visited that belonged to the set $\mathcal{P}$. The children of a node v that belongs to $\mathcal{P}$, say p_i, is the set $\{v' \in \text{Ve}(Q) \cap \mathcal{R}_i \cap \mathcal{R}_j \mid v' \notin \mathcal{R}_j\}$. There are no children of a node v that does not belong to $\mathcal{P}$.* Also recall that $\mathcal{P}$ can contain multiple copies of the same element, say $p_i = p_j$ but with distinct $\mathcal{R}_i$ and $\mathcal{R}_j$. As before, an agent is capable of identifying the children of the node of $\mathcal{G}_{\text{nav}}(s, Q)$ that it occupies by virtue of its sensing capabilities and can then order them based the *ordering scheme* described in the previous section. Each agent is capable of moving along straight lines connecting neighboring vertices of the navigation graph $\mathcal{G}_{\text{nav}}(s, Q)$ the via two motion primitives `Move-to-Child Algorithm` and `Move-to-Parent Algorithm`.

The description of the content of the memory of each agent and of the communication being exchanged between agents in order to allow each agent to execute `Move-to-Child Algorithm` and `Move-to-Parent Algorithm` is postponed to the following section. For now, it suffices to say that each agent maintains in memory a variable `status`, with possible values $\{$`UNASSIGNED`, `ASSIGNED`$\}$, initialized to `UNASSIGNED`. The agent sets its `status` to `ASSIGNED` when it identifies a vertex of the environment as a node of $\mathcal{G}_{\text{nav}}(s, Q)$ that belongs to P^*. This vertex is the position where the

agent will tentatively position itself. Since this vertex belongs to P^*, there exists a non-empty set of associated diagonals of the node. It also stores the contents of this set, denoted by $\mathrm{ad}_i(v)$. Here i denotes the UID of the agent and v is the current vertex occupied by the agent. At any node $v \in \mathcal{P}$, say $v = p_i$, an agent can identify the edges of the set $\mathcal{R}_i$ that are shared with the neighboring sets. At least one (possibly two) of these edges is the edge that contains p_i. Let us term the remaining edges as *gaps* of $\mathcal{R}_i$. The agent also maintains a list T_{diag}, whose elements are lists of positive integers themselves. The first element T_{diag_1} is a list of the number of triangles that are still uncovered beyond each of the gaps of $\mathcal{R}_i$, where 0 and 1 denote no uncovered triangles and one uncovered triangle, respectively and 2 denotes the fact that the agent has no information about the number of triangles that are uncovered; see Figure 3.22. If $\mathcal{R}_i$ has no gaps or if $v \notin \mathcal{P}$,

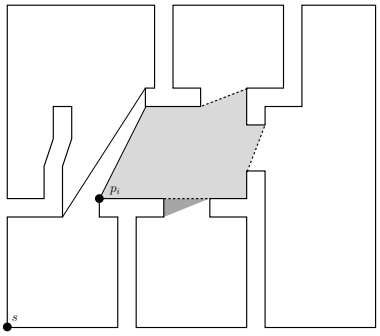


Figure 3.22: Illustration of the elements comprising the list T_{diag}. If the agent is located at p_i, with parent s, then the lightly shaded region represents $\mathcal{R}_i$. The *gaps* of $\mathcal{R}_i$ are depicted by the dashed lines. The list T_{diag_1} will comprise of three elements, each corresponding to one gap. Assuming that at a certain time there is one triangle that is uncovered beyond the first gap (shown by the darkly shaded triangle), and no triangles uncovered beyond the next two gaps, then $T_{\mathrm{diag}_1} = (1, 0, 0)$.

then $T_{\mathrm{diag}_1} = \emptyset$. The list T_{diag_2} corresponds to the parent of v and so on.

Algorithm: `Depth-first Deployment`

Input: Simply connected polygon Q, $s \in \mathrm{Ve}(Q)$, k agents located at s

For every `PROCESS` interval for agent j located at node v of $\mathcal{G}_{\mathrm{nav}}(s, Q)$ **do**

(1) **If** `ASSIGNED` message received from another agent l at the same node, **then**

(2) $\mathrm{ad}_j(v) := \mathrm{ad}_j(v) \cup \mathrm{ad}_l(v)$;

(3) **If** `status` = `UNASSIGNED`, **then**

(4) Let $p_i := v$; compute $\mathcal{R}_i$;let $\mathcal{R}_i := \mathcal{R}'_i \cup \mathcal{R}''_i$, where $\mathcal{R}'_i$ and $\mathcal{R}''_i$ are as in Remark 3.3.2.

(5) **If** ($\mathrm{ad}_j(v)$ contains diagonals in both $\mathcal{R}'_i$ and $\mathcal{R}''_i$), **then**

(6) `status` := `ASSIGNED`; `Stay` at current node; **Return**

(7) **If** (`status` = `ASSIGNED`) **AND** ($l < j$), **then**

(8) `Stay` at current node; **Return**

(9) **If** (`status` = `ASSIGNED`) **AND** ($l > j$), **then**

(10) `status` := `UNASSIGNED`; `Move-to-Parent Algorithm`; **Return**

(11) **If** `status` = `ASSIGNED`, **then** `Stay` at current node; **Return**
/* Therefore, status is `UNASSIGNED` */

(12) Let $p_i := v$; compute R_i; let $\mathcal{R}_i := \mathcal{R}'_i \cup \mathcal{R}''_i$, where $\mathcal{R}'_i$ and $\mathcal{R}''_i$ are as in Remark 3.3.2.

(13) Compute the children of p_i in $\mathcal{G}_{\mathrm{nav}}(s, Q)$ and order them using the ordering scheme.

(14) Acquire $\mathrm{ad}_l(p)$ from all agents l with `ASSIGNED` status located at the children p of p_i and update $T_{\mathrm{diag}\,1}$.

(15) **If** $p_{\mathrm{last}} \notin \mathcal{R}_i$ **then** /* the agent has arrived at p_i the first time */

(16) Set the elements of $T_{\mathrm{diag}\,1}$ that were not updated equal to 2

(17) **If** (all elements of $T_{\mathrm{diag}\,1}$ are less or equal to 1) **OR** ($T_{\mathrm{diag}\,1} = \emptyset$) **then**

(18) Compute the set $P_i^* := \{p \in P^*(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}), r) \mid \mathrm{ad}(p)$ contains a diagonal d in $\mathcal{R}_i$
where d is not an edge of $\mathcal{R}_i$ to which p_i belongs$\}$.

(19) **For** each $p \in P_i^*$, **do** compute $\mathrm{ad}_{\mathrm{tmp}}(p) := \{d \in (\mathrm{ad}(p) \cap \mathcal{R}_i), d$ is not an edge of $\mathcal{R}_i$ to which p_i belongs$\}$.

(20) **For** all children $p \in P_i^*$ occupied by agent l with `ASSIGNED` status, **do**

(21) **If** $\mathrm{ad}_l(p) \subsetneq \mathrm{ad}_{\mathrm{tmp}}(p)$ **then** send $\mathrm{ad}_{\mathrm{tmp}}(p)$ to agent l.

(22) Find first child $p \in P_i^*$ s.t. no agent with `ASSIGNED` status is located on it.

(23) **If** p does not exist, **then**

(24) **If** $p_i \in P_i^*$, **then**

(25) **If** no agent with `ASSIGNED` status present on p_i, **then**

(26) `status` := `ASSIGNED`; $\mathrm{ad}_j(p) := \mathrm{ad}_j(p) \cup \mathrm{ad}_{\mathrm{tmp}}(p)$; `Stay` at current node; **Return**

(27) $N := \mathrm{children}(p_i) \cup \{p_i\}$; $\mathrm{ad}_j(p) := \mathrm{ad}_j(p) \cup \mathrm{ad}_{\mathrm{tmp}}(p)$.

(28) **If** $p_i \notin P_i^*$, **then**

(29) $N := \text{children}(p_i)$.

(30) Let p_l be the parent of p_i.

(31) Let d' and d'' be the indices of the gaps in $\mathcal{R}_l$ corresponding to $R_i' \cap \mathcal{R}_l$ and $R_i'' \cap \mathcal{R}_l$, respectively.

(32) **If** $\cup_{w \in N} \mathcal{N}_{\text{cvr}}(w, \text{ad}_{\text{tmp}}(w))$ contain all triangles in $\mathcal{R}_i'$, **then** $T_{\text{diag}_2}(d') := 0$, **else** $T_{\text{diag}_2}(d') := 1$.

(33) **If** $\cup_{w \in N} \mathcal{N}_{\text{cvr}}(w, \text{ad}_{\text{tmp}}(w))$ contain all triangles in $\mathcal{R}_i''$, **then** $T_{\text{diag}_2}(d'') := 0$, **else** $T_{\text{diag}_2}(d'') := 1$.

(34) `status` := `UNASSIGNED`; `Move-to-Parent Algorithm`; **Return**
/* Therefore, p exists */

(35) `status` := `ASSIGNED`; $\text{ad}_j(p) := \text{ad}_{\text{tmp}}(p)$; `Move-to-Child Algorithm` towards p; **Return**
/* Therefore, at least one element of T_{diag_1} is greater than 1*/

(36) Compute next child that belongs to $\mathcal{P}$ such that $T_{\text{diag}_1}(d) > 1$ for at least one diagonal d that the child belongs to; `Move-to-Child Algorithm`; **Return**

Remark 3.5.10 (Update of T_{diag_1})**.** *Let p_i be the node of the navigation graph occupied by an agent j. The update T_{diag_1} takes place as follows:*

(i) Let p_j be a child of p_i that is occupied by another agent l with `ASSIGNED` status. If $\text{ad}_l(p_j)$ contains a diagonal belonging to $\mathcal{R}_j$, then it follows that all triangles on the opposite side of the gap between $\mathcal{R}_i$ and $\mathcal{R}_j$ as p_i are covered; thus the corresponding element of T_{diag_1} is set equal to zero.

(ii) Let v be any child of p_i that is occupied by another agent l with `ASSIGNED` status. Then if $\text{ad}_l(p)$ contains the diagonal $\mathcal{R}_i \cap \mathcal{R}_j$, then all triangles on the opposite side of the gap between $\mathcal{R}_i$ and $\mathcal{R}_j$ as p_i are covered; thus the corresponding element of T_{diag_1} is set equal to zero.

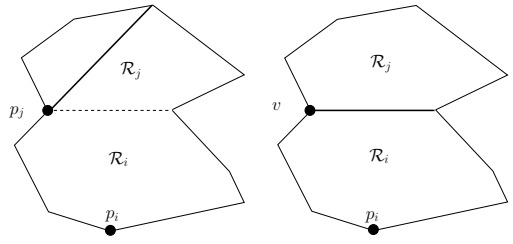


Figure 3.23: Update of T_{diag_1} in the `Depth-first Deployment` algorithm.

Remark 3.5.11 (Computation of P_i^*)**.** *The computation in steps (20) and (21) of the* `Depth-first Deployment` *described earlier are always possible. To see this, please refer to Figure 3.24. Since the number of triangles that are uncovered beyond each gap is at most one (by virtue of the fact that all elements of* T_{diag_1} *are less than or equal to one), the triangulation graph that is dual to the triangles in* $\mathcal{R}_i$ *together with the uncovered triangles beyond each gap and the triangle in* $\mathcal{R}_j$ *that shares an edge with* $\mathcal{R}_i$*, where* p_j *is the parent of the agent at* p_i*, is known completely from* p_i*. Here, the root is the triangle in* $\mathcal{R}_j$*. Also, let us assume that the locations of* $P^*(\mathcal{T}(\mathcal{R}, \mathcal{P}), r)$ *coinciding with the children of* p_i *are known together with their corresponding associated diagonals. We shall see later that this is indeed the case when all elements of* T_{diag_1} *are less than or equal to one. It is also clear that with this information a removable fork and a removable chain of the local triangulation graph is a removable fork and a removable chain, respectively, in the global triangulation graph. Thus, rules* `R1` *and* `R2` *of* `Kernel Computation Algorithm` *can be applied locally on triangulation graph. If rule* `R3` *is applied, then the associated diagonal is the diagonal that the root shares with its child. In this case, that diagonal is the edge of* $\mathcal{R}_i$ *containing* p_i*. Thus, not applying* `R3` *is consistent with the definition of* P_i^* *in step (20).*

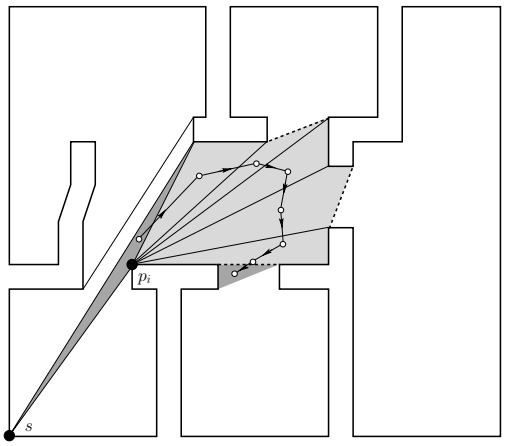


Figure 3.24: Local computation of the points P_i^* in the `Depth-first Deployment` algorithm.

We are now ready to state the main result of this section.

Theorem 3.5.12 (Convergence and Run Time Analysis)**.** *Given a simply connected polygon* Q *with* n *vertices, assume that* k *agents are initially collocated at* $s \in \text{Ve}(Q)$*. Then the following statements hold:*

(i) there exists a finite time t^*, such that for all times greater than t^* there is at least one agent on $\min\{|P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}),r)|,k\}$ points of $P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}),r)$;

(ii) if $k \geq \frac{n}{3}$, then the deployment problem is solved in finite time by the algorithm.

(iii) assuming unit speed for any agent and bounded diameter of Q, the time taken for task completion is $t^* \in O(n)$.

Proof. We first prove statement (i) along the same lines as the proof for Theorem 3.4.5 (i). We begin by showing that the first agent to acquire the `ASSIGNED` status must be located on a node belonging to $P^*(\mathcal{T}_Q(\mathcal{R},\mathcal{P}),r)$.

For the first agent that acquires the `ASSIGNED` status, steps (1)-(11) of the `Depth-first Deployment` algorithm are not executed. In step (12) it is assumed that the agent belongs to a node $p_i \in \mathcal{P}$. This is true because, throughout the execution of the algorithm, as long as the status of the agent is `UNASSIGNED`, the agent executes only the `Move-to-Parent Algorithm` motion primitive or the `Move-to-Child Algorithm` motion primitive to move towards a child in $\mathcal{P}$. Thus, steps (12) and (13) are executed using sensory information and the memory contents. Step (14) is again not executed since there are no other agents with `ASSIGNED` status. Initially the agent is located at the root with p_{last} initialized to the root location. Therefore, $p_{\text{last}} \in \mathcal{R}_i$ (initially p_i is the root), hence steps (15) and (16) are not executed. But all elements of T_{diag_1} are initialized to be equal to two and hence steps (17)-(35) are not executed. In step (36), the agent moves towards the next child that belongs to $\mathcal{P}$. The above sequence of steps is repeated till the agent reaches a node of the navigation graph having no children. In this case, $T_{\text{diag}_1} = \emptyset$. Therefore, the condition in step (17) of the algorithm are true. From Remark 3.5.11, we know that the computation in steps (18) and (19) are indeed always possible. Since the agent at p_i has no children, p does not exist in step (22) of the algorithm. If now, $p_i \in P_i^*$, then we are done. If not, then the agent moves to its parent, say p_l, having set the T_{diag_2} corresponding to the gaps $\mathcal{R}_l \cap \mathcal{R}_i$ equal to either zero or one. Before moving to the parent, the list T_{diag_1} is deleted and the list T_{diag_2} becomes the current T_{diag_1}. Thus, if an agent returns to a node from its child belonging to $\mathcal{P}$, then the values of the elements in T_{diag_1} corresponding to the relevant gaps are set to either zero or one. Now, in step (36) the agent moves to the next child in $\mathcal{P}$ such that an element of T_{diag_1} corresponding

to that child is greater than one. Therefore, after a finite number of steps again the agent is at a node of the navigation graph such that the condition in step (17) is true. Notice, that at least one element of T_{diag_1} is greater than one. Also, if the agent is located on p_i, then $\mathcal{R}_i$ contains at least one triangle as well. Thus, there must exist either a removable fork or a removable chain in the local triangulation graph described in Remark 3.5.11. Hence, after a finite time, the set P_i^* is not empty and thus the agent acquires the `ASSIGNED` status at either step (26) or (35).

Any subsequent agent that acquires the `ASSIGNED` status will acquire it either at step (26) or step (33) of the algorithm. Since the previous agents with `ASSIGNED` status belong to nodes in $P^*(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}), r)$ with the correct associated diagonals, it follows that for any other agent, the computation in steps (18) and (19) of the algorithm are correct as well. Thus, any agent with `ASSIGNED` status that is not moving must be on a node that belongs to $P^*(\mathcal{T}_Q(\mathcal{R}, \mathcal{P}), r)$.

Finally, the remainder of the proof can be completed along the lines of the proof for Theorem 3.4.5 (i).

Statement (ii) is a consequence of statement (i) and Theorem 3.5.6 (iii).

To prove statement (iii), notice that once an agent at a node p_i has visited all the gaps of $\mathcal{R}_i$ once and returned to p_i, all elements of T_{diag_1} are either equal to zero or one. It then, in the worst case, might have to visit each of its children once more before going back to the parent of p_i. The number of children can be at most twice the number of gaps of p_i. Thus, each gap of $\mathcal{R}_i$ is visited at most thrice by an agent. Since, the cardinality of $\mathcal{R}$ is at most $n-2$, the cardinality of the gaps is at most $n-3$. Thus, each agent in the worst case takes $O(n)$ time to visit all the nodes of the navigation graph. The worst case time for the entire group is obtained when all the agents move along the same path in the navigation graph and get deployed one at a time. □

3.5.4 Distributed information processing

In this section, we describe the various ingredients of the `Depth-first Deployment` algorithm described in the previous section.

Communication

The information vector $\mathcal{I}$ of each agent to be broadcast consists of the logic variable $\{\texttt{status}, \text{ad}_i(v)\}$, where as in the previous section, `status` assumes values in the sets $\{\texttt{ASSIGNED}, \texttt{UNASSIGNED}\}$ and $\text{ad}_i(v)$ are the associated diagonals of a vertex v. This is broadcast along with the UID i of the agent as described in Section 3.2.

Motion primitives

From Lemma 3.5.2 (ii), we know that adjacent nodes of the navigation graph are mutually visible. Therefore, the `Move-to-Parent Algorithm` and `Move-to-Child Algorithm` are identical to those in the previous section.

Ordering the children

The children are ordered in the same way as in the algorithm for connected deployment.

Memory management

In addition to the quantities required in the previous section, an agent stores of list T_{diag}.

The above geographic and other required information is gathered and managed by the agents via the following state transition laws and communication protocols. At this time, we make full use of the computation, communication, and sensing abilities of the agents described in Section 3.2.

(i) The memory content $\mathcal{M}$ of each agent is a $7-$tuple $(p_{\text{parent}}, p_{\text{last}}, g_1, g_2, T_{\text{diag}}, \texttt{status}, \text{ID}^*)$, where T_{diag} is a list whose elements are lists themselves. Then T_{diag_1} refers to the first list of T_{diag}. Let an agent be at a node v. If v does not belong to $\mathcal{P}$, then T_{diag_1} is an empty set. If $v \in \mathcal{P}$, say p_j, then T_{diag_1} is a list of positive integers assuming values in the set $\{0, 1, 2\}$. The size of the list is equal to the number of gaps in $\mathcal{R}_j$. The list T_{diag_2} refers to the list of integers associated with the parent of v and so on. For any agent i, at time $t = 0$, $\mathcal{M}_i(0) = \{(a_i(0)), a_i(0), a_i(0), a_i(0), (2, 2, \cdots), \texttt{UNASSIGNED}, i\}$.

During run time, with the exception of T_{diag}, the other components of $\mathcal{M}$ are updated as described in the previous section. The list T_{diag} is maintained as follows:

(ii) After an agent moves from a node p_i to a child node v via `Move-to-Child Algorithm`, T_{diag} is updated as follows: if $v \in \mathcal{P}$, say equal to p_j, then a list $(2, 2, 2, \cdots)$ is added to the beginning to T_{diag} where the length of the list is equal to the number of gaps in $\mathcal{R}_j$.

(iii) After an agent moves from a node v to the parent node p_i via `Move-to-Parent Algorithm`, T_{diag} is updated as follows: the first element of T_{diag} is deleted; other updates are done as in the `Depth-first Deployment` algorithm.

(iv) `status` and ad_i are managed according to the `Depth-first Deployment` algorithm.

The following lemma quantifies the memory and communication complexities involved in the distributed information processing scheme described above.

Lemma 3.5.13 (Memory and communication complexity)**.** *Let k be the total number of agents and the UIDs of the agents belong to $\{1, \ldots, k\}$ and let the diameter of Q be bounded. For any agent i, at any time t, the following properties hold:*

(i) The number of bits required to store $\mathcal{M}_i$ is $O(d \log k + dw)$, where d is the depth of $\mathcal{G}_{\text{nav}}(s, Q)$ and w is the maximum number of gaps in any set $\mathcal{R}_j \in \mathcal{R}$; and

(ii) the number of bits in any broadcast message is $O(\log k)$.

Proof. The memory is a 7-tuple; the objects p_{parent}, g_1, g_2, T_{diag} and ID^* are lists of length at most d and the objects p_{last} and `status` are of length one. Since the diameter of Q is bounded, the number of bits required to store the relative locations of the elements of p_{parent} with finite resolution is $O(1)$. This is also true for the elements of g_1 and g_2. The elements of T_{diag} are lists of integers themselves and the maximum number of elements in each of the lists is w. Thus the memory required to store T_{diag} is $O(dw)$. The value of an element of ID^* is bounded by the UID of the agent which is bounded by k. It, therefore, takes at most $\log k$ bits to store an element of ID^*. Statement (i) follows directly. A broadcast message consists of the UID, `status` and the associated diagonals of a given vertex. The former is bounded by k and the latter two are of constant size. Statement (ii) follows. □

3.5.5 Simulation results

In this section we present simulation results for the `Depth-first Deployment` algorithm described in Section 3.4.3. The algorithms have been implemented in `MATLAB`. The environment, Q, the root s and the navigation graph $\mathcal{G}_{\text{nav}}(s, Q)$ are as shown in Figure 3.11. Note that Q is chosen to represent a typical floor plan. Figure 3.25 shows, from left to right, the evolution of the simulation. At the end of the simulation, the agents are located at the points P^*_{con}, which in this case coincides with nodes of the navigation graph. The yellow sets denote the star-shaped sets owned by the agents that are already `ASSIGNED`. The star-shaped regions that are not owned by any of the agents are still visible by other agents.

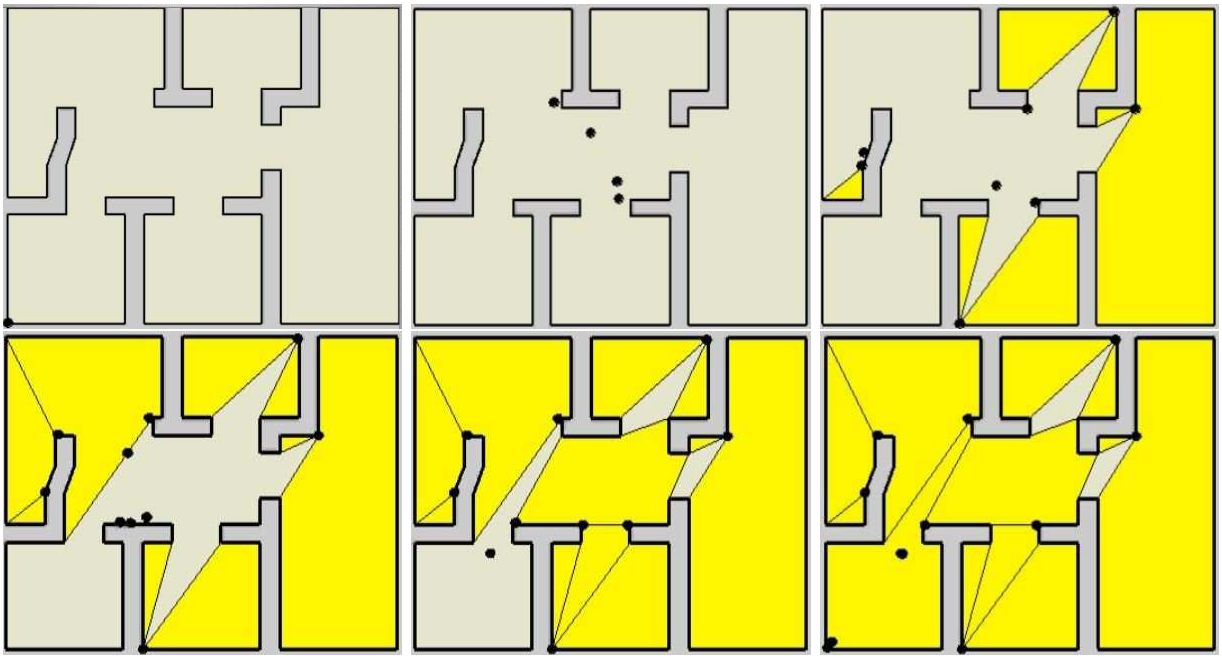


Figure 3.25: Execution of the `Depth-first Deployment` algorithm over a prototypical floor plan. As forecasted by our analysis, any point in the environment is visible to at least one agent.

3.6 Conclusions

We consider a group of robotic agents initially collocated at the same point of an unknown non-convex environment. We consider the problem of designing a distributed algorithm to deploy the agents over the environment so that all points in environment are visible to at least one agent.

Using a top-down incremental partition algorithm together with bottom-up deployment scheme, we design the `Depth-first Deployment` algorithm which solves the above problem. The number of agents required to guarantee that the task is achieved is $\lfloor n/3 \rfloor$, where n is the number of vertices of the environment. Remarkably, this number is the identical to that even if the environment were known a priori; see the famous Art Gallery Theorem [76]. A second problem that we consider is the deployment problem but under the additional constraint that the visibility graph of the final configuration of the agents is connected. Using an approach similar to that in the `Depth-first Deployment` algorithm, we propose the `Connected Depth-first Deployment` algorithm that solves the new problem. The number of agents required to guarantee that the task is achieved is $(n-1)/2$. Again, remarkably, this number differs by at most one from the number that is obtained even if the environment is known a priori; see [77].

CHAPTER 4

Multirobot rendezvous in nonconvex environments

4.1 Introduction

In this chapter we design algorithms to steer a group of robots equiipped with visibility sensors to a common location, or rendezvous, in a nonconvex environment. The rendezvous problem is a fundamental motion coordination problem for collections of robots. In its essence, it is the most basic formation control problem and can be used as a building block for more sophisticated behaviors. It is related to the classic consensus problem in distributed algorithms. This problem is, here, tackled in a fully distributed manner, i.e., our robots do not have any global knowledge about the position of all other robots, do not have any global knowledge about the environment, and do not share a common reference frame. The information that is available to an individual robot is only what is provided by a local "visibility sensor." In other words, a robot can measure the relative position of a second robot if and only if the robots are visible to each other in the nonconvex environment and lie within a given distance of each other.

The literature on multirobot systems is very extensive. Examples include the survey in [101] and the special issue [102] of the IEEE Transaction on Robotics and Automation. Our multi-robot model is inspired by the literature on networks of mobile interacting robots: an early contribution is the model proposed in [103] consisting of a group of identical "distributed anonymous mobile robots" characterized as follows. Each robot completes repeatedly a cycle of operations: it senses the relative position of all other robots, elaborates this information, and moves. In this early work, the robots share a common clock. A related model is presented in [104], where the robots evolve asynchronously, have limited visibility, and share a common reference frame. For these types of multirobot systems, the "multi-agent rendezvous" problem and the first "circumcenter algorithm" have been introduced in [81]. This circumcenter algorithm has been extended to various asyn-

chronous strategies in [104, 105], where rendezvous is referred to as the "gathering" problem. The circumcenter algorithm has been extended beyond planar problems to arbitrary dimensions in [21], where its robustness properties are also characterized. *None of these previous works considers the problem in nonconvex environments with line-of-sight visibility sensors.*

We conclude the literature review by mentioning that formation control and rendezvous problems have been widely investigated with different assumptions on the inter-agent sensing. For example, a control law for groups with *time-dependent* sensing topology is proposed in [106]; this and similar works, however, depend upon a critical assumption of connectivity of the inter-agent sensing graph. This assumption is imposed without a model for when two robots can detect and measure each other's relative position. In this chapter, we consider *position-dependent* graphs and, extending to visibility sensors a key idea in [81], we show how to constrain the robots' motion to maintain connectivity of the inter-agent sensing graph.

Next, we describe the essential details of our model. We consider a group of robotic agents moving in a nonconvex environment without holes. We assume each robot is modeled as a point mass. We assume that each robot is equipped with an *omnidirectional limited-range visibility sensor*; the nomenclature is adopted from [93, Section 11.5]. Such a sensor is a device (or combination of devices) that determines within its line of sight and its sensing range the following quantities: (i) the relative position of other robots, and (ii) the relative position of the boundary of environment. By omnidirectional we mean that the field-of-vision for the sensor is 2π radians. Examples of visibility sensors are scanning laser range finders with accurate distance measurements at high angular density,[1] time-of-flight range cameras,[2] and optical depth sensors based on structured light systems, e.g., see [107]. The range data obtained from the sensors can be processed to obtain a geometric representation of the area visible from a robot, e.g., see [108]. We do not directly address in this work issues related to feature extraction from range data. We assume that the algorithm regulating the robots' motion is memoryless, i.e., we consider static feedback laws. Given this model, the goal is to design a discrete-time algorithm which ensures that the robots converge to a common location within the environment. See Figure 4.1 for a graphical description of a simulation.

This chapter's main contribution is a novel provably correct algorithm that achieves rendezvous

[1] E.g., the Hokuyo URG-04LX, see `http://www.hokuyo-aut.jp`, and the Sick S2000, see `http://www.sick.com`.
[2] E.g., the SwissRanger SR-3000, see `http://www.swissranger.ch`.

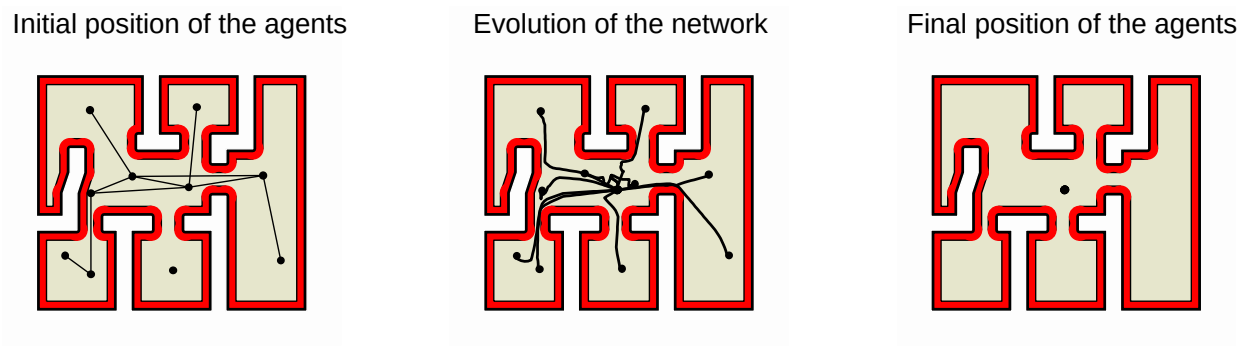


Figure 4.1: Execution of the `Perimeter Minimizing Algorithm` described in Section 4.5.1 on a group of robots distributed in a polygon, Q, shaped like a typical floor plan. The graph shown in the left-most figure is the r-range visibility graph $\mathcal{G}_{r\text{-vis},Q_\epsilon}$ (see Section 4.2).

in a nonconvex planar environment among robots with omnidirectional range-limited visibility sensors. Rendezvous is achieved among all robots if the inter-agent sensing graph is connected at the initial time or becomes connected at any time during the evolution.

The technical approach contains a number of novel contributions and proceeds as follows. First, we review a few useful geometric notions [84], such as robust visibility [109] and proximity graphs [110], and introduce various novel visibility graphs. Second, to maintain connectivity during the system evolution, we design novel constraint sets that (i) ensure that the visibility between two robots is preserved, and (ii) are upper semicontinuous or closed maps of the robots' positions. Third, based on a discussion on visibility graphs, we define a new proximity graph, called the locally-cliqueless visibility graph, which contains fewer edges than the visibility graph, and has the same connected components. This construction is useful in the connectivity maintenance problem. Fourth, we provide a careful analysis of the algorithm we propose. As novel Lyapunov function, we consider the perimeter of the relative convex hull of the robot positions. The main theorem is proved via our recent version of the LaSalle Invariance Principle for set-valued maps [21]. Fifth and final, extensive simulations validate our theoretical results and establish the convergence of our algorithm under more realistic assumptions than the ones adopted in the theoretical analysis: our algorithm performance is still adequate assuming asynchronous agent operation, noise errors in sensing and control, or finite-size disk robots.

The remainder of the chapter is organized as follows. Section 4.2 contains some useful geometric

notions. In Section 4.3 we model robots with visibility sensors and we introduce the rendezvous and connectivity maintenance problems. In Section 4.4 we introduce constraint sets and locally-cliqueless visibility graphs. In Section 4.5 we propose the `Perimeter Minimizing Algorithm` for the rendezvous problem. Numerical simulations are presented in Section 4.6. Additional analysis results and all proofs are presented in Appendices 4.8-4.10.

4.2 Geometric notions

In this section we introduce some useful geometric notions. We begin by reviewing some standard notation. Let $\mathbb{Z}_{\geq 0}$, $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_{>0}$ denote the sets of nonnegative integer, real, nonnegative real, and positive real numbers, respectively. For $p \in \mathbb{R}^2$ and $r \in \mathbb{R}_{>0}$, let $B(p, r)$ denote the *closed ball* centered at p of radius r. Given a bounded set $X \subset \mathbb{R}^2$, let $\text{co}(X)$ denote the convex hull of X, and let $\text{CC}(X)$ denote the *circumcenter* of X, i.e., the center of the smallest-radius circle enclosing X. For $p, q \in \mathbb{R}^2$, let $]p, q[= \{\lambda p + (1-\lambda)q \mid 0 < \lambda < 1\}$ and $[p, q] = \{\lambda p + (1-\lambda)q \mid 0 \leq \lambda \leq 1\}$ denote the *open* and *closed segment* with extreme points p and q, respectively. Let $|X|$ denote the cardinality of a finite set X in $\mathbb{R}^2$. Given a set of points $X \subset \mathbb{R}^2$, and another point $p \in \mathbb{R}^2$, let $\text{dist}(p, X)$ denote the Euclidean distance of p to the set X. The diameter $\text{diam}(X)$ of a compact set X is the maximum distance between any two points in X.

Now, let us turn our attention to the type of environments we are interested in. Given any compact and connected subset Q of $\mathbb{R}^2$, let ∂Q denote its boundary. A point q of ∂Q is *strictly concave* if for all $\epsilon > 0$ there exists q_1 and q_2 in $B(q, \epsilon) \cap \partial Q$ such that the open interval $]q_1, q_2[$ is outside Q. A *strict concavity* of ∂Q is either an isolated strictly concave point or a connected set of strictly concave points. According to this definition, a strict concavity is either an isolated point (e.g., points r_1 and r_2 in Figure 4.2) or an arc (e.g., arc a_1 in Figure 4.2). Also, any strictly concave point belongs to exactly one strict concavity.

Definition 4.2.1 (Allowable environments)**.** *A set $Q \subset \mathbb{R}^2$ is* allowable *if*

(i) Q is compact and simply connected;

(ii) ∂Q is continuously differentiable except on a finite number of points;

(iii) ∂Q has a finite number of strict concavities.

Recall that, roughly speaking, a set is simply connected if it is connected and it contains no hole. A particular case of the environment described above is a polygonal environment, the concavities being the reflex vertices[3] of the environment.

At almost all strictly concave points v, one can define the tangent to ∂Q. (Here, the wording "almost all" points means all except for a finite number.) At all such points v, the *internal tangent half-plane* $H_Q(v)$ is the half-plane whose boundary is tangent to ∂Q at v and whose interior does not contain any points of the concavity; see Figure 4.2.

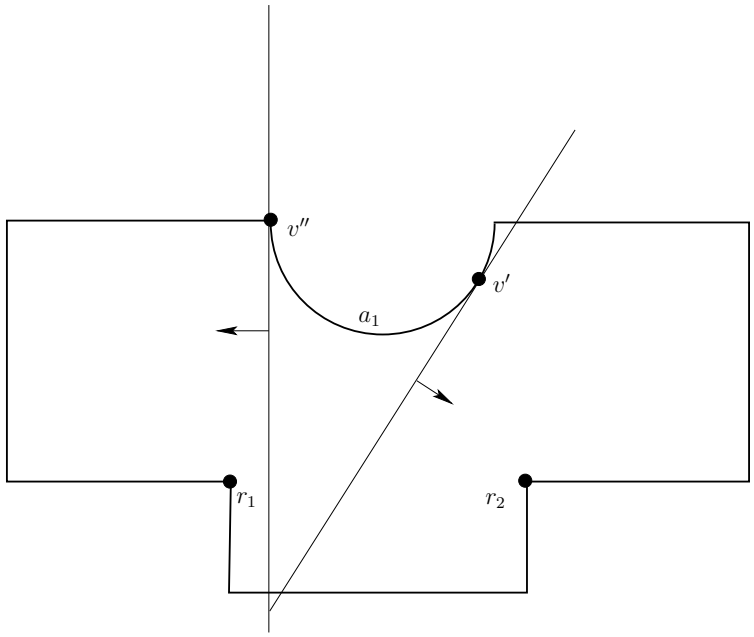


Figure 4.2: An allowable environment Q: the closed arc a_1 and the isolated points r_1, r_2 are strict concavities. v' is a point on a_1 where the slope of ∂Q is defined. $H_Q(v')$ is the half-plane with the tangent to ∂Q at v' as the boundary and the interior in the direction of the arrow. v'' is a point on a_1 where the slope of ∂Q is not well-defined. In this case, we define the tangent to be the one shown in the plot. $H_Q(v'')$ is the half-plane with the tangent to ∂Q at v'' as the boundary and the interior in the direction of the arrow.

A point $q \in Q$ is *visible* from $p \in Q$ if $[p, q] \subset Q$. The *visibility set* $\mathcal{V}(p) \subset Q$ from $p \in Q$ is the set of points in Q visible from p. This notion can be extended as follows (see [109]):

Definition 4.2.2 (Robust visibility)**.** *Take $\epsilon > 0$ and $Q \subset \mathbb{R}^2$.*

(i) The point $q \in Q$ is ϵ-robustly visible *from the point $p \in Q$ if $\cup_{q' \in [p,q]} B(q', \epsilon) \subset Q$.*

[3] A vertex of a polygon is reflex if its interior angle is strictly greater than π.

(ii) *The* ϵ-robust visibility set $\mathcal{V}(p, \epsilon) \subset Q$ *from* $p \in Q$ *is the set of points in* Q *that are* ϵ*-robustly visible from* p*.*

(iii) *The* ϵ-contraction Q_ϵ *of the set* Q *is the set* $\{p \in Q \mid ||p - q|| \geq \epsilon \text{ for all } q \in \partial Q\}$*.*

These notions are illustrated in Figure 4.3. Loosely speaking, two points p, q are mutually 0-robustly visible if and only if they are mutually visible. We present the following properties without proof in the interest of brevity.

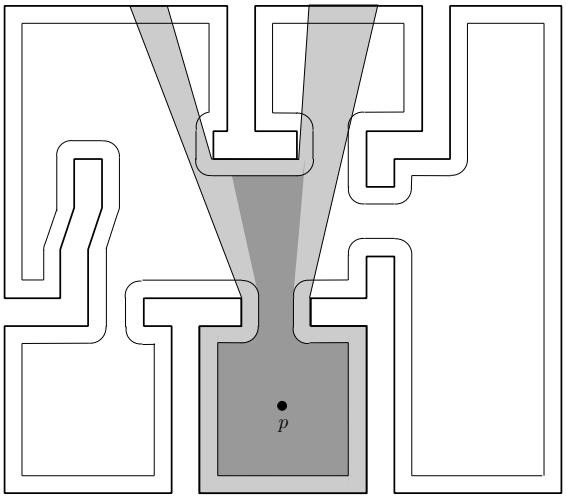


Figure 4.3: Robust visibility notions. Q is the outer polygonal environment; the ϵ-contraction Q_ϵ is the region with the curved boundary and containing the point p; the visibility set $\mathcal{V}(p)$ is the region shaded in light gray; the ϵ-robust visibility set $\mathcal{V}(p, \epsilon)$ is the region shaded in darker gray. Note that the isolated concavities of Q give rise to strictly concave arcs in Q_ϵ.

Lemma 4.2.3. *Given an allowable environment* Q *and* $\epsilon > 0$*, the following statements hold:*

(i) $q \in Q$ *is* ϵ*-robustly visible from* $p \in Q$ *if and only if* $[p, q] \subset Q_\epsilon$*;*

(ii) *if* ϵ *is sufficiently small, then* Q_ϵ *is allowable;*

(iii) *all strict concavities of* ∂Q_ϵ *have non-zero length and are continuously differentiable.*

Remarks 4.2.4. *(i) In light of Lemma 4.2.3(ii), in what follows we assume that* ϵ *is small enough for* Q_ϵ *to be connected and therefore allowable.*

(ii) Robust visibility is a useful concept in many practically meaningful ways. For example, according to this notion, points are visible only if they are at least at a distance ϵ *from the*

boundary. This is useful when an object is arbitrarily close to the boundary and is indistinguishable from the boundary itself. Additionally, the parameter ϵ might be thought of as a measure of the physical size of the robot. Thus confining the robots to the ϵ-robust visibility set guarantees free movement of the robot in the environment. Indeed, the notion of ϵ-contraction is related to the classical work on motion planning in [111], see also [93]. □

We now define some graphs which will be useful in describing the interactions between robots.

Definition 4.2.5 (Proximity graphs)**.** *A* proximity graph *is a graph whose nodes are a set of points $\mathcal{P} = \{p_1, \ldots, p_n\}$ and whose edges are a function of $\mathcal{P}$. Given $\mathcal{P} \subset Q$, $\epsilon > 0$ and $r > 0$, define:*

(i) The visibility graph $\mathcal{G}_{\text{vis},Q}$ *at $\mathcal{P}$ is the graph with node set $\mathcal{P}$ and with edge set $\mathcal{E}_{\text{vis},Q}(\mathcal{P})$ defined by: $(p_i, p_j) \in \mathcal{E}_{\text{vis},Q}(\mathcal{P})$ if and only if $[p_i, p_j] \subset Q$.*

(ii) The ϵ-robust visibility graph $\mathcal{G}_{\text{vis},Q_\epsilon}$ *is the visibility graph at $\mathcal{P}$ for Q_ϵ.*

(iii) The r-disk graph $\mathcal{G}_{r\text{-disk}}$ *at $\mathcal{P}$ is the graph with node set $\mathcal{P}$ and with edge set $\mathcal{E}_{r\text{-disk}}(\mathcal{P})$ defined by: $(p_i, p_j) \in \mathcal{E}_{r\text{-disk}}(\mathcal{P})$ if and only if $\|p_i - p_j\| \leq r$.*

(iv) The r-range visibility graph $\mathcal{G}_{r\text{-vis},\,Q}$ *at $\mathcal{P}$ is the graph with node set $\mathcal{P}$ and with edge set $\mathcal{E}_{r\text{-disk}}(\mathcal{P}) \cap \mathcal{E}_{\text{vis},Q}(\mathcal{P})$.*

(v) A Euclidean Minimum Spanning Tree $\mathcal{G}_{\text{EMST},\mathcal{G}}$ *at $\mathcal{P}$ of a proximity graph $\mathcal{G}$ is a minimum-length spanning tree of $\mathcal{G}(\mathcal{P})$ whose edge (p_i, p_j) has length $\|p_i - p_j\|$. If $\mathcal{G}(\mathcal{P})$ is not connected, then $\mathcal{G}_{\text{EMST},\mathcal{G}}(P)$ is the union of Euclidean Minimum Spanning Trees of its connected components.*

In other words, two points p, q are neighbors in the r-range visibility graph, for instance, if and only if they are mutually visible and separated by a distance less than or equal to r. Example graphs are shown in Figure 4.4. General properties of proximity graphs are defined in [21, 110]. For simplicity, when $\mathcal{G}$ is the complete graph, we denote the Euclidean Minimum Spanning Tree of $\mathcal{G}$ by $\mathcal{G}_{\text{EMST}}$.

We say that two proximity graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ *have the same connected components* if, for all sets of points $\mathcal{P}$, the graphs $\mathcal{G}_1(\mathcal{P})$ and $\mathcal{G}_2(\mathcal{P})$ have the same number of connected components consisting of the same vertices.

Definition 4.2.6 (Neighbors set)**.** *Given a set of points $\mathcal{P} = \{p_1, \ldots, p_n\}$ and a proximity graph $\mathcal{G}$, we let $\mathcal{N}_i(\mathcal{G}, \mathcal{P}) = \mathcal{N}_{i,\mathcal{G}}(\mathcal{P})$ denote the* set of neighbors including itself *of p_i. In other words, if $\{p_{i_1}, \ldots, p_{i_m}\}$ are the neighbors of p_i in $\mathcal{G}$ at $\mathcal{P}$, then $\mathcal{N}_i(\mathcal{G}, \mathcal{P}) = \mathcal{N}_{i,\mathcal{G}}(\mathcal{P}) = \{p_{i_1}, \ldots, p_{i_m}\} \cup \{p_i\}$.*

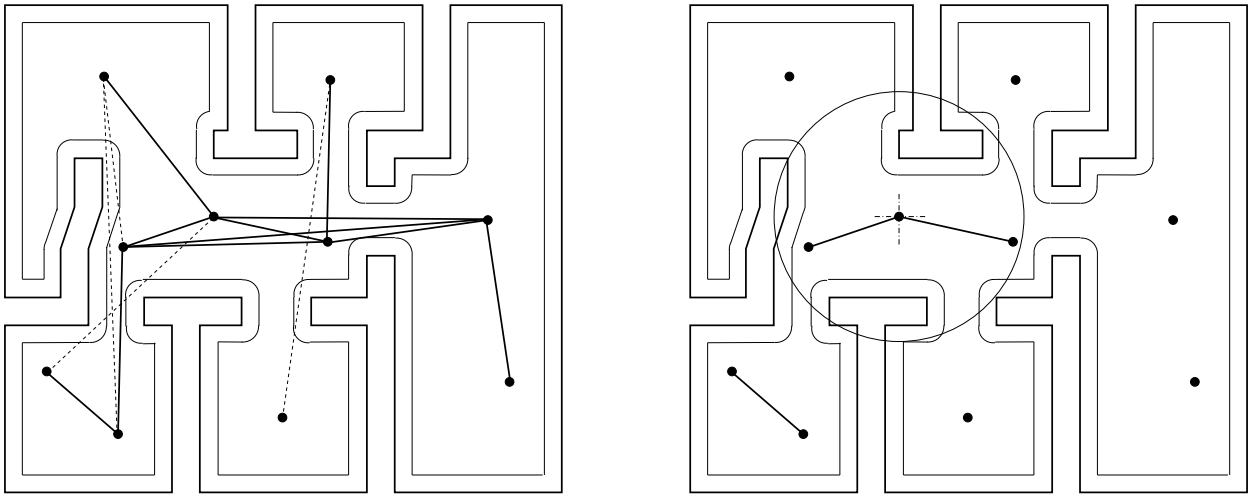


Figure 4.4: The figure on the left shows the visibility graph (whose edges are the solid lines as well as the dashed lines) and the ϵ-robust visibility graph (whose edges are the solid lines alone) of a set of points in a nonconvex polygon. The figure on the right shows the r-range ϵ-robust visibility graph. The disk in the figure shows the sensing range for one of the agents.

Definition 4.2.7 (Relative convex hull)**.** *Take an allowable environment Q.*

(i) *$X \subseteq Q$ is* relatively convex *if the shortest path inside Q connecting any two points of X is contained in X.*

(ii) *The* relative convex hull $\mathrm{rco}(X, Q)$ *of $X \subset Q$ is the smallest*[4] *relatively convex subset of Q that contains X.*

(iii) *If X is a finite set of points, then a* vertex *of* $\mathrm{rco}(X, Q)$ *is a point $p \in X$ with the property that* $\mathrm{rco}(X \setminus \{p\}, Q)$ *is a strict subset of* $\mathrm{rco}(X, Q)$*. The set of vertices of* $\mathrm{rco}(X, Q)$ *is denoted by* $\mathrm{Ve}(\mathrm{rco}(X, Q))$.

The relative convex hull of an example set of points and its vertices are shown in Figure 4.5. In what follows we will need the notion of perimeter of certain sets, and in particular, of the relative convex hull of a collection of points.

[4]That is, $\mathrm{rco}(X, Q)$ is the intersection of all relatively convex subsets of Q that contain X.

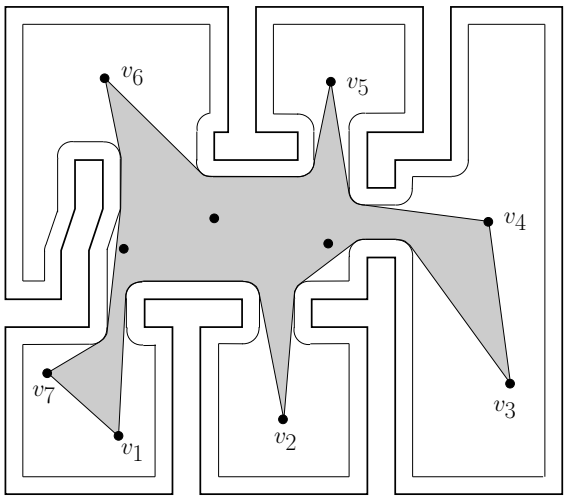


Figure 4.5: Relative convex hull $\mathrm{rco}(X, Q_\epsilon)$ of a set of points X (solid disks) inside a the ϵ-contraction of an allowable set Q. The set of vertices $\mathrm{Ve}(\mathrm{rco}(X, Q_\epsilon))$ is the set $\{v_1, \ldots, v_7\}$.

Definition 4.2.8 (Perimeter)**.** *Take an allowable environment Q and a closed subset $X \subset Q$.*

(i) If X has measurable boundary ∂X and is equal to the closure of its interior, then $\mathrm{perimeter}(X)$ *is the length of ∂X.*

(ii) If $\mathrm{rco}(X, Q)$ *is not equal to the closure of its interior, then* $\mathrm{perimeter}(\mathrm{rco}(X, Q))$ *is the length of the shortest measurable curve inside Q enclosing X.*

Remarks 4.2.9. *(i) If* $\mathrm{rco}(X, Q)$ *is equal to the closure of its interior, then its boundary is the shortest measurable curve inside Q enclosing X (i.e., the two definitions of perimeter are equivalent). On the other hand, if* $\mathrm{rco}(X, Q)$ *is a segment, then Definition 4.2.8(ii) says that the perimeter of* $\mathrm{rco}(X, Q)$ *is twice its length.*

(ii) The key property of Definition 4.2.8 is that, if X is a finite set of points in Q, then the perimeter of $\mathrm{rco}(X, Q)$ *depends continuously on the points in X.* □

%myclearpage

4.3 Synchronous robots with visibility sensors and the rendezvous and connectivity maintenance problems

In this section we model a group of n robots with visibility sensors in a given allowable environment Q. We assume that ϵ is a known positive constant sufficiently small so that Q_ϵ is allowable. For $i \in \{1, \ldots, n\}$, we model the ith robot as a point $p_i \in Q$ and we refer to Section 4.6 for an extension to a disk model. We make the following modeling assumptions:

Synchronized controlled motion model: Robot i moves at time $t \in \mathbb{Z}_{\geq 0}$ for a unit period of time, according to the discrete-time control system

$$p_i[t+1] = p_i[t] + u_i[t]. \tag{4.1}$$

We assume that there is a maximum step size $s_{\max} > 0$ for any robot, that is, $\|u_i\| \leq s_{\max}$. Note that the n identical robots are *synchronized* in the sense that the calculation of $u[t]$ in equation (4.1) takes place at the same times t for all robots.

Sensing model: Robot i senses (i) the presence and the position of any other robot that is visible and within distance r from p_i, and (ii) the subset of ∂Q that is visible and within distance $(r+\epsilon)$ from p_i. This in turn implies that the robot can sense the subset of ∂Q_ϵ that is visible and within distance r from p_i. It is convenient to define the *sensing region from position* p_i to be $\mathcal{S}(p_i) = \mathcal{V}(p_i, \epsilon) \cap B(p_i, r)$. The range r is the same for all robots.

Note that, by definition, two robots with visibility sensors detect each other's presence and relative position if and only if they are neighbors in the robust visibility graph $\mathcal{G}_{\text{vis},Q_\epsilon}$.

Remark 4.3.1 (No common reference frame)**.** *The model presented above assumes the ability of robots to sense absolute positions of other robots; this assumption is only made to keep the presentation as simple as possible. In this and subsequent remarks, we treat the more realistic setting in which the n robots have n distinct reference frames $\Sigma_1, \ldots, \Sigma_n$. We let Σ_0 denote a fixed reference frame. Notation-wise, a point q, a vector w, and a set of points S expressed with respect to frame Σ_i are denoted by q^i, w^i and S^i, respectively. For example, this means that Q^i is the environment Q as expressed in frame Σ_i. We assume that the origin of Σ_i is p_i and that*

the orientation of Σ_i with respect to Σ_0 is $R_i^0 \in \mathrm{SO}(2)$. Therefore, changes of reference frames are described by the equations: $q^0 = R_i^0 q^i + p_i^0$, $w^0 = R_i^0 w^i$, and $S^0 = R_i^0 S^i + p_i^0$. If we let $\mathcal{V}_{Q^j}(p_i^j, \epsilon)$ denote the visibility set expressed in Σ_j, for $j \in \{0, 1, \ldots, n\}$, then one can define

$$\mathcal{S}(p_i^j, Q^j) = \mathcal{V}_{Q^j}(p_i^j, \epsilon) \cap B(p_i^j, r),$$

and verify that $\mathcal{S}(p_i^0, Q^0) = R_i^0 \mathcal{S}(p_i^i, Q^i) + p_i^0$. Note that $p_i^i = 0$.

Finally, we can describe our motion and sensing model under the no common reference frame assumption. Robot i moves according to

$$p_i^0[t+1] = p_i^0[t] + R_i^0[t] u_i[t], \tag{4.2}$$

and it senses the robot positions p_j^i and the subset of $(\partial Q)^i$ that are within the sensing region $\mathcal{S}(p_i^i, Q^i)$. □

We now state the two control design problems addressed in this chapter for groups of robots with visibility sensors.

Problem 4.3.2 (Rendezvous)**.** *The* rendezvous problem *is to steer each agent to a common location inside the environment Q_ϵ. This objective is to be achieved (1) with the limited information flow described in the model above, and (2) under the reasonable assumption that the initial position of the robots $\mathcal{P}[0] = \{p_1[0], \ldots, p_n[0]\}$ gives rise to a connected robust visibility graph $\mathcal{G}_{\mathrm{vis},Q_\epsilon}$ at $\mathcal{P}[0]$.*□

As one might imagine, the approach to solving the rendezvous problem involves two main ideas: first, the underlying proximity graph should not lose connectivity during the evolution of the group; second, while preserving the connectivity of the graph, the agents must move closer to each other. This discussion motivates a second complementary objective.

Problem 4.3.3 (Connectivity maintenance)**.** *The* connectivity maintenance problem *is to design (state dependent) control constraints sets with the following property: if each agent's control takes values in the control constraint set, then the agents move in such a way that the number of connected components of $\mathcal{G}_{\mathrm{vis},Q_\epsilon}$ (evaluated at the agents' states) does not increase with time.* □

4.4 The connectivity maintenance problem

In this section, we maintain the connectivity of the group of agents with visibility sensors by designing control constraint sets that guarantee that every edge of $\mathcal{G}_{r\text{-vis},\, Q_\epsilon}$ (i.e., every pair of mutually range-limited visible robots) is preserved. We have three objectives in doing so. First, the sets need to depend continuously on the position of the robots. Second, the sets need to be computed in a distributed way based only on the available sensory information. Third, the control constraint sets should be as "large" as possible so as to minimally constrain the motion of the robots. Because it appears difficult to formalize the notion of "largest continuous constraint set that can be computed in a distributed fashion," we instead propose a geometric strategy to compute appropriate constraint sets and we show in the next section that our proposed geometric strategy is sufficiently efficient for the rendezvous problem.

4.4.1 Preserving mutual visibility: The `Constraint Set Generator Algorithm`

Consider a pair of robots in an environment Q that are ϵ-robustly visible to each other and separated by a distance not larger than r. To preserve this range-limited mutual visibility property, we restrict the motion of the robots to an appropriate subset of the environment. This idea is inspired by [81] and we begin by stating the result therein. Let the sensing region of robot i located at p_i be $\mathcal{S}(p_i) = B(p_i, r)$, for some $r > 0$. If at any time instant t, $\|p_i[t] - p_j[t]\| \leq r$ then to ensure that at the next time instant $t+1$, $\|p_i[t+1] - p_j[t+1]\| \leq r$, it suffices to impose the following constraints on the motion of robots i and j:

$$p_i[t+1],\ p_j[t+1] \in B\Big(\frac{p_i[t] + p_j[t]}{2}, \frac{r}{2}\Big),$$

or, equivalently,

$$u_i[t],\ u_j[t] \in B\Big(\frac{p_j[t] - p_i[t]}{2}, \frac{r}{2}\Big).$$

In summary, $B(\frac{p_j - p_i}{2}, \frac{r}{2})$ is the control constraint set for robot i and j. This constraint is illustrated in Figure 4.6 (left).

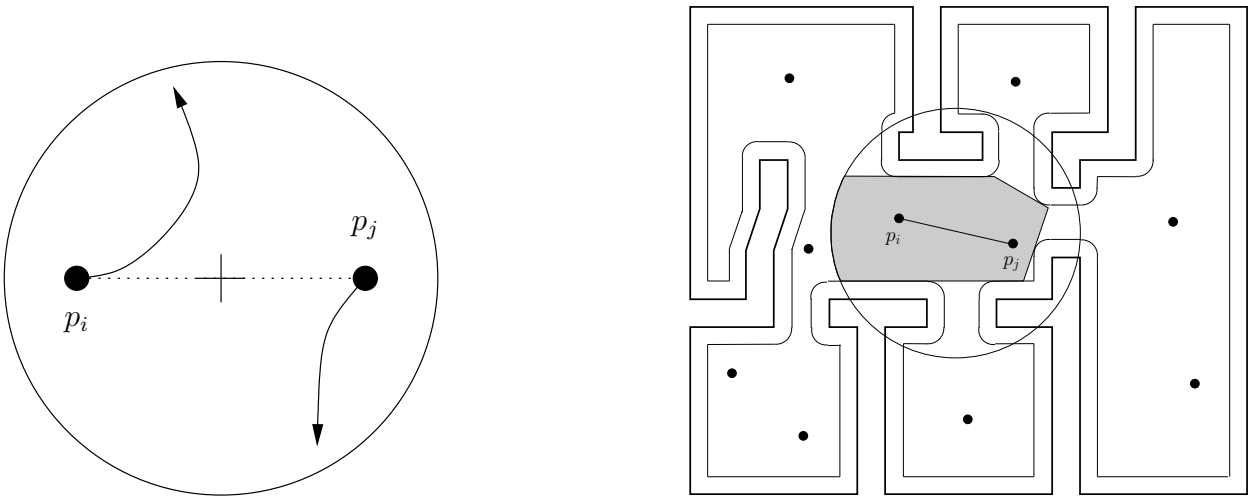


Figure 4.6: In the figure on the left, starting from p_i and p_j, the robots are restricted to move inside the disk centered at $\frac{p_i+p_j}{2}$ with radius $\frac{r}{2}$. In the figure on the right, the robots are constrained to move inside the shaded region which is a convex subset of Q_ϵ intersected with the disk centered at $\frac{p_i+p_j}{2}$ with radius $\frac{r}{2}$.

Let us now consider the case when a robot i is located at p_i in a nonconvex environment Q with sensing region $\mathcal{S}(p_i) = \mathcal{V}(p_i, \epsilon) \cap B(p_i, r)$. If at any time instant t, we have that $\|p_i[t] - p_j[t]\| \leq r$ and $[p_i[t], p_j[t]] \in Q_\epsilon$, then to ensure that $\|p_i[t+1] - p_j[t+1]\| \leq r$ and $[p_i[t+1], p_j[t+1]] \in Q_\epsilon$, it suffices to require that:

$$p_i[t+1],\ p_j[t+1] \in \mathcal{C},$$

where $\mathcal{C}$ is any convex subset of $Q_\epsilon \cap B\big(\frac{p_i[t]+p_j[t]}{2}, \frac{r}{2}\big)$; see Figure 4.6 (right). Equivalently,

$$u_i[t] \in \mathcal{C} - p_i[t],\ u_j[t] \in \mathcal{C} - p_j[t],$$

where $\mathcal{C} - p_i[t]$ and $\mathcal{C} - p_i[t]$ are the sets $\{p - p_i[t] \mid p \in \mathcal{C}\}$ and $\{p - p_j[t] \mid p \in \mathcal{C}\}$, respectively. Note that both robots i and j must independently compute the same set $\mathcal{C}$. Given the positions p_i, p_j in an environment Q, Table 4.1 describes the `Constraint Set Generator Algorithm`, a geometric strategy for each robot to compute a constraint set $\mathcal{C} = \mathcal{C}_Q(p_i, p_j)$ that changes continuously with p_i and p_j. Figure 4.7 illustrates a step-by-step execution of the algorithm.

In step `3:` of the algorithm, note that there can be multiple distinct points belonging to distinct concavities satisfying the required property. In that case, v can be chosen to be any one of them.

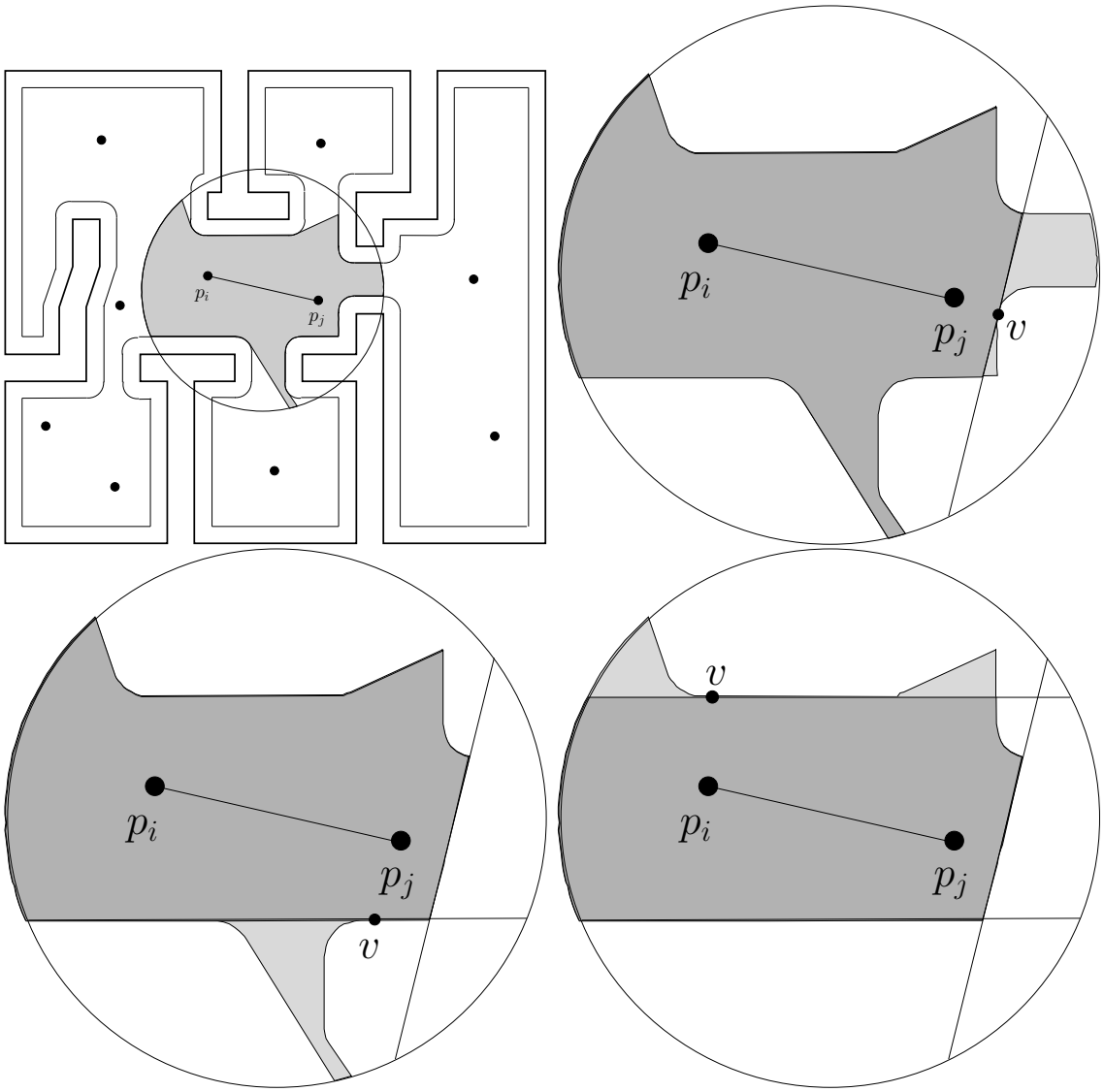


Figure 4.7: From left to right and top to bottom, a sample incomplete run of the `Constraint Set Generator Algorithm` (cf. Table 4.1). The top left figure shows $\mathcal{C}_{\text{temp}} := \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. In all the other figures, the lightly and darkly shaded regions together represent $\mathcal{C}_{\text{temp}}$. The darkly shaded region represents $\mathcal{C}_{\text{temp}} \cap H_Q(v)$, where v is as described in step `3:`. The final outcome of the algorithm, $\mathcal{C}_Q(p_i, p_j)$, is shown in Figure 4.6 (right).

Table 4.1: `Constraint Set Generator Algorithm`

Goal:	Generate convex sets to act as constraints to preserve mutual visibility
Given:	$(p_i, p_j) \in Q_\epsilon^2$ such that $[p_i, p_j] \subseteq Q_\epsilon$ and $p_j \in B(p_i, r)$

Robot $i \in \{1, \ldots, n\}$ executes the following computations:

1: $\mathcal{C}_{\text{temp}} := \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$
2: **while** $\partial\mathcal{C}_{\text{temp}}$ contains a concavity **do**
3: $\quad v :=$ a strictly concave point of $\partial\mathcal{C}_{\text{temp}}$ closest to the segment $[p_i, p_j]$
4: $\quad \mathcal{C}_{\text{temp}} := \mathcal{C}_{\text{temp}} \cap H_{Q_\epsilon}(v)$
5: **end while**
6: **return:** $\mathcal{C}_Q(p_i, p_j) := \mathcal{C}_{\text{temp}}$

The following lemma justifies this observation.

Lemma 4.4.1. *Throughout the execution of the* `Constraint Set Generator Algorithm` *in Table 4.1, let v_1, v_2 be two strictly concave points on $\partial\mathcal{C}_{\text{temp}}$ that are closest to $[p_i, p_j]$. Then $v_1 \in \mathcal{C}_{\text{temp}} \cap H_{Q_\epsilon}(v_2)$ and vice versa.*

Next, we characterize the main properties of the `Constraint Set Generator Algorithm` and the corresponding convex sets. Notice that the constraint set is defined at any point in the following set:

$$J = \left\{(p_i, p_j) \in Q_\epsilon^2 \mid [p_i, p_j] \in Q_\epsilon,\ \|p_i - p_j\| \leq r\right\}.$$

Proposition 4.4.2. (Properties of the `Constraint Set Generator Algorithm`) *Given an allowable environment Q with κ strict concavities, $\epsilon > 0$ and $(p_i, p_j) \in J$, the following statements hold:*

(i) The `Constraint Set Generator Algorithm` *terminates in at most κ steps;*

(ii) $\mathcal{C}_Q(p_i, p_j)$ is nonempty, compact and convex;

(iii) $\mathcal{C}_Q(p_i, p_j) = \mathcal{C}_Q(p_j, p_i)$; and

(iv) The set-valued map $\mathcal{C}_Q$ is closed[5] *at every point of J.*

[5]Let Ω map points in X to all possible subsets of Y. Then the set-valued map, Ω, is open at a point $x \in X$ if for any sequence $\{x_k\}$ in X, $x_k \to x$ and $y \in \Omega(x)$ implies the existence of a number m and a sequence $\{y_k\}$ in Y such that $y_k \in \Omega(x_k)$ for $k \geq m$ and $y_k \to y$. The map Ω is closed at a point $x \in X$ if for any sequence $\{x_k\}$ in X, $x_k \to x$, $y_k \to y$ and $y_k \in \Omega(x_k)$ imply that $y \in \Omega(x)$. Ω is continuous at any point $x \in X$ if it is both open and closed at x

Remark 4.4.3 (No common reference frame: continued)**.** *Consider a group of robots with visibility sensors and no common reference frame. With the notation and assumptions described in Remark 4.3.1, on can verify that the constraint sets transform under changes of coordinate frames according to:*

$$\mathcal{C}_{Q^0}(p_i^0, p_j^0) = R_i^0 \mathcal{C}_{Q^i}(p_i^i, p_j^i) + p_i^0. \tag{4.3}$$

We omit the proof in the interest of brevity. □

For each pair of mutually visible robots, the execution of the `Constraint Set Generator Algorithm` outputs a control constraint set such that, if the robots' motions are constrained to it, then the robots remain mutually visible. Clearly, given a connected graph at time t, if every robot remains connected with all its neighbors at time $t+1$ (i.e., each pair of mutually visible robots remain mutually visible), then the connectivity of the graph is preserved. This can be accomplished as follows. For robot $i \in \{1, \ldots, n\}$ at $p_i \in Q_\epsilon$, define the control constraint set

$$C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{r\text{-vis, }Q_\epsilon}}) = \bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_{r\text{-vis, }Q_\epsilon}}} \mathcal{C}_Q(p_i, p_j). \tag{4.4}$$

Now, if $u_i \in C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{r\text{-vis, }Q_\epsilon}}) - p_i$, for all $i \in \{1, \ldots, n\}$, then all neighboring relationships in $\mathcal{G}_{r\text{-vis, }Q_\epsilon}$ are preserved at the next time instant. Using inputs that satisfy these constraints, the number of edges in $\mathcal{G}_{r\text{-vis, }Q_\epsilon}$ is guaranteed to be nondecreasing.

4.4.2 The locally-cliqueless visibility graph

In this section, we propose the construction of constraint sets that are, in general, larger than $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{r\text{-vis, }Q_\epsilon}})$. To do this, we define the notion of *locally-cliqueless graph.* The locally-cliqueless graph of a proximity graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$, and therefore has generally fewer edges, but it has the same number of connected component as $\mathcal{G}$. This fundamental property will be very useful in the design of less conservative constraint sets.

Before proceeding with the definition of locally-cliqueless graph, let us recall that (i) a *clique* of a graph is a complete subgraph of it, and (ii) a *maximal clique of an edge* is a clique of the graph that contains the edge and is not a strict subgraph of any other clique of the graph that also

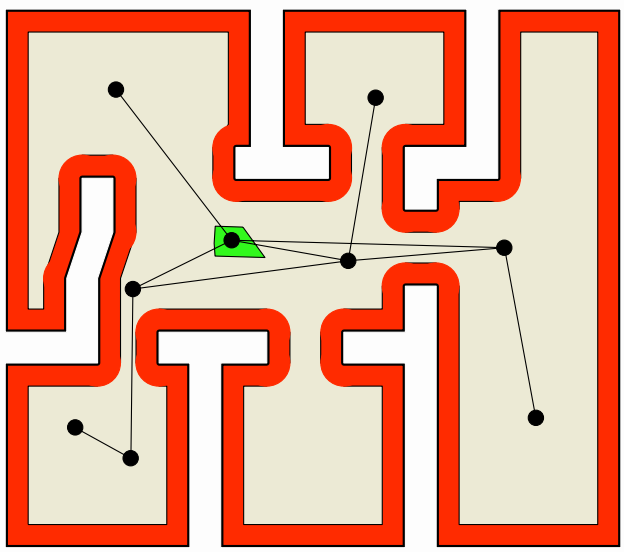


Figure 4.8: The green convex set in the center represents $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{r\text{-vis},\mathrm{Q}_\epsilon}})$. The black disks represent the position of the robots. The straight line segments between pairs of robots represent edges of $\mathcal{G}_{r\text{-vis},\mathrm{Q}_\epsilon}$. Here, p_i is the black disk contained in the constraint set.

contains the edge.

Definition 4.4.4. (Locally-cliqueless graph of a proximity graph) *Given a point set $\mathcal{P}$ and an allowable environment Q, the* locally-cliqueless graph $\mathcal{G}_{\mathrm{lc},\mathcal{G}}$ *at $\mathcal{P}$ of a proximity graph $\mathcal{G}$ is the proximity graph with node set $\mathcal{P}$ and with edge set $\mathcal{E}_{\mathrm{lc},\mathcal{G}}(\mathcal{P})$ defined by: $(p_i, p_j) \in \mathcal{E}_{\mathcal{G}_{\mathrm{lc},\mathcal{G}}}(\mathcal{P})$ if and only if $(p_i, p_j) \in \mathcal{E}_{\mathcal{G}}(\mathcal{P})$ and (p_i, p_j) belongs to a set $\mathcal{E}_{\mathcal{G}_{\mathrm{EMST}}}(\mathcal{P}')$ for any maximal clique $\mathcal{P}'$ of the edge (p_i, p_j) in $\mathcal{G}$.*

In combinatorial optimization, it is a well-known that finding the maximal clique of a graph is an NP complete problem. However, efficient polynomial time heuristics are detailed in [112].

For simplicity, we will refer to the locally-cliqueless graph of the proximity graphs $\mathcal{G}_{\mathrm{vis},Q}$, $\mathcal{G}_{\mathrm{vis},Q_\epsilon}$ or $\mathcal{G}_{\mathrm{r\text{-}vis},Q_\epsilon}$ as *locally-cliqueless visibility graphs.* Figure 4.9 shows an example of a locally-cliqueless visibility graph.

Theorem 4.4.5. (Properties of a locally-cliqueless graph of a proximity graph) *Let $\mathcal{G}$ be a proximity graph. Then, the following statements hold:*

(i) $\mathcal{G}_{\mathrm{EMST},\mathcal{G}} \subseteq \mathcal{G}_{\mathrm{lc},\mathcal{G}} \subseteq \mathcal{G}$;

(ii) $\mathcal{G}_{\mathrm{lc},\mathcal{G}}$ *and* $\mathcal{G}$ *have the same connected components.*

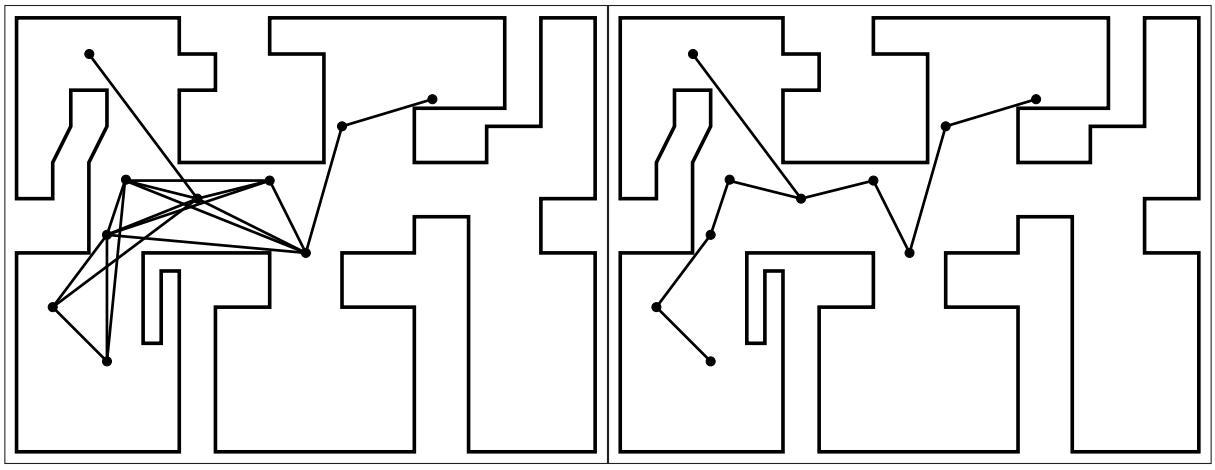


Figure 4.9: Visibility graph (left) and locally-cliqueless visibility graph (right).

In general, the inclusions in Theorem 4.4.5(i) are strict. Figure 4.10 shows an example where $\mathcal{G}_{\text{EMST},\mathcal{G}_{\text{vis},Q}} \subsetneq \mathcal{G}_{\text{lc},\mathcal{G}_{\text{vis},Q}} \subsetneq \mathcal{G}_{\text{vis},Q}$.

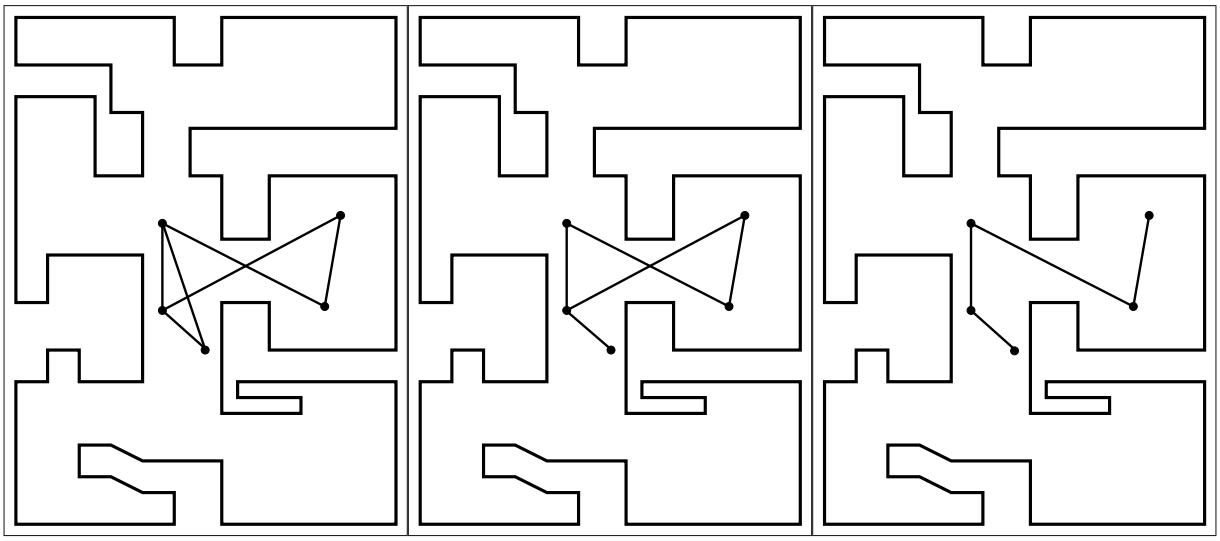


Figure 4.10: From left to right, visibility graph, locally-cliqueless graph and Euclidean Minimum Spanning Tree of the visibility graph.

The next result follows directly from Theorem 4.4.5.

Corollary 4.4.6. (Properties of locally-cliqueless visibility graphs) *Let Q and Q_ϵ, $\epsilon > 0$, be allowable environments. Let $\mathcal{G}$ be either one of the graphs $\mathcal{G}_{\text{vis},Q}$, $\mathcal{G}_{\text{vis},Q_\epsilon}$ or $\mathcal{G}_{\text{r-vis},Q_\epsilon}$. Then, the following statements hold:*

(i) $\mathcal{G}_{\text{EMST},\mathcal{G}} \subseteq \mathcal{G}_{\text{lc},\mathcal{G}} \subseteq \mathcal{G}$;

(ii) $\mathcal{G}_{\mathrm{lc},\mathcal{G}}$ and $\mathcal{G}$ have the same connected components.

Let us now proceed to define new constraint sets that are in general larger than the ones defined in (4.4). For simplicity, let $\mathcal{G} = \mathcal{G}_{r\text{-vis},Q_\epsilon}$, and consider its locally-cliqueless graph $\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}$. For robot $i \in \{1,\dots,n\}$ at position p_i, define the constraint set

$$C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}}) = \bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}}} \mathcal{C}_Q(p_i,p_j). \tag{4.5}$$

Since $\mathcal{G}_{\mathrm{lc},\mathcal{G}}$ is a subgraph of $\mathcal{G}$ (cf. Corollary 4.4.6(i)), we have $\mathcal{N}_{i,\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}} \subseteq \mathcal{N}_{i,\mathcal{G}} = \mathcal{N}_{i,\mathcal{G}_{r\text{-vis},\,Q_\epsilon}}$, and therefore

$$C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{r\text{-vis},\,Q_\epsilon}}) \subseteq C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}}).$$

In general, since $\mathcal{G}_{\mathrm{lc},\mathcal{G}}$ is a strict subgraph of $\mathcal{G}$, the set $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}})$ is strictly larger that $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{r\text{-vis},\,Q_\epsilon}})$.

Now, if $u_i \in C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}}) - p_i$ for all $i \in \{1,\dots,n\}$, then all neighboring relationships in the graph $\mathcal{G}_{\mathrm{lc},\,\mathcal{G}}$ are preserved at the next time instant. As a consequence, it follows from Corollary 4.4.6(ii) that the connected components of $\mathcal{G}_{r\text{-vis},\,Q_\epsilon}$ are also preserved at the next time instant. Thus, we have found constraint sets (4.5) for the input that are larger than the constraint sets (4.4), and are yet sufficient to preserve the connectivity of the overall group.

Remark 4.4.7. ***(Distributed computation of locally-cliqueless visibility graphs)*** *According to the model specified in Section 4.3, each robot can detect all other robots in its sensing region $\mathcal{S}(p_i) = \mathcal{V}(p_i,\epsilon) \cap B(p_i,r)$, i.e., its neighbors in the graph $\mathcal{G}_{r\text{-vis},\,Q_\epsilon}$. Given the construction of the constraint sets in this section, it is important to guarantee that the set of neighbors of robot i in the locally-cliqueless graph $\mathcal{G}_{\mathrm{lc},\mathcal{G}}$ can be computed locally by robot i. From the definition of the locally-cliqueless graph, this is indeed possible if a robot i can detect whether another robot j in its sensing region $\mathcal{S}(p_i)$ belongs to a clique of the graph $\mathcal{G}_{r\text{-vis},\,Q_\epsilon}$. This is equivalent to being able to check if two robots $p_k, p_l \in \mathcal{S}(p_i)$ satisfy the condition that $p_k \in \mathcal{S}(p_l)$ and vice versa. Note that $p_k \in \mathcal{S}(p_l)$ is equivalent to $\|p_k - p_l\| \leq r$ and $[p_k,p_l] \subseteq Q_\epsilon$. Given that $p_k - p_l = (p_k - p_i) - (p_l - p_i)$, the vector $p_k - p_l$ (and hence $\|p_k - p_l\|$) can be computed based on local sensing alone. Now, checking*

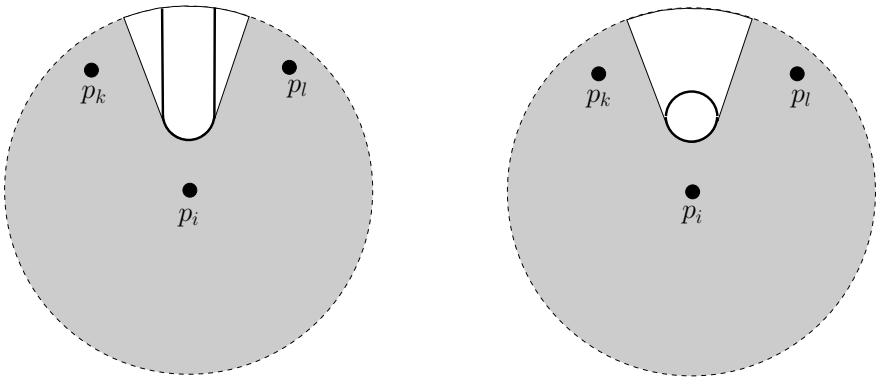


Figure 4.11: The dashed circle is centered at p_i and is of radius r. The thick curves represent the boundary of Q_ϵ; the one on the left represents the outer boundary whereas the one on the right represents a hole in the environment.

if $[p_k, p_l] \subseteq Q_\epsilon$ is possible only if Q_ϵ does not contain any hole; see Figure 4.11. In such a case, it suffices to check if the entire line segment $[p_k, p_l]$ is visible from p_i or not.

Along these same lines it is possible to state that the locally-cliqueless visibility graph can be computed also under the "no common reference frame" model described in Remarks 4.3.1 and 4.4.3.

□

4.5 The rendezvous problem: Algorithm design and analysis results

In this section, we solve the rendezvous problem through a novel `Perimeter Minimizing Algorithm`. The algorithm is inspired by the one introduced in [81] but is unique in many different ways. The rendezvous algorithm uses different graphs to maintain connectivity and to move closer to other robots. Instead of moving towards the circumcenter of the neighboring robots, the robots move towards the center of a suitably defined motion constraint set.

The section is organized as follows. We present the algorithm in Subsection 4.5.1 followed by its main convergence properties in Subsection 4.5.2.

4.5.1 The `Perimeter Minimizing Algorithm`

We begin with an informal description of the `Perimeter Minimizing Algorithm` over graphs $\mathcal{G}_{\text{sens}}$ and $\mathcal{G}_{\text{constr}}$. The sensing graph $\mathcal{G}_{\text{sens}}$ is $\mathcal{G}_{r\text{-vis},Q_\epsilon}$ while the constraint graph $\mathcal{G}_{\text{constr}}$ is either $\mathcal{G}_{\text{sens}}$ or $\mathcal{G}_{\text{lc},\mathcal{G}_{\text{sens}}}$:

> Every robot i performs the following tasks: (i) it acquires the positions of other robots that are its neighbors according to $\mathcal{G}_{\text{sens}}$; (ii) it computes a point that is "closer" to the robots it senses, and (iii) it moves toward this point while maintaining connectivity with its neighbors according to $\mathcal{G}_{\text{constr}}$.

The algorithm is formally described in Table 4.2; Figure 4.1 in the Introduction illustrates an example execution.

Table 4.2: `Perimeter Minimizing Algorithm`

Assumes: (i) $s_{\max} > 0$ is the maximal step size
(ii) Q, Q_ϵ are allowable
(iii) $\mathcal{G}_{\text{sens}}$ is $\mathcal{G}_{r\text{-vis},\,Q_\epsilon}$; $\mathcal{G}_{\text{constr}}$ is either $\mathcal{G}_{\text{sens}}$ or $\mathcal{G}_{\text{lc},\mathcal{G}_{\text{sens}}}$

Each robot $i \in \{1, \ldots, n\}$ executes the following steps at each time instant:

1: acquire $\{p_{i_1}, \ldots, p_{i_m}\}$:= positions of robots within p_i sensing region
2: compute $\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}$ and $\mathcal{N}_{i,\mathcal{G}_{\text{constr}}}$
3: compute $X_i := C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}}) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}, \mathcal{V}(p_i, \epsilon))$
4: compute $p_i^* := \text{CC}(X_i)$
5: **return:** $u_i := \dfrac{\min(s_{\max}, \|p_i^* - p_i\|)}{\|p_i^* - p_i\|}(p_i^* - p_i)$

Remarks 4.5.1. *(i) According to the algorithm proposed in [81] the robots move towards the circumcenter of their neighbors position. In the* `Perimeter Minimizing Algorithm` *the robots move towards the circumcenter of their constraint set.*

(ii) We prove later in Lemma 4.9.2 in Appendix 4.9 that X_i is convex. Therefore $\text{CC}(X_i) \in X_i$, and hence $p_i^* \in X_i$. Also, $p_i \in X_i$. Therefore, $u_i \in X_i - p_i \subseteq C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}}) - p_i$ and, in turn, p_i at the next time instant belongs to $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}})$. From our discussion in Section 4.4, this implies that the graph $\mathcal{G}_{\text{constr}}$ remains connected (or, more generally, that the number of

connected components of $\mathcal{G}_{\text{constr}}$ does not decrease). Therefore, by Corollary 4.4.6, the number of connected components of $\mathcal{G}_{\text{sens}}$ also does not decrease.

(iii) If the initial positions of the robots are in Q_ϵ, then the robots will remain forever in Q_ϵ because $p_i^* \in X_i \subseteq Q_\epsilon$.

(iv) All information required to execute the steps in the algorithm is available to a robot through the sensing model described in Section 4.3. The constraint on the input size, $\|u_i\| \leq s_{\max}$, is enforced in step `5:`. □

Finally, we conclude this section by completing our treatment of robots without a common reference frame.

Remark 4.5.2 (No common reference frame: continued)**.** *Consider a group of robots with visibility sensors and no common reference frame as discussed in Remarks 4.3.1 and 4.4.3. Because the relative convex hull and the circumcenter of a set transform under changes of coordinate frames in the same way as the constraint set does in equation* (4.3)*, one can verify that*

$$u_i(p_1^0, \ldots, p_n^0) = R_i^0 u_i(p_1^i, \ldots, p_n^i),$$

where $u_i(p_1^0, \ldots, p_n^0)$ is computed with environment Q^0 and $u_i(p_1^i, \ldots, p_n^i)$ is computed with environment Q^i. This equality implies that the robot motion with control $u_i(p_1^0, \ldots, p_n^0)$ in equation (4.1) *is identical to the robot motion with control $u_i(p_1^i, \ldots, p_n^i)$ in equation* (4.2)*.* □

4.5.2 Main convergence result

To state the main results on the correctness of the `Perimeter Minimizing Algorithm`, we require some preliminary notation. First, note that given the positions of the robots $\{p_1, \ldots, p_n\}$ at any time instant t, the algorithm computes the positions at time instant $t+1$. We can therefore think of the `Perimeter Minimizing Algorithm` as the map $T_{\mathcal{G}_{\text{sens}}, \mathcal{G}_{\text{constr}}} : Q_\epsilon^n \to Q_\epsilon^n$. Second, in what follows we will work with tuple of points $P = (p_1, \ldots, p_n) \in Q^n$. We let $\mathcal{G}(P)$ denote the proximity graph $\mathcal{G}(\mathcal{P})$ and $\text{rco}(P, Q)$ denote the relative convex hull of the set $\mathcal{P}$ inside Q, where $\mathcal{P}$ is the point set given by $\{p_i \mid i \in \{1, \ldots, n\}\}$. Third, we introduce a Lyapunov function that encodes the

rendezvous objective. Given an allowable environment Q, we recall the notions of relative convex hull and of perimeter from Section 4.2, and we define $V_{\text{perim},Q} : Q^n \to \mathbb{R}_{\geq 0}$ by

$$V_{\text{perim},Q}(P) = \text{perimeter}(\text{rco}(P, Q)).$$

Lemma 4.5.3 (Properties of Lyapunov function)**.** *The function $V_{\text{perim},Q}$ has the following properties:*

(i) $V_{\text{perim},Q}$ is continuous and invariant under permutations of its arguments;

(ii) $V_{\text{perim},Q}(P) = 0$ for $P = (p_1, \ldots, p_n)$ if and only if $p_i = p_j$ for all $i, j \in \{1, \ldots, n\}$.

The technical result on continuity plays an important role in the convergence analysis of the algorithm. Fact (ii) implies that achieving the rendezvous objective is equivalent to making $V_{\text{perim},Q_\epsilon}$ equal to zero. We are now ready to state the main result of this chapter.

Theorem 4.5.4. (Rendezvous is achieved via the `Perimeter Minimizing Algorithm`) *Let Q and Q_ϵ be allowable environments. Let $p_1, \ldots, p_n$ be a group of robots with visibility sensors in Q_ϵ. Any trajectory $\{P[t]\}_{t \in \mathbb{Z}_{\geq 0}} \subset Q_\epsilon$ generated by $P[t+1] = T_{\mathcal{G}_{\text{sens}}, \mathcal{G}_{\text{constr}}}(P[t])$ has the following properties:*

(i) if the locations of two robots belong to the same connected component of $\mathcal{G}_{\text{sens}}$ at $P[t_0]$ for some t_0, then they remain in the same connected component of $\mathcal{G}_{\text{sens}}$ at $P[t]$ for all $t \geq t_0$;

(ii) $V_{\text{perim},Q_\epsilon}(P[t+1]) \leq V_{\text{perim},Q_\epsilon}(P[t])$; and

(iii) the trajectory $\{P[t]\}_{t \in \mathbb{Z}_{\geq 0}}$ converges to a point $P^* \in Q_\epsilon$ such that either $p_i^* = p_j^*$ or $p_i^* \notin \mathcal{S}(p_j^*)$ for all $i, j \in \{1, \ldots, n\}$.

As a direct consequence of the theorem, note that if the graph $\mathcal{G}_{\text{sens}}$ is connected at any time during the evolution of the system, then all the robots converge to the same location in Q_ϵ.

4.6 Practical implementation issues

In Section 4.5, we designed a provably correct rendezvous algorithm for an ideal model of *point* robots with *perfect sensing and actuation capabilities*. Also, we assumed that it was possible for

the robots to operate *synchronously*. However, such an ideal model is not realistic in practical situations. In this section, we investigate, via extensive computer simulations, the effects of deviations from this ideal scenario.

4.6.1 Nominal experimental set-up

The computer simulation was written in C++ using the Computational Geometry Algorithmic Library (CGAL) (`http://www.cgal.org`). However, it was found that Boolean operations on polygons using the utilities present in CGAL were not adequate in terms of speed for the purpose of running extensive simulations. Hence, Boolean operations on polygons were performed using the General Polygon Clipping Library (GPC) (`http://www.cs.man.ac.uk/~toby/alan/software/gpc.html`).

For the purpose of simulations, the environment considered is a typical floor plan; see Figure 4.1. Experiments were performed with 20 robots starting from 10 randomly generated initial conditions from a uniform distribution. For each initial condition, experiments were repeated 20 times. The environment size is roughly 80×70, the step size of a robot is taken as $s_{\max} = 0.5$ and the sensing radius $r = 30$. For simplicity, $\mathcal{G}_{\text{constr}}$ is taken to be the same as $\mathcal{G}_{\text{sens}}$. To utilize the ϵ-robust visibility notion in providing robustness to asynchronism and sensing and control errors, at each time instant, ϵ is set to be 0.97 times the ϵ at the previous time instant. Initially, ϵ is set equal to 3. In case a robot approaches a reflex vertex of the environment closely, it reduces its speed. This is done to reduce the risk of collision due to errors in sensing the exact location of the reflex vertex. We now describe the various assumptions we make on the model to simulate the actual implementation on physical robots followed by the respective simulation results. The algorithm performance is then evaluated based on the following three performance measures: (i) the average number of steps taken by the robots to achieve the rendezvous objective; (ii) the fraction of the edges of $\mathcal{G}_{\text{sens}}$ that are preserved at the end of the simulation; and (iii) the number of connected components of $\mathcal{G}_{\text{sens}}$ at the end of the simulation compared with the number of connected components at the initial time.

4.6.2 Robustness against asynchronism, sensing and control noise, and finite-size disk robot models

(A1) *Asynchronism:* The robots operate asynchronously, i.e., do not share a common processor clock. All the robots start operating at the same time. Each robot's clock speed is a random number uniformly distributed on the interval $[0.9, 1]$. At integral multiples of its clock speed, a robot wakes up, senses the positions of other robots within its sensing region and takes a step according to the `Perimeter Minimizing Algorithm`. Note that at the time when any given robot wakes up, there may be other robots that are moving. No sensing and control errors were introduced in the model. It is observed that under this asynchronous implementation, the performance of the algorithm is very similar to the synchronous implementation; see Figure 4.12. In the subsequent implementations, we assume the robots to operate synchronously.

(A2) *Distance error in sensing and control:* The visibility sensors measure the relative distance of another object according to the following multiplicative noise model. If d_{act} is the actual distance to the object, then the measured distance is given by $(1 + e_{\text{dist}})d_{\text{act}}$, where e_{dist} is a random variable uniformly distributed in the interval $[-0.1, 0.1]$. The objects to which distances are measured are other robots in the sensing range and the boundary of the environment. For simplicity, instead of measuring a sequence of points along the boundary, as a real range sensor does, we assume that only the vertices of the environment are measured and the sensed region is reconstructed from that information. The sensor error, therefore, occurs in the measurement of other robots and environment vertices. The actuators moving the robots are also subject to a multiplicative noise distance model with the same error parameter. The results are shown in Figure 4.13. It is observed that only in 1 out of the 10 initial conditions does the number of connected components of $\mathcal{G}_{\text{sens}}$ increase as compared to the number of connected components of $\mathcal{G}_{\text{sens}}$ initially. In all cases, the fraction of edges preserved is almost equal to 1. Also, in 7 out of 10 cases, the performance is almost identical to the synchronized implementation with no noise. In the subsequent simulations, we assume the robots to operate synchronously with no distance error in sensing and control.

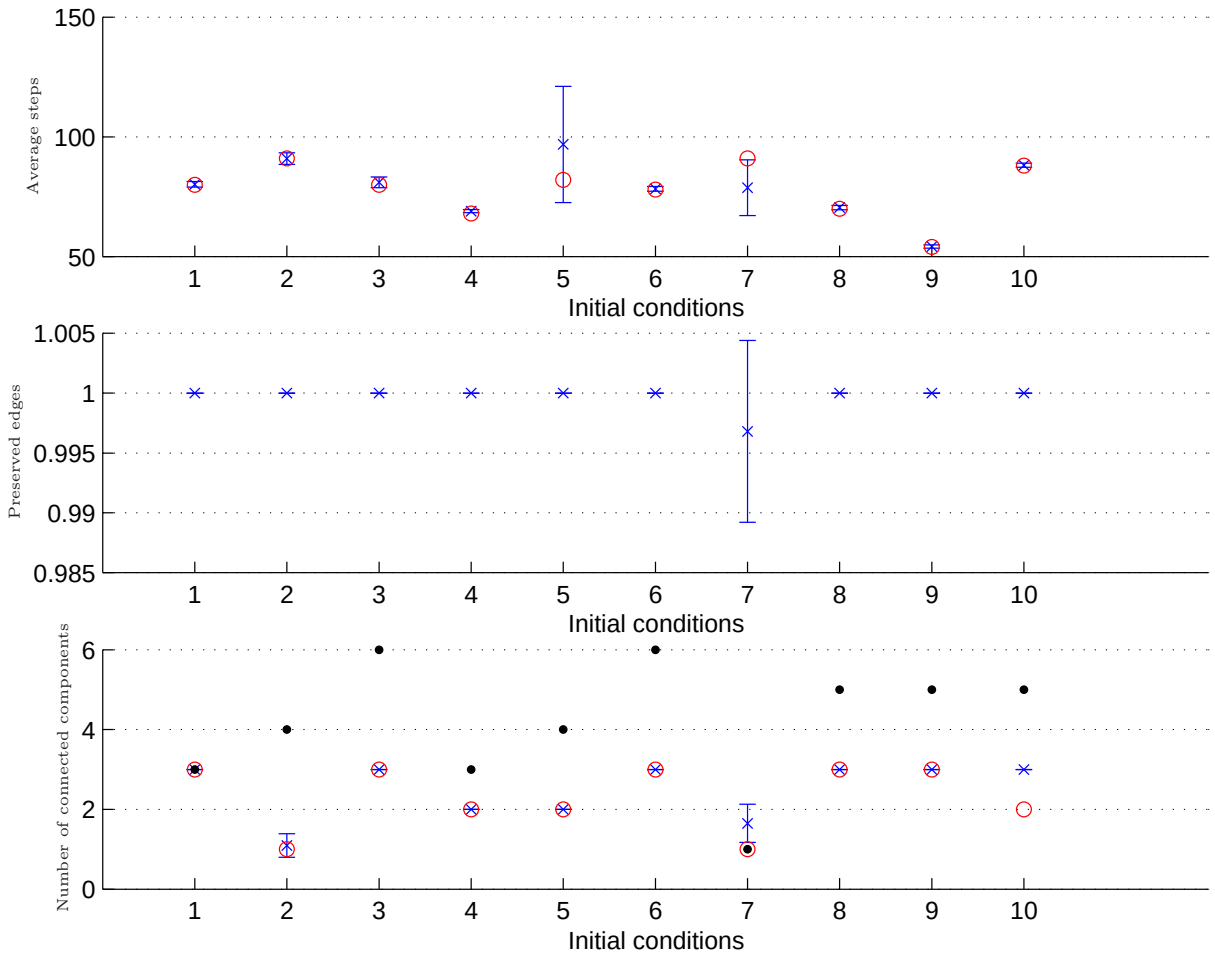


Figure 4.12: Computer simulation results with asynchronism. The top figure represents the average number of steps taken per robot for convergence(crosses). Also shown is the number of steps taken by each robot in synchronous implementation (red circle). The center figure shows the fraction of the edges of the initial sensing graph that are preserved till the end. The bottom figure shows the number of connected components of the sensing graph initially (small black discs) and at the end of asynchronous (blue crosses) and synchronous (red circle) implementations. The blue crosses in the figure denote the mean of the observed quantities and the vertical bars denote standard deviation.

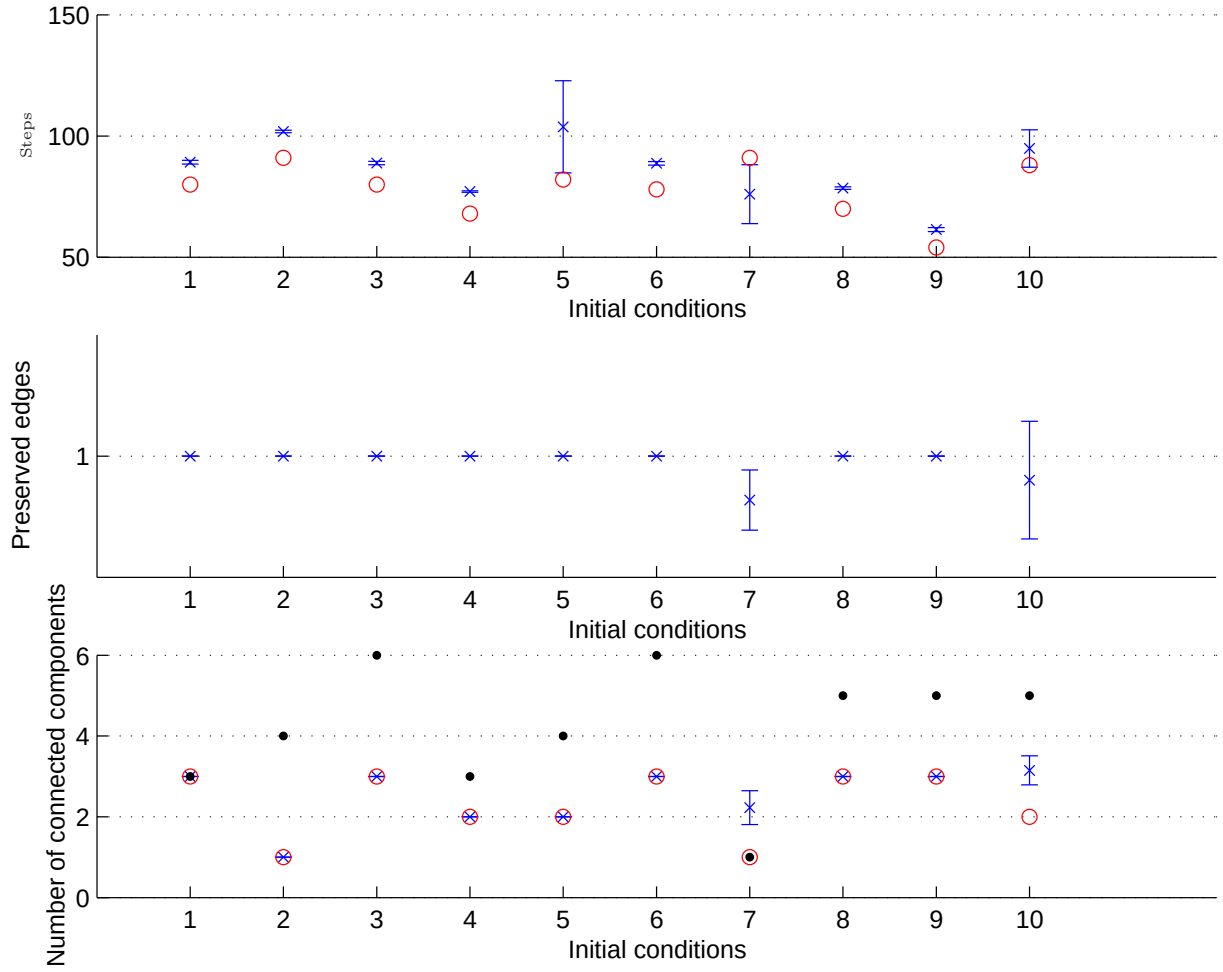


Figure 4.13: Computer simulation results with distance error and no directional error in sensing and control.The meaning of the quantities is as in Figure 4.12.

(A3) *Direction error in sensing and control:* The visibility sensors measure the relative angular location of another object according to the following additive noise model. If θ_{act} is the actual angular location of any object in the local reference frame of a robot, the measured angular location is given by $\theta_{\text{act}} + e_\theta$, where e_θ is a random variable uniformly distributed in the interval $[-5, 5]$. As before, the actuators moving the robots are also subject to an additive noise directional model with the same error parameter. The results are shown in Figure 4.14. It is observed that the algorithm no longer performs similar to the synchronized implementation with no noise. Thus, it is more sensitive to directional errors than distance errors in sensing and control. However, almost all of the edges of $\mathcal{G}_{\text{sens}}$ are preserved for all the initial conditions. Also, in 9 out of the 10 initial conditions the number of connected components of $\mathcal{G}_{\text{sens}}$ decreases during the course of evolution of the group. In the following simulations, no asynchronism or sensing and control errors are assumed.

(A4) *Disk robot model:* The robots are assumed to be disks of radius 0.2, but do not obstruct the views of others. During any step, a robot moves a distance of at most $s_{\max} = 0.5$. The next position of the center of the robot, therefore, lies in a *motion disc* of radius 0.7 centered at the center of the robot; see the red and green discs in Figure 4.15. A *colliding neighbor* of a robot i is any neighbor j according to $\mathcal{G}_{\text{sens}}$ such that the motion discs of i and j intersect and that the motion disc of j intersects the physical disc of i on the path between i and its next point. If a robot has no colliding neighbors, then its motion is according to `Perimeter Minimizing Algorithm`. If on the other hand a robot has exactly one colliding neighbor, then it tries to swerve around it while reducing its speed. Finally, if the robot has more than two colliding neighbors then it stays at the current location. To ensure free movement of the robots inside the environment, ϵ is not allowed to fall below 0.2, i.e., the radius of a robot disc. Simulations were performed without any asynchronism or sensing and control errors for the 10 initial conditions of the robots as in the previous experiments.The terminating condition for the simulations is that robots belonging to each connected component of $\mathcal{G}_{\text{sens}}$ form a "cohesive" group[6]; see Figure 4.15. For all initial conditions, the number of cohesive groups in the final configurations is equal to the number of connected components of $\mathcal{G}_{\text{sens}}$

[6]For each connected component of $\mathcal{G}_{\text{sens}}$, the graph having nodes as the robot locations and with an edge between two nodes whenever the corresponding motion discs of the robots intersect is connected

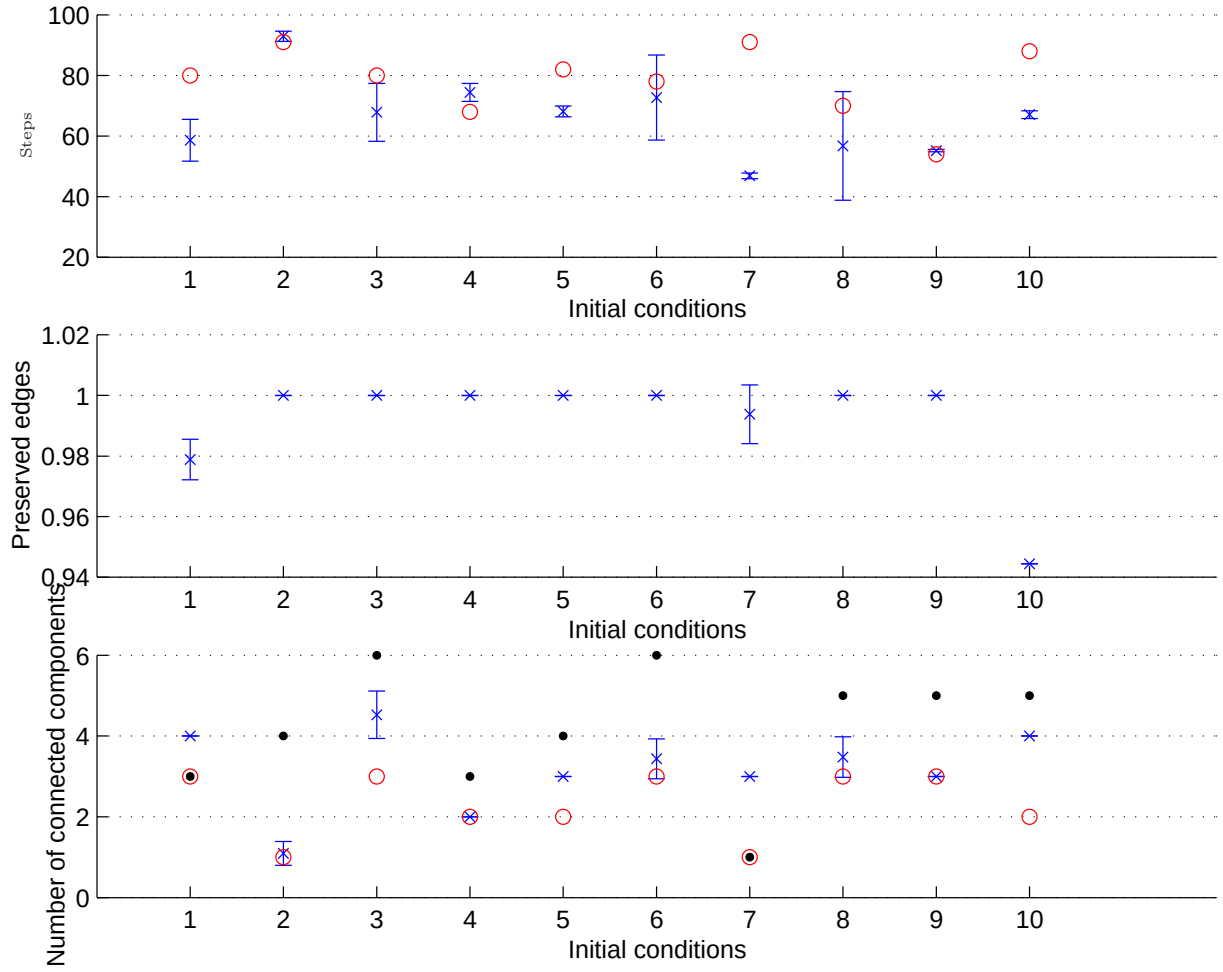


Figure 4.14: Computer simulation results with direction error and no distance error in sensing and control. The meaning of the quantities is as in Figure 4.12.

when the simulations are performed assuming point robots. Thus, the algorithm yields the same performance if a disk robot model is assumed instead of a point robot model.

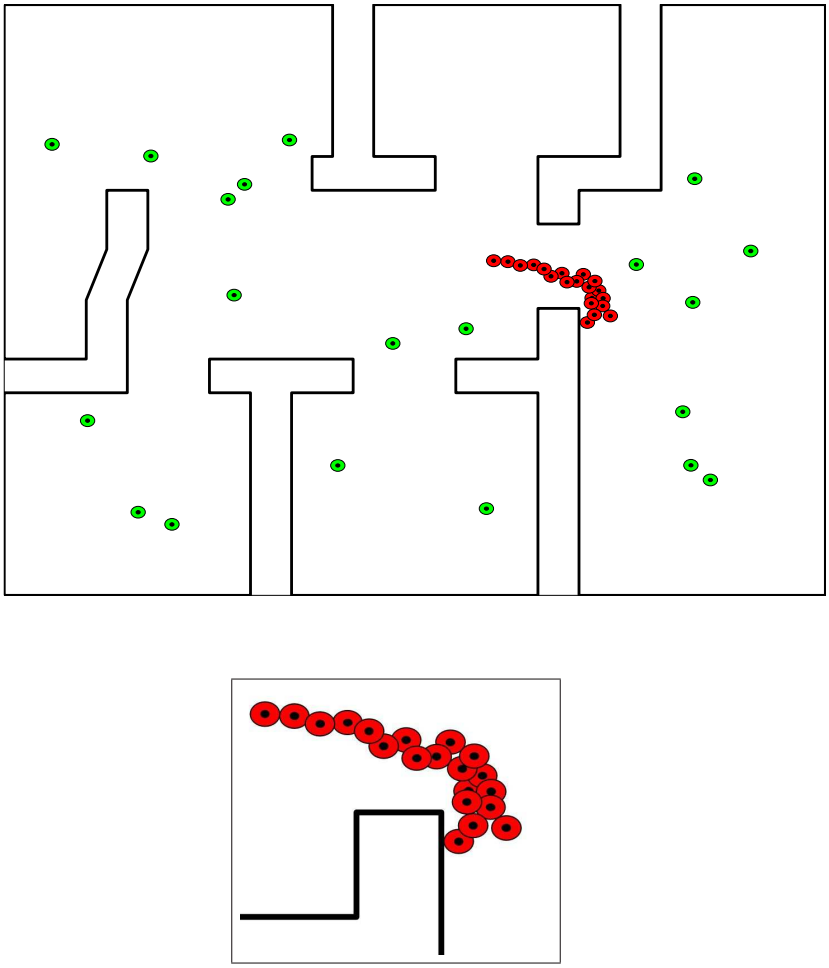


Figure 4.15: Computer simulation results with no asynchronism and no sensing and control errors. The black and red discs denote the robots and their motion discs respectively. The initial position of the robots correspond to initial condition 2 in Figures 4.12, 4.13 and 4.14 and are shown by the small black discs scattered over the environment with the green discs denoting their motion discs. The robots converge to positions corresponding to a single cohesive group part of the same component of $\mathcal{G}_{\text{sens}}$.

Thus, we see that the `Perimeter Minimizing Algorithm` is robust to various deviations from the ideal scenario. Also, the magnitudes of e_{dist} and e_θ are in line with the state-of-the-art. Finally, in the next section we analyze the computation complexity of the algorithm.

4.6.3 Computation complexity with finite resolution sensing

In addition to the issues that might arise in a practical implementation of the `Perimeter Minimizing Algorithm`, another important consideration is the time taken for a robot to complete each step of the algorithm. This is dependent on the computational complexity of the algorithm, that we characterize in the following.

A real visibility sensor, e.g., a range scanner, will sense the position of other robots and the boundary of the environment with some finite resolution; in particular, the boundary of the sensing region will be described by a set of points. It is reasonable to assume that the cardinality of this set of points is bounded, say by M, for all robots, irrespective of the shape of the environment and the location of the robot in it. For example, if a laser range sensor is used to measure the distance to the boundary and a measurement is taken at intervals of one degree, then M is equal to 360.

Proposition 4.6.1 (Computational complexity)**.** *Let Q be any allowable environment. Let M be the resolution of the visibility sensor located at any robot in Q. Then the following statements are true:*

(i) The computation complexity of the `Constraint Set Generator Algorithm` *is $O(\kappa M)$;*

(ii) The computation complexity of the `Perimeter Minimizing Algorithm` *is $\tau(M)+O(M^3 \log M)$;*

(iii) If $\mathcal{G}_{\text{constr}} = \mathcal{G}_{\text{sens}}$, then the computation complexity of the `Perimeter Minimizing Algorithm` *is $O(M^2 \log M)$,*

where $\tau(M)$ is time taken for the computation of $\mathcal{N}_{i,\mathcal{G}_{\text{constr}}}$ given the set $\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}$ assuming $|\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}| \leq M$, and κ is the number of strict concavities of Q.

As discussed in Section 4.4.2, if $\mathcal{G}_{\text{constr}} = \mathcal{G}_{\text{lc},\mathcal{G}_{\text{sens}}}$, then the computation of $\mathcal{G}_{\text{constr}}$ from $\mathcal{G}_{\text{sens}}$ can be performed using efficient polynomial time heuristics. The time $\tau(M)$ above depends on the specific heuristic used. Thus, we see that the running time of each step of the `Perimeter Minimizing Algorithm` is polynomial in the number of data points obtained by the visibility sensor.

This analysis helps us assess whether it is feasible to implement this algorithm on an actual robot without demanding an unreasonable computational power.

4.7 Conclusions

In this chapter, we present a provably correct discrete-time synchronous `Perimeter Minimizing Algorithm` algorithm for rendezvous of robots equipped with visibility sensors in a nonconvex environment. The algorithm builds on a novel solution to the connectivity maintenance problem also proposed in this chapter. The performance of the algorithm under asynchronous operation of the robots, presence of noise in sensing and control, and nontrivial robot dimension is investigated and found to be quite satisfactory. The computational complexity of the algorithm under the assumption of finite sensing resolution is also investigated.

4.8 Proofs of results in Section 4.4

Proof of Lemma 4.4.1: Let $d = \text{dist}(v_1, [p_i, p_j]) = \text{dist}(v_2, [p_i, p_j])$ and let $L = \{p \in \mathbb{R}^2 \mid \text{dist}(p, [p_i, p_j]) \leq d\}$. Then $v_1, v_2 \in \partial L$; see Figure 4.16. We now prove our result by contradiction. Let $v_1 \notin H_{Q_\epsilon}(v_2)$. Then $v_1 \in \partial L \setminus H_{Q_\epsilon}(v_2)$ as shown in Figure 4.16. Since v_1 is visible from p_i, the boundary ∂Q_ϵ

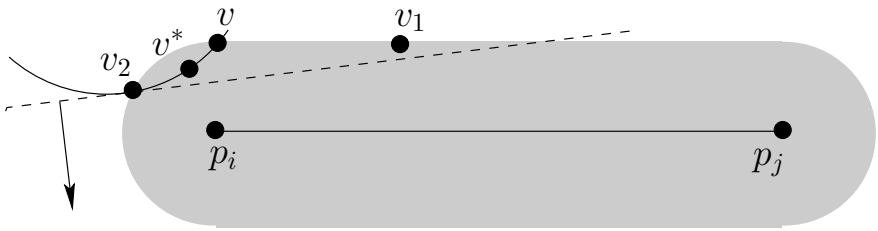


Figure 4.16: The shaded region represents L. The solid curve passing through v_2 represents a portion of the boundary of Q_ϵ. The dashed line is the boundary of $H_{Q_\epsilon}(v_2)$ which is along the tangent to ∂Q_ϵ at v_2. The interior of $H_{Q_\epsilon}(v_2)$ is in the direction of the arrow.

must intersect ∂L at a point v in between v_2 and v_1, as illustrated in Figure 4.16. But this means that there exists a point v^* belonging to the concavity containing v_2 that is strictly closer to the segment $[p_i, p_j]$ than v_2; this is a contradiction. ■

Proof of Proposition 4.4.2: We first prove statement (i). Note that initially $\mathcal{C}_{\text{temp}} = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. Therefore $\partial\mathcal{C}_{\text{temp}}$ has at most κ concavities. This is because the concavities of $\partial\mathcal{C}_{\text{temp}}$ are induced by the concavities of ∂Q_ϵ, and the number of concavities of ∂Q_ϵ is the same as the number of concavities of ∂Q. Now, by construction, the number of concavities in $\partial(\mathcal{C}_{\text{temp}} \cap H_{Q_\epsilon}(v))$ is strictly

less than the number of concavities of $\partial\mathcal{C}_{\text{temp}}$. It follows then that the `Constraint Set Generator Algorithm` terminates in at most κ steps. This completes the proof of statement (i).

Now note that $[p_i, p_j] \subseteq \mathcal{C}_Q(p_i, p_j)$. Hence $\mathcal{C}_Q(p_i, p_j)$ is always non-empty. Notice that $\mathcal{C}_Q(p_i, p_j)$ can be written in the following way

$$\mathcal{C}_Q(p_i, p_j) = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i + p_j}{2}, \frac{r}{2}) \cap H_{Q_\epsilon}(v_{i,j,1}) \ldots \cap H_{Q_\epsilon}(v_{i,j,n_{ij}}), \tag{4.6}$$

where $\text{dist}(v_{i,j,1}, [p_i, p_j]) \leq \ldots \leq \text{dist}(v_{i,j,n_{ij}}, [p_i, p_j])$. The $v_{i,j,k}$'s are computed according to step `3:` in the `Constraint Set Generator Algorithm`. Therefore, $\mathcal{C}_Q(p_i, p_j)$ is the result of the intersection of a finite number of closed sets, and therefore is closed as well. Also $\mathcal{C}_Q(p_i, p_j) \subseteq Q_\epsilon$ and is thus bounded. Therefore, it is compact. Finally, the `Constraint Set Generator Algorithm` terminates when $\mathcal{C}_{\text{temp}}$ has no concavities, or in other words, when $\mathcal{C}_{\text{temp}}$ is convex. This proves statement (ii).

To prove statement (iii), let us write $\mathcal{C}_Q(p_j, p_i)$ as

$$\mathcal{C}_Q(p_j, p_i) = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i + p_j}{2}, \frac{r}{2}) \cap H_{Q_\epsilon}(v_{j,i,1}) \ldots \cap H_{Q_\epsilon}(v_{j,i,n_{ji}}),$$

in the same way as (4.6). We first claim that $n_{ij} = n_{ji}$ and $\{v_{i,j,1}, \ldots, v_{i,j,n_{ij}}\} = \{v_{j,i,1}, \ldots, v_{j,i,n_{ji}}\}$. Before proving this claim, let us assume it is true and see where it leads. Because of this assumption, the expressions for $\mathcal{C}_Q(p_i, p_j)$ and $\mathcal{C}_Q(p_j, p_i)$ would differ only due to the terms $\mathcal{V}(p_i, \epsilon)$ and $\mathcal{V}(p_j, \epsilon)$. Now, let $q \in \mathcal{C}_Q(p_i, p_j)$. Because $\mathcal{C}_Q(p_i, p_j)$ is a convex subset of Q_ϵ containing p_j and q, the line segment $[p_j, q] \subseteq \mathcal{C}_Q(p_i, p_j) \subseteq Q_\epsilon$. Hence, $q \in \mathcal{V}(p_j, \epsilon)$. This in turn implies that $q \in \mathcal{C}_Q(p_j, p_i)$. Therefore, we have $\mathcal{C}_Q(p_i, p_j) \subseteq \mathcal{C}_Q(p_j, p_i)$. The opposite inclusion can be shown to be true in a similar fashion. Thus, we have $\mathcal{C}_Q(p_i, p_j) = \mathcal{C}_Q(p_j, p_i)$.

We now prove that $n_{ij} = n_{ji}$ and $\{v_{i,j,1}, \ldots, v_{i,j,n_{ij}}\} = \{v_{j,i,1}, \ldots, v_{j,i,n_{ji}}\}$. Note that $v_{i,j,1}$ is the strictly concave point nearest to $[p_i, p_j]$ belonging to $\mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. In general, there can be more than one nearest strictly concave point, say $v_{i,j,1}, \ldots, v_{i,j,s_{ij}}$. Let $d = \text{dist}(v_{i,j,1}, [p_i, p_j])$ and $L = \{p' \in \mathcal{C}_{\text{temp}} | \, \text{dist}(p', [p_i, p_j]) \leq d\}$, where $\mathcal{C}_{\text{temp}} = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. Note that L is convex since there cannot be any strictly concave points of $\partial\mathcal{C}_{\text{temp}}$ in the interior of L. Since $p_j, v_{i,j,1} \in L$ then $[v_{i,j,1}, p_j] \subseteq L \subseteq Q_\epsilon$. Therefore $v_{i,j,1} \in \mathcal{V}(p_j, \epsilon)$ and hence $v_{i,j,1} \in \mathcal{V}(p_j, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. There-

fore, $v_{i,j,1}, \ldots, v_{i,j,s_{ij}} \in \mathcal{V}(p_j, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. Similarly, if $v_{j,i,1}, \ldots, v_{j,i,s_{ji}}$ are the strictly concave points nearest to $[p_i, p_j]$ belonging to $\mathcal{V}(p_j, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$, then $v_{j,i,1}, \ldots, v_{j,i,s_{ji}} \in \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i+p_j}{2}, \frac{r}{2})$. Therefore $\text{dist}(v_{i,j,1}, [p_i, p_j]) = \text{dist}(v_{j,i,1}, [p_i, p_j])$ and $s_{ij} = s_{ji}$. This implies that $v_{j,i,k} \in \{v_{i,j,1}, \ldots, v_{i,j,s_{ij}}\}$ for all $k \in \{1, \ldots, s_{ji}\}$. Proceeding recursively, we get that $n_{ij} = n_{ji}$ and $\{v_{i,j,1}, \ldots, v_{i,j,n_{ij}}\} = \{v_{j,i,1}, \ldots, v_{j,i,n_{ji}}\}$. This completes the proof of statement (iii).

We now prove statement (iv). First, let us write

$$\mathcal{C}_Q(p_i, p_j) = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i + p_j}{2}, \frac{r}{2}) \bigcap \cap_{k=1}^{n_{ji}} H_{Q_\epsilon}(v_{j,i,k}).$$

Since Q is bounded, there exists $p_0 \in Q$ such that $\mathcal{C}_Q(p_i, \overline{p_j}) \subseteq Q \subseteq B(p_0, \text{diam}(Q))$. Therefore, we can write

$$\mathcal{C}_Q(p_i, p_j) = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i + p_j}{2}, \frac{r}{2}) \bigcap \cap_{k=1}^{n_{ji}} \left(H_{Q_\epsilon}(v_{j,i,k}) \cap B(p_0, \text{diam}(Q))\right).$$

The map $p_i \rightarrow \mathcal{V}(p_i, \epsilon)$ is upper semicontinuous at any point $p_i \in Q_\epsilon$ because there exists a neighborhood around any point p_i where the visibility set can only shrink. Also, the map $p_i \rightarrow \mathcal{V}(p_i, \epsilon)$ has a compact range. Thus the notions of upper semicontinuity and closedness are equivalent and, hence, $p_i \rightarrow \mathcal{V}(p_i, \epsilon)$ is closed at any point $p_i \in Q_\epsilon$. The second term is the closed ball of radius $\frac{r}{2}$ centered at the point $\frac{p_i+p_j}{2}$ and clearly depends continuously on p_i.

Let $a_1, \ldots, a_\kappa$ be the strict concavities of Q_ϵ. Now $v_{i,j,k}$ belongs to exactly one strict concavity, say a_{i_k}. Given a_{i_k}, $v_{i,j,k}$ is a function of (p_i, p_j). We can write $v_{i,j,k} := v_{i,j,k}(p_i, p_j) = \arg\min\{\text{dist}(v, [p_i, p_j]) \mid v \in a_{i_k}\}$. Since a_{i_k} is a continuous curve, it is clear that $v_{i,j,k}$ is a continuous function of (p_i, p_j). We now show that $H_{Q_\epsilon}(v_{i,j,k}(p_i, p_j)) \cap B(p_0, \text{diam}(Q))$ is in turn a set-valued map continuous with respect to (p_i, p_j). Since $v_{i,j,k}$ is a continuous function of (p_i, p_j), from Propositions 1 and 8 in [113], it suffices to show that $H_{Q_\epsilon}(v) \cap B(p_0, \text{diam}(Q))$ is continuous at any point v over the domain a_{i_k}. This is equivalent to showing that $H_{Q_\epsilon}(v) \cap B(p_0, \text{diam}(Q))$ is closed as well as open at any point $v \in a_{i_k}$. Without loss of generality, let us assume that $\partial H_{Q_\epsilon}(v)$ is parallel to the x axis as shown in Figure 4.17. Let us first show that $H_{Q_\epsilon}(v) \cap B(p_0, \text{diam}(Q))$ is open at any point $v \in a_{i_k}$. Let $\{v_l\}$ be a sequence in a_{i_k} such that $v_l \rightarrow v$, and let q be any point on $\partial H_{Q_\epsilon}(v)$. Let q_l be the point on $\partial H_{Q_\epsilon}(v_l)$ that is either vertically above or below q. Now

$\|q - q_l\| \leq |s(v_l)| \operatorname{diam}(Q)$, where $s(v_l)$ is the slope of the line defining the boundary of $H_{Q_\epsilon}(v_l)$. Since a_{i_k} is continuously differentiable, as $v_l \to v$, we have that $s(v_l) \to s(v) = 0$. Thus, $q_l \to q$. Hence, $H_{Q_\epsilon} \cap B(p_0, \operatorname{diam}(Q))$ is open at any point $v \in a_{i_k}$. We now show that $H_{Q_\epsilon} \cap B(p_0, \operatorname{diam}(Q))$ is closed at any point $v \in a_{i_k}$. As before, let $\{v_l\}$ be a sequence in a_{i_k} such that $v_l \to v$. Let $q_l \in H_{Q_\epsilon}(v_l) \cap B(p_0, \operatorname{diam}(Q))$ and let $q_l \to q$. But $\operatorname{dist}(q_l, H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q))) \leq |s(v_l)| \operatorname{diam}(Q)$ and $s(v_l) \to s(v) = 0$ as $v_l \to v$. Therefore, $\operatorname{dist}(q_l, H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q))) \to 0$. Since $q_l \to q$ and the distance of a point to the convex set $H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q))$ is a continuous function, we deduce $\operatorname{dist}(q_l, H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q))) \to \operatorname{dist}(q, H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q))) = 0$. Hence $q \in H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q)))$. Therefore $H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q)))$ is a closed map and, in

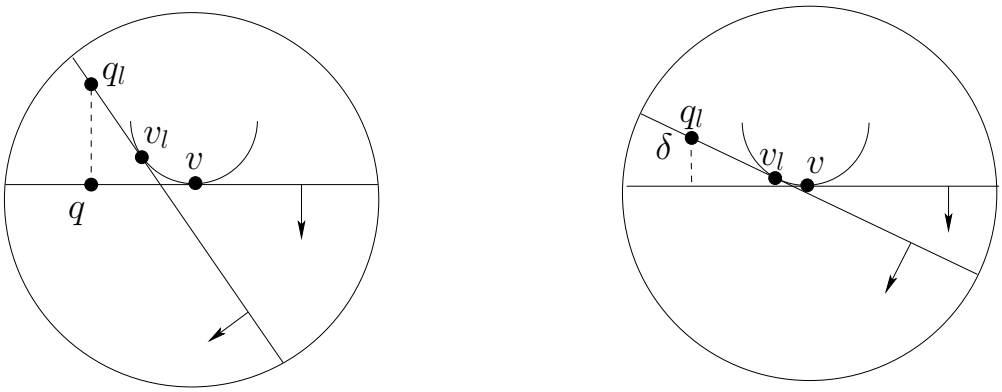


Figure 4.17: Illustration of various notions used in proving that $H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q))$ is continuous at any point v over the domain a_{i_k}. The circle represents the boundary of the set $B(p_0, \operatorname{diam}(Q))$. The arc represents a_{i_k}. The lines tangent to a_{i_k} at v and v_l are the boundaries of the half-planes $H_{Q_\epsilon}(v)$ and $H_{Q_\epsilon}(v_l)$ respectively. The interior of the half-planes is in the direction of the arrows. In the figure on the right, δ represents $\operatorname{dist}(q_l, H_{Q_\epsilon}(v) \cap B(p_0, \operatorname{diam}(Q)))$.

turn, is continuous over the domain a_{i_k}. This implies that $H_{Q_\epsilon}(v_{i,j,k}(p_i, p_j)) \cap B(p_0, \operatorname{diam}(Q)))$ is continuous at p_i for fixed a_{i_k}.

Now, let $(p_i^k, p_j^k) \to p_i$ and let $q^k \to q$ where $q^k \in \mathcal{C}_Q(p_i^k, p_j^k)$. We need to show that $q \in \mathcal{C}_Q(p_i, p_j)$. Since $q^k \in \mathcal{C}_Q(p_i^k, p_j^k)$, we have that $q^k \in \mathcal{V}(p_i^k, \epsilon)$. Since the map $p_i \to \mathcal{V}(p_i, \epsilon)$ is closed, we have that $q \in \mathcal{V}(p_i, \epsilon)$. Also, $q^k \in B(\frac{p_i^k + p_j^k}{2}, \frac{r}{2})$. Since the map $(p_i, p_j) \to B(\frac{p_i + p_j}{2}, \frac{r}{2})$ is continuous, it follows that $q \in B(\frac{p_i + p_j}{2}, \frac{r}{2})$. Therefore, what remains to be shown is that $q \in H_{Q_\epsilon}(v_{i,j,l}(p_i, p_j))$ for all $l \in \{1, \ldots, n_{ij}\}$. In what follows, for any points (p_i, p_j), (p_i^k, p_j^k) and (p_i', p_j'), let $v_{i,j,l}(p_i, p_j)$, $v_{i,j,l}(p_i^k, p_j^k)$ and $v_{i,j,l}(p_i', p_j')$ belong to the concavities a_{i_l}, $a_{i_l}^k$ and a_{i_l}' respectively and with l belonging to $\{1, \cdots, n_{ij}\}$, $\{1, \cdots, n_{ij}^k\}$ and $\{1, \cdots, n_{ij}'\}$ respectively. Since

$q^k \in H_{Q_\epsilon}(v_{i,j,l}(p_i^k, p_j^k))$, if there exists k_0 such that for all $k \geq k_0$, we have that $a_{i_l}^k \in \{a_{i_1}, \cdots, a_{i_{n_{ij}}}\}$, then we are done. Equivalently, it suffices to show that if a is any strict concavity such that $a \notin \{a_{i_1}, \cdots, a_{i_{n_{ij}}}\}$ then there exists a neighborhood $\mathcal{B}$ around p_i such that for all $p_i' \in \mathcal{B}$, $a \notin \{a'_{i_1}, \cdots, a'_{i_{n'_{ij}}}\}$. But if $a \notin \{a_{i_1}, \cdots, a_{i_{n_{ij}}}\}$, then a is in the exterior of at least one of $\mathcal{V}(p_i, \epsilon)$, $B(\frac{p_i+p_j}{2}, \frac{r}{2})$ and $H_{Q_\epsilon}(v_{i,j,l}(p_i, p_j))$. The existence of $\mathcal{B}$ now follows from the fact that the maps $p_i \to \mathcal{V}(p_i, \epsilon)$, $p_i \to B(\frac{p_i+p_j}{2}, \frac{r}{2})$ and $p_i \to H_{Q_\epsilon}(v_{i,j,l}(p_i)) \cap B(p_0, \operatorname{diam} Q)$ are all closed. ■

Proof of Lemma 4.4.5: The second inclusion in fact (i) is a direct consequence of the definition of $\mathcal{G}_{\text{lc},\mathcal{G}}$. We prove now that $\mathcal{G}_{\text{EMST},\mathcal{G}} \subseteq \mathcal{G}_{\text{lc},\mathcal{G}}$ by contradiction. Let $\mathcal{P}$ be any point set and assume, without loss of generality, that $\mathcal{G}(\mathcal{P})$ is connected (otherwise, the same reasoning carries over for each connected component of $\mathcal{G}(\mathcal{P})$). For simplicity, further assume that the distances $\|p_k - p_l\|$, $k, l \in \{1, \ldots, n\}$, $k \neq l$ are all distinct. This ensures that there is a unique Euclidean Minimum Spanning Tree of $\mathcal{G}(\mathcal{P})$. If this is not the case, the same reasoning exposed here carries through for each Euclidean Minimum Spanning Tree associated with $\mathcal{G}(\mathcal{P})$. Let $(p_i, p_j) \in \mathcal{E}_{\mathcal{G}_{\text{EMST},\mathcal{G}}}(\mathcal{P})$ and $(p_i, p_j) \notin \mathcal{E}_{\mathcal{G}_{\text{lc},\mathcal{G}}}(\mathcal{P})$. Since necessarily $(p_i, p_j) \in \mathcal{E}_{\mathcal{G}}(\mathcal{P})$, the latter implies that there exists a maximal clique $\mathcal{P}'$ of the edge (p_i, p_j) in $\mathcal{G}$ such that $(p_i, p_j) \notin \mathcal{E}_{\mathcal{G}_{\text{EMST}}}(\mathcal{P}')$. If we remove the edge (p_i, p_j) from $\mathcal{G}_{\text{EMST},\mathcal{G}}(\mathcal{P})$, the tree becomes disconnected into two connected components T_1 and T_2, with $p_i \in T_1$ and $p_j \in T_2$. Now, there must exist an edge $e \in \mathcal{E}_{\mathcal{G}_{\text{EMST}}}(\mathcal{P}')$ with one vertex in T_1 and the other vertex in T_2 and with length strictly less than $\|p_i - p_j\|$. To see this, let $\{e_1, \ldots, e_d\}$ be the edges of $\mathcal{G}_{\text{EMST}}(\mathcal{P}')$ obtained in incremental order by running Prim's algorithm (e.g., see [47]) starting from the vertex p_i. Because p_i is in T_1 and p_j is in T_2, there must exist at least an edge in $\{e_1, \ldots, e_d\}$ with one vertex in T_1 and the other vertex in T_2. Let $s \in \{1, \ldots, d\}$ be such that e_s is the first edge having one vertex in T_1 and another vertex in T_2. Since $(p_i, p_j) \notin \mathcal{E}_{\mathcal{G}_{\text{EMST}}}(\mathcal{P}')$, according to Prim's algorithm, the length of e_s must be strictly less than $\|p_i - p_j\|$ (otherwise, the edge (p_i, p_j) will be part of the Euclidean Minimum Spanning Tree of $\mathcal{P}'$). If we add the edge e_s to the set of edges of $T_1 \cup T_2$, the resulting graph G is acyclic, connected and contains all the vertices $\mathcal{P}$, i.e., G is a spanning tree. Moreover, since the length of e_s is strictly less than $\|p_i - p_j\|$ and T_1 and T_2 are induced subgraphs of $\mathcal{G}_{\text{EMST},\mathcal{G}}(\mathcal{P})$, we conclude that G has shorter length than $\mathcal{G}_{\text{EMST},\mathcal{G}}(\mathcal{P})$, which is a contradiction. Fact (ii) is a consequence of fact (i). ■

4.9 Additional analysis results and proof of Theorem 4.5.4

In this section we provide some key technical results that help establish Theorem 4.5.4. We first present the technical results on which the proof depends.

Lemma 4.9.1 (Properties of the relative convex hull)**.** *For any allowable Q, Q_ϵ, the following statements hold:*

(i) if $\mathcal{P}_1, \mathcal{P}_2 \subseteq Q$, then $\mathrm{rco}(\mathcal{P}_1, Q) \subseteq \mathrm{rco}(\mathcal{P}_1 \cup \mathcal{P}_2, Q)$;

(ii) if $\mathrm{rco}(P', Q)$ is a strict subset of $\mathrm{rco}(P'', Q)$, then $V_{\mathrm{perim},Q}(P') < V_{\mathrm{perim},Q}(P'')$;

(iii) if $\mathcal{P} \subset Q_\epsilon$ and $\mathcal{G}(\mathcal{P}) \subseteq \mathcal{G}_{\mathrm{sens}}(\mathcal{P})$, then for $p_i \in \mathcal{P}$, we have that $\{p_i\} \cap \mathrm{Ve}(\mathrm{rco}(\mathcal{P}, Q_\epsilon)) \subseteq \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}))$.

Proof. Statements (i) and (ii) are obvious from the definitions of the relative convex hull and its perimeter. To prove statement (iii), assume $p_i \notin \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}))$. Then, p_i belongs either to $\partial\,\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}) \setminus \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}))$ or to the interior of $\mathrm{co}(\mathcal{N}_{i,\mathcal{G}})$. In the former case, $p_i \in [p_k, p_l]$ where $p_k, p_l \in \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}))$; see Figure 4.18(left). Note that since $\mathcal{G} \subseteq \mathcal{G}_{\mathrm{sens}}$ and p_k, p_l are neighbors of p_i in $\mathcal{G}$, we have $[p_i, p_k] \subseteq Q_\epsilon$ and $[p_i, p_l] \subseteq Q_\epsilon$. Since $p_i \in [p_k, p_l]$, this implies that $[p_k, p_l] \subseteq Q_\epsilon$. Therefore $[p_k, p_l] \subseteq \mathrm{rco}(\mathcal{P}, Q_\epsilon)$ because the shortest path in Q_ϵ between p_k, p_l is contained in $\mathrm{rco}(\mathcal{P}, Q_\epsilon)$. Thus, p_i cannot be a vertex of $\mathrm{rco}(\mathcal{P}, Q_\epsilon)$. Let us now look at the case when p_i belongs

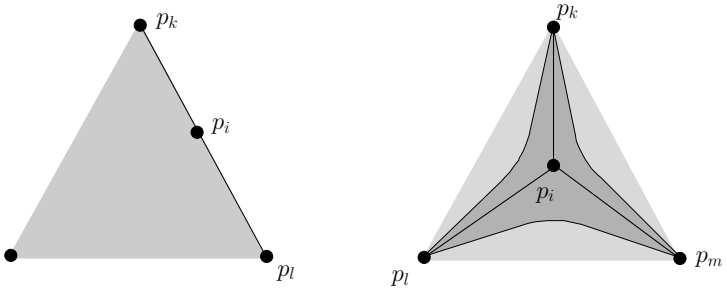


Figure 4.18: Illustration of the cases when $p_i \notin \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}))$. The point set represented by the black disks is $\mathcal{N}_{i,\mathcal{G}}$. The shaded region is $\mathrm{co}(\mathcal{N}_{i,\mathcal{G}})$. The straight line segments between any two points represent the fact that the two points are visible in Q_ϵ. The curved lines between pairs of points in the figure on the right represent shortest paths in Q_ϵ between the points.

to the interior of $\mathrm{co}(\mathcal{N}_{i,\mathcal{G}})$; see Figure 4.18(right). Then, since the strict concavities of Q_ϵ are all continuously differentiable curves, the shortest path between any two points $p_k, p_l \in \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}}))$

will be a sequence of straight lines and curves as shown in Figure 4.18(right). By the definition of the relative convex hull, the shortest paths between two points are contained in the relative convex hull. Thus the region bounded by the shortest paths between adjacent vertices of $\mathrm{co}(\mathcal{N}_{i,\mathcal{G}})$ is contained in the relative convex hull. Clearly, since p_i is the interior of this region, it cannot be a vertex of $\mathrm{rco}(\mathcal{P}, Q_\epsilon)$. This concludes the proof of statement (iii). □

See Figure 4.19 for a graphical explanation of Lemma 4.9.1.

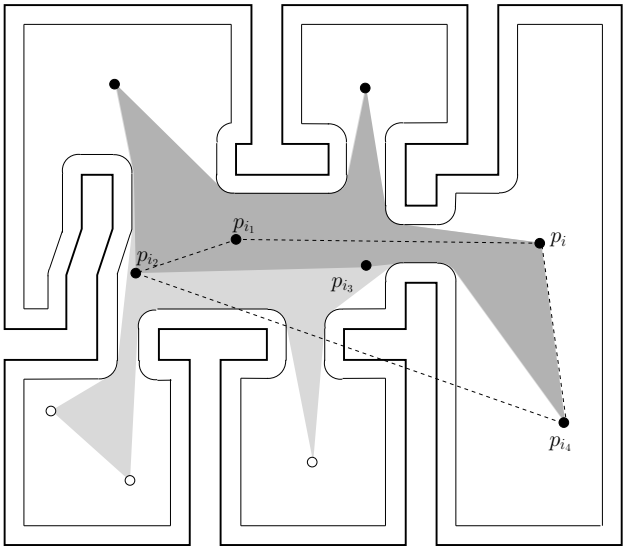


Figure 4.19: Illustration of Lemma 4.9.1. The outer polygonal environment is Q and the inner curved environment is Q_ϵ. The set of points represented by black and white disks are $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. Note that $\mathrm{rco}(\mathcal{P}_1, Q_\epsilon)$, represented by the dark shaded region, is a *subset* of $\mathrm{rco}(\mathcal{P}_1 \cup \mathcal{P}_2, Q_\epsilon)$, represented by the union of the dark and light shaded regions (c.f. Lemma 4.9.1 (i)). In fact $\mathrm{rco}(\mathcal{P}_1, Q_\epsilon)$ is a *strict subset* of $\mathrm{rco}(\mathcal{P}_1 \cup \mathcal{P}_2, Q_\epsilon)$ and hence $\mathrm{rco}(\mathcal{P}_1, Q_\epsilon)$ has a strictly smaller perimeter than $\mathrm{rco}(\mathcal{P}_1 \cup \mathcal{P}_2, Q_\epsilon)$ (c.f. Lemma 4.9.1 (ii)). Now, $p_i \in \mathrm{Ve}(\mathrm{rco}(\mathcal{P}_1 \cup \mathcal{P}_2))$. The set $\{p_i\} \cup \{p_{i_1}, \ldots, p_{i_4}\}$ is equal to the set $\mathcal{N}_{i,\mathcal{G}_{\mathrm{sens}}}$ where $\mathcal{G}_{\mathrm{sens}}$ is such that $r = +\infty$. Note that $p_i \in \mathrm{Ve}(\mathrm{co}(\mathcal{N}_{i,\mathcal{G}_{\mathrm{sens}}}))$ where $\mathrm{co}(\mathcal{N}_{i,\mathcal{G}_{\mathrm{sens}}})$ is the region bounded by the dashed lines (c.f. Lemma 4.9.1 (iii)).

Lemma 4.9.2 (Properties of the constraint set)**.** *For any allowable Q, Q_ϵ, if $P \in Q_\epsilon^n$ and $\mathcal{G}_2(P) \subseteq \mathcal{G}_1(P) \subseteq \mathcal{G}_{\mathrm{sens}}(P)$, then for all p_i, the following statements hold:*

(i) The set $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2}) \cap \mathrm{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$ is convex; and

(ii) $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1}) \cap \mathrm{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon)) = C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1}) \cap \mathrm{co}(\mathcal{N}_{i,\mathcal{G}_1})$.

Proof. Let $p, q \in C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2}) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. Since $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2})$ is an intersection of convex sets, it is convex as well. This implies that $[p, q] \subseteq C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2})$. Since $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2}) \subseteq Q_\epsilon$, we deduce that $[p, q] \subseteq Q_\epsilon$. Now, $p, q \in \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. Therefore, by the definition of the relative convex hull, the shortest path between p and q in Q_ϵ is also contained in $\text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. But the shortest path is the segment $[p, q]$ since $[p, q] \subseteq Q_\epsilon$. Thus, $[p, q] \subseteq \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. This completes the proof of statement (i).

To prove statement (ii), note that $\text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon)) \subseteq \text{co}(\mathcal{N}_{i,\mathcal{G}_1})$. Hence $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1}) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon)) \subseteq C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1}) \cap \text{co}(\mathcal{N}_{i,\mathcal{G}_1})$. To prove the other inclusion, let $q \in C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1}) \cap \text{co}(\mathcal{N}_{i,\mathcal{G}_1})$. Since $q \in \text{co}(\mathcal{N}_{i,\mathcal{G}_1})$, then q belongs to the closed triangle formed by p_i and two other points belonging to $\text{Ve}(\text{co}(\mathcal{N}_{i,\mathcal{G}_1}))$, say p_k, p_l; see Figure 4.20. Also $q \in C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1})$. From the construction of $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_1})$, it follows that $q \in \mathcal{V}(p_j, \epsilon)$ for all $p_j \in \mathcal{N}_{i,\mathcal{G}_1}$. In particular, this means that $[q, p_l] \subset Q_\epsilon$ and $[q, p_k] \subset Q_\epsilon$.

Since $p_i, p_l \in \mathcal{N}_{i,\mathcal{G}_1}$ and $[p_i, p_l] \subseteq \mathcal{V}(p_i, \epsilon)$, we deduce that $[p_i, p_l] \subseteq \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$ because the relative convex hull contains the shortest path in $\mathcal{V}(p_i, \epsilon)$ between any two points contained in it. Let the line containing the segment $[q, p_k]$ intersect the segment $[p_i, p_l]$ at q'. Note that $q \in [q', p_k]$ because by choice of p_k, p_l, we have that q belongs to the closed triangle with vertices p_i, p_k, p_l. Now, since $q' \in [p_i, p_l]$ and $[p_i, p_l] \subseteq \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$, then $q' \in \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. We now claim that the segment $[q', p_k] \subseteq \mathcal{V}(p_i, \epsilon)$. To see this, note that $[q', p_k]$ is a subset of the polygon given by the ordered set of vertices (p_i, p_k, q, p_l). Since all the edges of the polygon, $[p_i, p_k]$, $[p_k, q]$, $[q, p_l]$, $[p_l, p_k]$, are a subset of Q_ϵ, we have that the interior of Q_ϵ and the interior of the polygon (p_i, p_k, q, p_l) do not intersect. Thus $[q', p_k] \subseteq \mathcal{V}(p_i, \epsilon)$. Now, since $q', p_k \in \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$, the shortest path between q', p_k in $\mathcal{V}(p_i, \epsilon)$ is also contained in $\text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. Thus $q \in [q', p_k] \subseteq \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. This completes the proof of statement (ii). □

Statement (i) in the above lemma states that the constraint set is convex. Statement (ii) gives an alternate way of computing the set X_i in step `3:` of the `Perimeter Minimizing Algorithm` when $\mathcal{G}_{\text{constr}}$ and $\mathcal{G}_{\text{sens}}$ are identical.

In what follows, we analyze the `Perimeter Minimizing Algorithm` by means of the *closed*

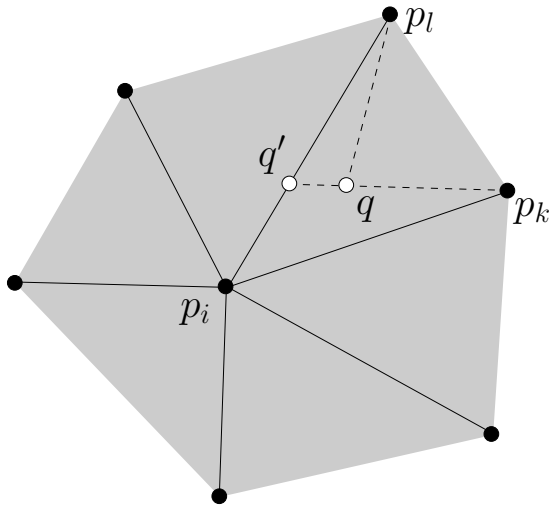


Figure 4.20: The set of points represented by black disks is $\mathcal{N}_i$. The shaded region represents $\mathrm{co}(\mathcal{N}_i)$. q, represented by the white disk, is any point in $\mathrm{co}(\mathcal{N}_i)$. Since q is ϵ-robustly visible from every point in $\mathcal{N}_i$, the dashed line segments are a subset of Q_ϵ. Also every point in $\mathcal{N}_i \setminus \{p_i\}$ is visible from p_i by the definition of $\mathcal{N}_i$. Hence the solid line segments are also a subset of Q_ϵ.

perimeter minimizing algorithm over any proximity graphs $\mathcal{G}_2 \subseteq \mathcal{G}_1 \subseteq \mathcal{G}_{\text{sens}}$, defined as follows:

$$
\begin{aligned}
T^c_{\mathcal{G}_1,\mathcal{G}_2}(P)_i &\in p_i + u_i, \quad \text{where} \\
X_i(P) &= C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2}) \cap \mathrm{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i,\epsilon)), \\
K_i(P) &= \Big\{ \lim_{k_m \to +\infty} \mathrm{CC}(X_i(P_{k_m})) \mid P_k \to P, \\
&\quad \{\mathrm{CC}(X_i(P_{k_m})\} \text{ is a convergent subseq. of } \{\mathrm{CC}(X_i(P_k)\}\Big\}, \\
u_i &\in \left\{ \frac{\min(s_{\max}, \|p_i^* - p_i\|)}{\|p_i^* - p_i\|}(p_i^* - p_i) \;\middle|\; p_i^* \in K_i(P) \right\},
\end{aligned}
\tag{4.7}
$$

and where $T^c_{\mathcal{G}_1,\mathcal{G}_2}(P)_i$ is the ith component of $T^c_{\mathcal{G}_1,\mathcal{G}_2}(P)$.

Note that $T_{\mathcal{G}_{\text{sens}},\mathcal{G}_{\text{constr}}}(P) \subseteq T^c_{\mathcal{G}_1,\mathcal{G}_2}(P)$ if $\mathcal{G}_1 = \mathcal{G}_{\text{sens}}$ and $\mathcal{G}_2 = \mathcal{G}_{\text{constr}}$. Therefore, the evolution under the `Perimeter Minimizing Algorithm` given by the trajectory $\{P[t]\}_{t\in\mathbb{Z}_{\geq 0}}$ with $P[t+1] \in T_{\mathcal{G}_{\text{sens}},\mathcal{G}_{\text{constr}}}(P[t])$ is just one of the possible evolutions under $T^c_{\mathcal{G}_1,\mathcal{G}_2}$ given by the trajectory $\{P^c[t]\}_{t\in\mathbb{Z}_{\geq 0}}$ with $P^c[t+1] \in T^c_{\mathcal{G}_1,\mathcal{G}_2}(P^c[t])$. We now begin stating the intermediate results for $T^c_{\mathcal{G}_1,\mathcal{G}_2}$. The first of these results essentially states that for at least one robot located at a vertex of the relative convex hull, the set X_i is large enough for it to move under the application of $T^c_{\mathcal{G}_1,\mathcal{G}_2}$. This, in turn, implies that the relative convex hull shrinks.

Lemma 4.9.3 (Nontrivial constraints)**.** *Let Q, Q_ϵ be allowable environments. Assume that $P \in Q_\epsilon^n$ and the proximity graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ satisfy*

A) not all points in P are equal; and

B) $\mathcal{G}_2(P) \subseteq \mathcal{G}_1(P) \subseteq \mathcal{G}_{\text{sens}}(P)$ and $\mathcal{G}_2(P)$ is connected.

Then there exists $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$ such that $\text{diam}(X_i) > 0$, where $X_i = C_{p_i, \mathcal{N}_{i,\mathcal{G}_2}} \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$.

Proof. Note that $p_i \subseteq X_i$ and X_i is convex. Thus, $\text{diam}(X_i) = 0$ if and only if $X_i = \{p_i\}$. The remainder of the proof is organized as follows. We first establish some necessary conditions for $X_i = \{p_i\}$. We then show that under the assumptions in the lemma, those necessary conditions cannot hold for every $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$. Note that $X_i = C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2}) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, Q_\epsilon))$ and $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_2}) = \bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} \mathcal{C}_Q(p_i, p_j)$. For any $p_j \in \mathcal{N}_{i,\mathcal{G}_2}, \mathcal{C}_Q(p_i, p_j) = \mathcal{V}(p_i, \epsilon) \cap B(\frac{p_i + p_j}{2}, \frac{r}{2}) \bigcap_{k=1}^{n_{ij}} H_{Q_\epsilon}(v_{i,j,k})$ where $v_{i,j,k}$'s are computed at step `3:` of the `Constraint Set Generator Algorithm`. Thus, we can write

$$X_i = \mathcal{V}(p_i, \epsilon) \cap \left(\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} \left(B(\frac{p_i + p_j}{2}, \frac{r}{2}) \cap \left(\bigcap_{k=1}^{n_{ij}} H_{Q_\epsilon}(v_{i,j,k}) \right) \right) \right) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, Q_\epsilon)),$$

which can be rewritten as

$$X_i = \mathcal{V}(p_i, \epsilon) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, Q_\epsilon)) \cap \left(\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} B(\frac{p_i + p_j}{2}, \frac{r}{2}) \right) \cap \left(\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} \bigcap_{k=1}^{n_{ij}} H_{Q_\epsilon}(v_{i,j,k}) \right). \tag{4.8}$$

Since $\text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon)) \subseteq \mathcal{V}(p_i, \epsilon)$, we have that $\mathcal{V}(p_i, \epsilon) \cap \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, Q_\epsilon)) = \text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. From assumptions (A) and (B), any robot i will have a neighbor in $\mathcal{G}_1(P)$ and $\mathcal{G}_2(P)$ that is not placed at p_i. Hence, $\mathcal{N}_{i,\mathcal{G}_1}$ and $\mathcal{N}_{i,\mathcal{G}_2}$ are point sets strictly larger than $\{p_i\}$. Therefore, $\text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$ is strictly larger than $\{p_i\}$. Therefore, there exists a neighborhood around p_i in $\text{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$, as shown by the shaded region in Figure 4.23. Let this neighborhood be $\mathcal{B}$.

Therefore, if p_i belongs to the interior of $\left(\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} B(\frac{p_i+p_j}{2}, \frac{r}{2}) \right) \cap \left(\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} \bigcap_{k=1}^{n_{ij}} H_{Q_\epsilon}(v_{i,j,k}) \right)$, then X_i will be strictly larger than $\{p_i\}$. Hence, if $X_i = \{p_i\}$ then p_i belongs to the boundary of one or more of the sets $B(\frac{p_i+p_j}{2}, \frac{r}{2})$ and $H_{Q_\epsilon}(v_{i,j,k})$, where $p_j \in \mathcal{N}_{i,\mathcal{G}_2}$ and $k \in \{1, \ldots, n_{ij}\}$.

If $\|p_i - p_j\| < r$, then p_i belongs to the interior of $B(\frac{p_i+p_j}{2}, \frac{r}{2})$; see Figure 4.21 (left). Also, if $v_{i,j,k} \notin [p_i, p_j]$ then p_i belongs to the interior of $H_{Q_\epsilon}(v_{i,j,k})$; see Figure 4.21 (right). Therefore, if p_i belongs to the boundary of $H_{Q_\epsilon}(v_{i,j,k})$, then $v_{i,j,k} \in [p_i, p_j]$ and if p_i belongs to the boundary of $B(\frac{p_i+p_j}{2}, \frac{r}{2})$, then $\|p_i - p_j\| = r$.

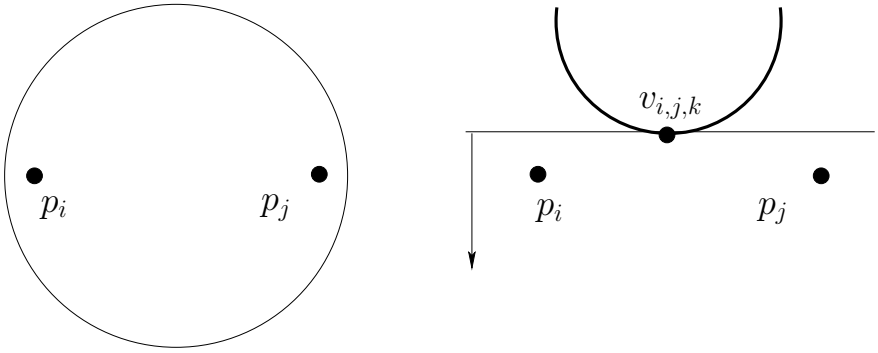


Figure 4.21: In the figure on the left, $\|p_i - p_j\| < R$. The circle represents $B(\frac{p_i+p_j}{2}, \frac{r}{2})$. In the figure on the curved line represents a strict concavity. $v_{i,j,k}$ represents the point on the concavity nearest to $[p_i, p_j]$. It can be seen that $\text{dist}(v_{i,j,k}, [p_i, p_j]) > 0$. The half-plane tangent to the strict concavity at $v_{i,j,k}$ and with interior in the direction of the arrow is $H_{Q_\epsilon}(v_{i,j,k})$.

Since $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$, from Lemma 4.9.1 (iii) we have that $p_i \in \text{Ve}(\text{co}(\mathcal{N}_{i,\mathcal{G}_2}))$. This implies that $\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} B(\frac{p_i + p_j}{2}, \frac{r}{2})$ is a convex set strictly containing $\{p_i\}$; see Figure 4.22.

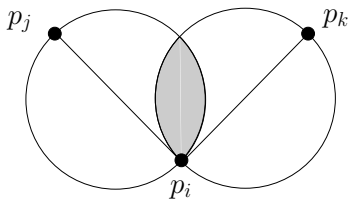


Figure 4.22: p_i is a vertex of the convex hull of the set $\{p_i, p_j, p_k\}$. Here $\|p_i - p_j\| = \|p_i - p_k\| = r$. Note that the angle between the vectors $p_j - p_i$ and $p_k - p_i$ is strictly less than π. The circles represent $B(\frac{p_i+p_j}{2}, \frac{r}{2})$ and $B(\frac{p_i+p_j}{2}, \frac{k}{2})$ and the shaded region represents their intersection. If $\|p_i - p_j\| < r$ or $\|p_i - p_k\| < r$, the region of intersection will still strictly contain $\{p_i\}$.

Therefore $\bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_2}} B(\frac{p_i + p_j}{2}, \frac{r}{2}) \cap \mathcal{B}$ strictly contains $\{p_i\}$. Therefore, p_i must belong to the boundary of some $H_{Q_\epsilon}(v_{i,j,k})$. It is easy to see that $H_{Q_\epsilon}(v_{i,j,k}) \cap \mathcal{B}$ is strictly larger than $\{p_i\}$. Therefore, p_i must belong to the boundary of another set $H_{Q_\epsilon}(v_{i,k,l})$ or $B(\frac{p_i+p_j}{2}, \frac{r}{2})$ as explained in Figure 4.23.

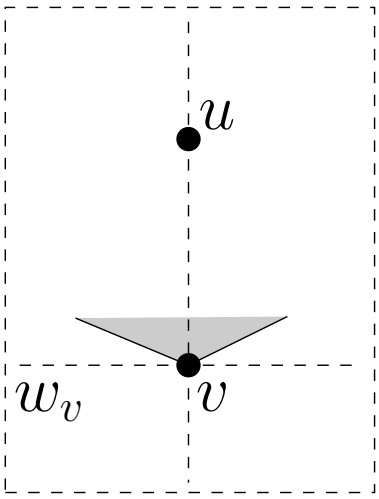


Figure 4.23: The thick lines represent the boundary of $\mathrm{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$. The shaded region is a subset of $\mathrm{rco}(\mathcal{N}_{i,\mathcal{G}_1}, \mathcal{V}(p_i, \epsilon))$ and forms an entire neighborhood around p_i. $H_{Q_\epsilon}(v_{i,j,l})$ and $H_{Q_\epsilon}(v_{i,k,m})$ are the half-planes with boundary being the lines passing through $[p_i, p_j]$ and $[p_i, p_k]$ respectively and interior in the direction of the arrows. In the figure on the right, $\|p_j - p_i\| = r$ and the angles between the vectors $p_j - p_i$ and $p_k - p_i$ is greater than $\frac{\pi}{2}$. The circle represents $B(\frac{p_i+p_k}{2}, \frac{r}{2})$.

Therefore, for $p_i \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$, we obtain the following necessary condition for $\mathrm{diam}(X_i) = 0$. There exist at least two neighbors of p_i located at p_j, p_k such that one or more than one of following cases hold:

(A) $\|p_j - p_i\| = r$, $v_{i,k,l} \in [p_i, p_k]$ for some $l \in \{1, \ldots, n_{ik}\}$, the inner product $\langle p_j - p_i, p_k - p_i \rangle \leq 0$ and $p_j \notin H_{Q_\epsilon}(v_{i,k,l})$;

(B) $v_{i,j,m} \in [p_i, p_j]$, $v_{i,k,l} \in [p_i, p_k]$ and $p_k \notin H_{Q_\epsilon}(v_{i,j,m})$, $p_j \notin H_{Q_\epsilon}(v_{i,k,l})$ for some $m \in \{1, \ldots, n_{ij}\}$ and some $l \in \{1, \ldots, n_{ik}\}$.

Cases (A) and (B) correspond to Figure 4.24 left and right respectively.

We now show that the necessary condition cannot hold for every $p_i \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$. Let $p_{i_1} \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$. Then p_{i_1} has at least two neighbors in $\mathcal{G}_2(P)$ located at p_j, p_k such that conditions (A) and/or (B) are satisfied. Now $p_{i_1} \in \partial\,\mathrm{rco}(P, Q_\epsilon)$ because $p_{i_1} \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$. Also, $v_{i,k,l} \in \partial Q_\epsilon \cap \mathrm{rco}(P, Q_\epsilon)$. Hence $v_{i,k,l} \in \partial\,\mathrm{rco}(P, Q_\epsilon)$. Now, since Q_ϵ does not contain any hole, $\mathrm{rco}(P, Q_\epsilon)$ is simply connected. Thus the segment $[p_{i_1}, v_{i,k,l}]$ partitions $\mathrm{rco}(P, Q_\epsilon)$ into two simply connected sets closed sets, Q'_{i_1} and Q''_{i_1} such that $Q'_{i_1} \cap Q''_{i_1} = [p_{i_1}, v_{i,k,l}]$. Let $p_k \in Q'_{i_1}$.

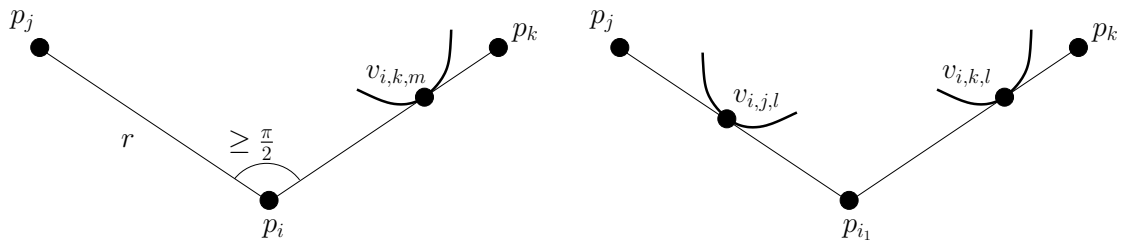


Figure 4.24: Illustration of cases (A) and (B) in the proof of Lemma 4.9.3. These cases illustrate the necessary conditions in order for the constraint set X_i to be a point.

Let us construct a segment $[v_{i,k,l}, d]$ by extending the segment $[p_{i_1}, v_{i,k,l}]$ till it intersects the boundary of $\text{rco}(P, Q_\epsilon)$ at d. We refer to Figure 4.25 for an illustration of this argument. Then the segment $[v_{i,k,l}, d]$ partitions $\text{rco}(P, Q_\epsilon)$ into two components such that again there exists $p'_{i_1} \in \text{Ve}(P, Q_\epsilon)$ in the component not containing p_{i_1}. We point that $p'_{i_1} = v_{i,k,l}$, $p'_{i_1} = p_k$ and $v_{i,k,l} = d$ are possible. Now, p'_{i_1} can only contain one neighbor in Q''_{i_1} and that can only be p_{i_1} since all other points in Q''_{i_1} are not ϵ-robust visible from p'_{i_1}.

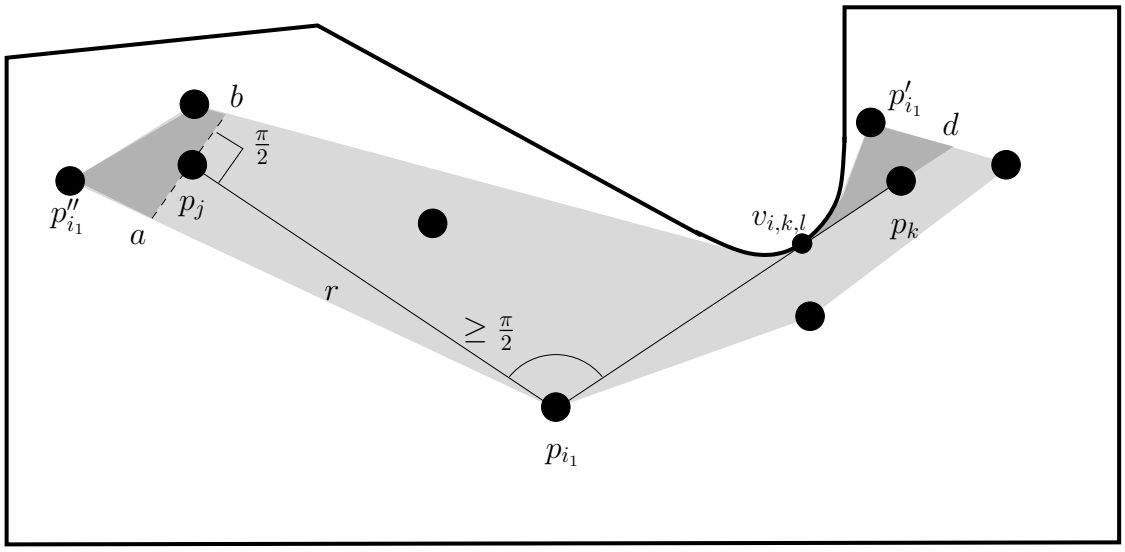


Figure 4.25: Illustration of the construction of various segments and the partitions induced by them in the proof of Lemma 4.9.3.

Let p_j be such that $\|p_j - p_{i_1}\| = r$ and the inner product $\langle p_j - p_{i_1}, p_k - p_{i_1} \rangle \leq 0$. Let us construct a line segment $[a, b]$ perpendicular to the segment $[p_{i_1}, p_j]$, passing through p_j and intersecting the boundary of $\text{rco}(P, Q_\epsilon)$ at points a and b. The segment $[a, b]$ partitions $\text{rco}(P, Q_\epsilon)$ into two components such that there exists $p''_{i_1} \in \text{Ve}(P, Q_\epsilon)$ in the component not containing p_{i_1}. Here we

point that $p''_{i_1} = p_j$ is possible. Now, p''_{i_1} can only contain one neighbor in Q'_{i_1} and that can only be p_{i_1} since all other points in Q'_{i_1} are at a distance strictly greater than r.

If, on the other hand, p_j is such that there exists $v_{i,j,m} \in [p_i, p_j]$ and $p_k \notin H_{Q_\epsilon}(v_{i,j,m})$, $p_j \notin H_{Q_\epsilon}(v_{i,k,l})$, then we proceed in a way similar to the case of p_k. Let us construct a line segment $[v_{i,j,m}, e]$ by extending the line segment $[p_i, v_{i,j,m}]$ till it intersects the boundary of $\mathrm{rco}(P, Q_\epsilon)$ at e. Then the segment $[v_{i,j,m}, e]$ partitions $\mathrm{rco}(P, Q_\epsilon)$ into two components such that again there exists $p''_{i_1} \in \mathrm{Ve}(P, Q_\epsilon)$ in the component not containing p_{i_1}. We point that $p''_{i_1} = v_{i,j,m}$, $p''_{i_1} = p_j$ and $v_{i,j,m} = e$ are possible. Now, as before p''_{i_1} can only contain one neighbor in Q'_{i_1} and that can only be p_{i_1} since all other points in Q'_{i_1} are not ϵ-robust visible from p''_{i_1}.

Thus, $X_{i_1} = \{p_{i_1}\}$ implies that there exists a partition of $\mathrm{rco}(P, Q_\epsilon)$ into Q'_{i_1} and Q''_{i_1} containing $p'_{i_1} \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$ and $p''_{i_1} \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$ respectively such that p'_{i_1} has only one neighbor in Q''_{i_1} and vice versa. Now, let $p_{i_2} = p''_{i_1}$. Therefore, $p_{i_2} \neq p_{i_1}$. Again $X_{i_2} = \{p_{i_2}\}$ implies that $\mathrm{rco}(P, Q_\epsilon)$ can be partitioned into Q'_{i_2} and Q''_{i_2}. Let $p_{i_1} \in Q'_{i_2}$. Therefore, there exists $p''_{i_2} \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$ such that p_{i_1} cannot be a neighbor of p''_{i_2}. Let $p_{i_3} = p''_{i_2}$. Therefore, $X_{i_2} = \{p_{i_2}\}$ implies the existence of $p_{i_3} \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$ such that $p_{i_3} \notin \{p_{i_1}, p_{i_2}\}$. Let the cardinality of $\mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$ be n_v. Proceeding recursively, $X_{i_{n_v}} = \{p_{i_{n_v}}\}$ implies the existence of $p_{i_{n_v+1}} \notin \{p_{i_1}, \ldots, p_{i_{n_v}}\}$. This is a contradiction. □

Now, let $G_1(P)$ and $G_2(P)$ be graphs with fixed topologies and let G_2 be a subgraph of G_1. Let $Y_{G_1,G_2} = \{P \in Q_\epsilon^n \mid G_2(P) \subseteq G_1(P) \subseteq \mathcal{G}_{\mathrm{sens}}(P)\}$.

We now present some results on smoothness of $T^c_{G_1,G_2}$ on Y_{G_1,G_2} and contraction of $V_{\mathrm{perim},Q_\epsilon}$ under the action of $T^c_{G_1,G_2}$. The technical approach in what follows is similar to the one in [21].

Lemma 4.9.4. (Properties of the `Perimeter Minimizing Algorithm`) *The set Y_{G_1,G_2} and the map $T^c_{G_1,G_2}$ have the following properties:*

(i) Y_{G_1,G_2} is compact, and $T^c_{G_1,G_2}$ is closed on Y_{G_1,G_2};

(ii) for all $i \in \{1, \ldots, n\}$, $T^c_{G_1,G_2}(P)_i \subseteq (\mathrm{rco}(P, Q_\epsilon) \setminus \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))) \cup \{p_i\}$;

(iii) if G_2 is connected, then there exists $p_i \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$ such that $T^c_{G_1,G_2}(P)_i \subseteq \mathrm{rco}(P, Q_\epsilon) \setminus \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$; and

(iv) $\text{rco}(T^c_{G_1,G_2}(P), Q_\epsilon)) \subseteq \text{rco}(P, Q_\epsilon)$ *at any* $P \in Y_{G_1,G_2}$,

where $T^c_{G_1,G_2}(P)_i$ *is the* i*th component of* $T^c_{G_1,G_2}(P)$.

Proof. We begin by proving statement (i). Showing that Y_{G_1,G_2} is compact is equivalent to showing that Y_{G_1,G_2} is closed and bounded. Since $Y_{G_1,G_2} \subseteq Q_\epsilon$, then Y_{G_1,G_2} is bounded. Now, let $\{P_k\}$ be any sequence in Y_{G_1,G_2} such that $P_k \to P$. Let us show that $P \in Y_{G_1,G_2}$. By definition, $G_2(P) \subseteq G_1(P)$. It remains to be shown that $G_1(P) \subseteq \mathcal{G}_{\text{sens}}(P)$. To see this, let (p_i, p_j) be an edge that does not belong to $\mathcal{G}_{\text{sens}}(P)$. We now show that it cannot belong to $G_1(P)$. Then, either $[p_i, p_j] \not\subset Q_\epsilon$ or $\|p_i - p_j\| > R$. If $[p_i, p_j] \not\subset Q_\epsilon$, we can construct neighborhoods around p_i and p_j such that for all p'_i and p'_j belonging to the neighborhoods, we have $[p'_i, p'_j] \not\subset Q_\epsilon$; see Figure 4.26. If, on the other hand,

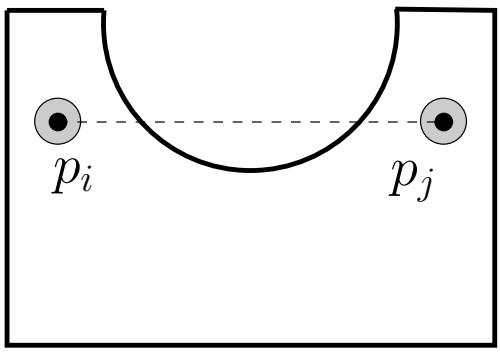


Figure 4.26: If $[p_i, p_j] \not\subset Q_\epsilon$, there exist neighborhoods of p_i and p_j such that $[p'_i, p'_j] \not\subset Q_\epsilon$, for all p'_i and p'_j belonging to the neighborhoods.

$\|p_i - p_j\| = R + \delta$, then let p'_i, p'_j be any points such that $\|p'_i - p_i\| < \frac{\delta}{2}$ and $\|p'_j - p_j\| < \frac{\delta}{2}$. Then $\|p'_i - p'_j\| = \|p'_i - p_i + p_i - p_j + p_j - p'_j\| \geq \|p_i - p_j\| - \|p'_i - p_i\| - \|p'_j - p_j\| > R + \delta - \frac{\delta}{2} - \frac{\delta}{2} = R$. Thus there exist neighborhoods around p_i and p_j such that for all p'_i and p'_j belonging to the neighborhoods, we have $\|p'_i - p'_j\| > R$. Therefore, we can construct a neighborhood around P in Q_ϵ such that for all P' in the neighborhood, (p'_i, p'_j) is not an edge of $\mathcal{G}_{\text{sens}}(P')$ if (p_i, p_j) is not an edge of $\mathcal{G}_{\text{sens}}(P)$. Since $P_k \to P$, there exists k_0 such that for all $k \geq k_0$, P_k belongs to this neighborhood. Therefore, for all $k \geq k_0$, (p^k_i, p^k_j) is not an edge of $\mathcal{G}_{\text{sens}}(P_k)$. Since $G_1(P_k) \subseteq \mathcal{G}_{\text{sens}}(P_k)$, for all $k \geq k_0$, (p^k_i, p^k_j) is not an edge of $G_1(P_k)$. Then, since $G_1(P)$ has the same topology as $G_1(P_k)$, we have that (p_i, p_j) is not an edge of $G_1(P)$. This completes the proof that Y_{G_1,G_2} is compact.

Showing that $T^c_{G_1,G_2}$ is closed over Y_{G_1,G_2} is equivalent to proving that the map $P \to K_i(P)$ is closed over Y_{G_1,G_2} for each $i \in \{1, \ldots, n\}$. Now, let $P_l \to P$ and let $q_l \to q$ where $q_l \in K_i(P_l)$.

We need to show that $q \in K_i(P)$.Since $q_l \in K_i(P_l)$, we have that $q_l = \lim_{m\to\infty} \mathrm{CC}(X_i(P'_{k_m}))$ for some sequence $P'_k \to P_l$. Hence, given $\delta > 0$, there exists $m_0(l)$ such that for all $m \geq m_0(l)$, we have $\|q_l - \mathrm{CC}(X_i(P'_{k_m}))\| \leq \delta$. Also, since $P'_k \to P_l$, there exists k_l such that for all $k \geq k_l$, we have $\|P'_k - P_l\| \leq \delta$. Without loss of generality, let us assume that $k_{m_0(l)} > k_l$. Now, let $P''_l = P'_{k_{m_0(l)}}$ and $q''_l = \mathrm{CC}(X_i(P''_l))$. Therefore, $\|P''_l - P_l\| \leq \delta$ and $\|q''_l - q_l\| \leq \delta$. Since $P_l \to P$ and $q_l \to q$, there exists l_0 such that for all $l \geq l_0$, we have that $\|P_l - P\| \leq \delta$ and $\|q_l - q\| \leq \delta$. But $\|P_l - P\| = \|P_l - P''_l + P''_l - P\| \geq \|P''_l - P\| - \|P_l - P"_l\|$. Therefore, $\|P''_l - P\| \leq \delta + \|P_l - P"_l\| \leq 2\delta$. Hence $P''_l \to P$. Similarly, it follows that for $l \geq l_0$. we have $\|q''_l - q\| \leq 2\delta$. Thus, $q''_l \to q$. It then follows from the definition of $K_i(P)$ that $q \in K_i(P)$.

We now prove statement (ii). It suffices to prove that $K_i(P) \subseteq (\mathrm{rco}(P, Q_\epsilon) \setminus \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))) \cup \{p_i\}$. Let $q \in K_i(P)$. Therefore, $q = \lim_{k_m\to\infty} \mathrm{CC}(X_i(P_{k_m}))$ where $\{P_{k_m}\}$ is some subsequence of $P_k \to P$. We begin by showing that the map $P \to X_i(P)$ is closed at P. Note that $X_i(P) = C_{p_i,Q}(\mathcal{N}_{i,G2}) \cap \mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon))$ and that $C_{p_i,Q}(\mathcal{N}_{i,G2}) = \cap_{p_j \in \mathcal{N}_{i,G_2}} \mathcal{C}_Q(p_i, p_j)$. If $p_j \in \mathcal{N}_{i,G_2}$, then (p_i, p_j) is an edge of $G_2(P) \subseteq \mathcal{G}_{\mathrm{sens}}(P)$. This implies that $(p_i, p_j) \in J$. But from Lemma 4.4.2 (iv), the map $(p_i, p_j) \to \mathcal{C}_Q(p_i, p_j)$ is closed over J. Therefore, the map $P \to \mathcal{C}_Q(p_i, p_j)$ is closed over Y_{G_1,G_2} for each $p_j \in \mathcal{N}_{i,G_2}$. Since G_2 has fixed topology, we have that the set of indices $\{j \mid p_j \in \mathcal{N}_{i,G_2}\}$ is constant over Y_{G_1,G_2}. Thus, the map $P \to C_{p_i,Q}(\mathcal{N}_{i,G2})$ is closed over Y_{G_1,G_2} since by Proposition 4 in [113], the intersection of closed maps is closed.

We now show that $\mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon)) = \mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$. Since $\mathcal{V}(p_i, \epsilon) \subseteq Q_\epsilon$, the inclusion $\mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon)) \subseteq \mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$ follows directly. Let q_1, $q_2 \in \mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon))$. Therefore, q_1, $q_2 \in \mathcal{V}(p_i, \epsilon)$. By definition the shortest path in $\mathcal{V}(p_i, \epsilon)$ between q_1 and q_2 is contained in $\mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon))$. Since Q_ϵ does not contain any holes, the shortest path between q_1 and q_2 in $\mathcal{V}(p_i, \epsilon)$ is also the shortest path between q_1 and q_2 in Q_ϵ. Therefore, by definition of a relative convex set, $\mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon))$ is convex relative to Q_ϵ. Since $\mathcal{N}_{i,G_1} \subseteq \mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon))$ and $\mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$ is the intersection of all relative convex set containing $\mathcal{N}_{i,G_1}$, we have that $\mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon) \subseteq \mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon))$. This proves that $\mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon)) = \mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$.

Now, since the shortest distances between two points in Q_ϵ change continuously as the position of the points, the relative convex hull of the points also changes continuously as a function of the points. Again the set $\{j \mid p_j \in \mathcal{N}_{i,G_1}\}$ is constant over Y_{G_1,G_2}. Thus, the map $P \to \mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$

is continuous over Y_{G_1,G_2} and hence, also closed. Finally, since $X_i(P)$ is the intersection of closed maps, we have that $P \to X_i(P)$ is closed.

Then from the definition of a closed map $q \in X_i(P)$. But $X_i(P) \subseteq \text{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, \epsilon)) = \text{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$ and since $\mathcal{N}_{i,G_1} \subseteq \{p_i \mid \{1, \ldots, n\}\}$, we have that $\text{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon) \subseteq \text{rco}(P, Q_\epsilon)$. Thus, $X_i(P) \subseteq \text{rco}(P, Q_\epsilon)$ and hence $q \in \text{rco}(P, Q_\epsilon)$. Now, let $q \in \text{Ve}(\text{rco}(P, Q_\epsilon))$. We show that then $\text{diam}(X_i(P)) = 0$. Our strategy is as follows. We first show that if $q \in \text{Ve}(\text{rco}(P, Q_\epsilon))$, then $\text{diam}(X_i(P_{k_m})) \to 0$. But that in turn implies that $q = p_i$ and hence $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$. But $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$ and $\text{diam}(X_i(P)) > 0$ together imply that $\text{diam}(X_i(P_{k_m}))$ does not tend to zero. This is a contradiction. Therefore, $\text{diam}(X_i(P)) = 0$.

We now start by showing that $\text{diam}(X_i(P_{k_m})) \to 0$. The range of the map $P \to X_i(P)$ is compact, the notions of upper semicontinuity and closedness are identical. Therefore, $P \to X_i(P)$ is upper semicontinuous at any point $P \in Y_{G_1,G_2}$. It follows from the definition of upper semicontinuity that given any $\delta > 0$, there exists m_0 such that for all $m \geq m_0$, we have $X_i(P_{k_m}) \subseteq \cup_{x \in X_i(P)} B(x, \delta)$.

Since $X_i(P) \subseteq \text{rco}(P, Q_\epsilon)$, q is a strictly convex point on the boundary of $X_i(P)$. By a strictly convex point, we mean that there exists a tangent to $\partial X_i(P)$ at q which intersects $\partial X_i(P)$ at exactly one point. Let q_1 be the point on the boundary of $\cup_{x \in X_i(P)} B(x, \delta)$ such that $\|q_1 - q\| = \delta$; see Figure 4.27. It follows that q_1 is a strictly convex point on the boundary of $\cup_{x \in X_i(P)} B(x, \delta)$. Let l_1 be the tangent at q_1. Now, let l_2 be the tangent to $B(q, \delta)$ parallel to l_1. It is easy to see that as $\delta \to 0$, the region of $\cup_{x \in X_i(P)} B(x, \delta)$ between the lines l_1 and l_2 tends to the point set $\{q\}$. Now, since $\text{CC}(X_i(P_{k_m})) \to q$, there exists m_1 such that for all $m \geq m_1$, $\text{CC}(X_i(P_{k_m})) \in B(q, \delta)$. For $m \geq \max\{m_0, m_1\}$, let y be a point on the opposite side of l_2 as $\text{CC}(X_i(P_{k_m}))$ and equidistant to l_2 as $\text{CC}(X_i(P_{k_m}))$; see Figure 4.27. All points on the right of l_2 are nearer to y than to $\text{CC}(X_i(P_{k_m}))$. Also, the distance of y to any point in $\cup_{x \in X_i(P)} B(x, \delta)$ on or to the left of l_2 tends to 0 as $\delta \to 0$.

Let $x_1 = \arg\max_x \{\|x - \text{CC}(X_i(P_{k_m}))\| \mid x \in X_i(P_{k_m})\}$ and let $x_2 = \arg\max_x \{\|x - y\| \mid x \in X_i(P_{k_m})\}$. By definition of the circumcenter and the fact that it is unique, $\|x_1 - \text{CC}(X_i(P_{k_m}))\| < \|x_2 - y\|$. Also, by construction $\|x_1 - \text{CC}(X_i(P_{k_m}))\| \geq \|x_2 - \text{CC}(X_i(P_{k_m}))\|$. It follows then that $\|x_2 - y\| > \|x_2 - \text{CC}(X_i(P_{k_m}))\|$. Therefore, x_2 belongs to the left of l_2. But as pointed earlier, $\|x_2 - y\| \to 0$ as $\delta \to 0$. But the circumradius of $X_i(P_{k_m})$ is equal to $\|x_1 - \text{CC}(X_i(P_{k_m}))\|$

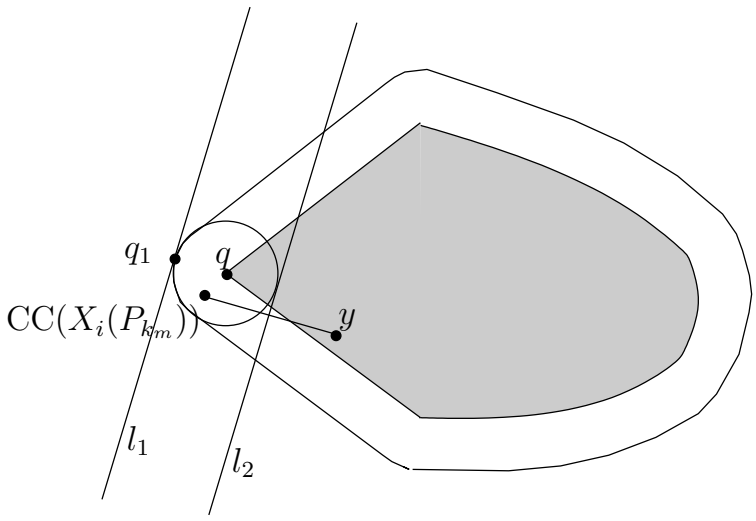


Figure 4.27: Illustration of notions used in the proof of Lemma 4.9.4. The shaded region represents $X_i(P)$. The outermost boundary represents the boundary of $\cup_{x \in X_i(P)} B(x, \delta)$. The small circle represents $B(q, \delta)$.

and $\|x_1 - \mathrm{CC}(X_i(P_{k_m}))\| < \|x_2 - y\|$. Since δ can be chosen arbitrarily small, it follows that $\|x_1 - \mathrm{CC}(X_i(P_{k_m}))\| \to 0$ as $m \to \infty$. This proves that if $q \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$, then $\mathrm{diam}(X_i(P_{k_m})) \to 0$.

Now let $p_i^{k_m}$ be the ith component of P_{k_m}. Since $p_i^{k_m}, \mathrm{CC}(X_i(P_{k_m})) \in X_i(P_{k_m})$ and $\mathrm{diam}(X_i(P_{k_m})) \to 0$, it follows that $\|p_i^{k_m} - \mathrm{CC}(X_i(P_{k_m}))\| \to 0$. But $P_{k_m} \to P$, and therefore, $p_i^{k_m} \to p_i$. This implies that $\mathrm{CC}(X_i(P_{k_m})) \to p_i$. Thus $p_i = q$ and $p_i \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$. We now claim that if $\mathrm{diam}(X_i(P)) > 0$ and $p_i \in \mathrm{Ve}(\mathrm{rco}(P, Q_\epsilon))$, then there exists a neighborhood of P in Y_{G_1,G_2} such that for all points P' in the neighborhood, we have that $\mathrm{diam}(X_i(P')) \geq d$ for some $d > 0$. But this is a contradiction since we have shown that $\mathrm{diam}(X_i(P_{k_m})) \to 0$ for some sequence $P_{k_m} \to P$.

To prove the above claim, note that since $\mathcal{V}(p_i, \epsilon) \subseteq \mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, Q_\epsilon))$ and $\mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, Q_\epsilon)) = \mathrm{rco}(\mathcal{N}_{i,G_1}, \mathcal{V}(p_i, Q_\epsilon))$, we can write

$$X_i = \mathrm{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon) \cap \left(\bigcap_{p_j \in \mathcal{N}_{i,G_2}} B(\frac{p_i + p_j}{2}, \frac{r}{2}) \right) \cap \left(\bigcap_{p_j \in \mathcal{N}_{i,G_2}} \bigcap_{k=1}^{n_{ij}} H_{Q_\epsilon}(v_{i,j,k}) \right). \tag{4.9}$$

Now, let us first consider the case when $X_i(P)$ has non-empty interior. Then one can choose a segment $[p', p'']$ in the interior of $X_i(P)$ whose length is arbitrarily close to the length of $\mathrm{diam}(X_i(P))$.

Therefore, $[p', p'']$ belongs in the interior of $\text{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$, $B(\frac{p_i+p_j}{2}, \frac{r}{2})$ and $H_{Q_\epsilon}(v_{i,j,k})$ for all i, j and k. As shown earlier in the proof of statement (ii), the map $p_i \rightarrow \text{rco}(\mathcal{N}_{i,G_1}(P), Q_\epsilon)$ is continuous for every i. Therefore, there exists a neighborhood $\mathcal{B}_1$ of $Y_{G_1.G_2}$ around P such that for all $P' \in \mathcal{B}_1$, the segment $[p', p''] \subseteq \text{rco}(\mathcal{N}_{i,G_1}(P'), Q_\epsilon)$.

From the proof of Proposition 4.4.2 (iv), we know that the map $(p_i, p_j) \rightarrow B(\frac{p_i+p_j}{2}, \frac{r}{2})$ is continuous over J. It follows then that for fixed i and j, there exists a neighborhood of J around (p_i, p_j) such that for all (p'_i, p'_j) in the neighborhood, $[p', p''] \subseteq B(\frac{p'_i+p'_j}{2}, \frac{r}{2})$. Since G_2 is fixed the set $\{j \mid j \in \mathcal{N}_{i,G_2}\}$ is fixed. Thus, there exists a neighborhood, $\mathcal{B}_2$ of Y_{G_1,G_2} of P such that for all $P' \in \mathcal{B}_2$, we have $[p', p''] \subseteq \left(\bigcap_{p_j \in \mathcal{N}_{i,G_2}} B(\frac{p_i+p_j}{2}, \frac{r}{2})\right)$. Also, from the proof of Proposition 4.4.2 (iv), for fixed j and a fixed strict concavity a, the map $(p_i, p_j) \rightarrow H_{Q_\epsilon}(v(p_i, p_j))$ is continuous over J, where $v(p_i, p_j)$ is the point on the strict concavity a nearest to the segment $[p_i, p_j]$. Again, from the proof of Proposition 4.4.2 (iv), there exists a neighborhood of J around (p_i, p_j) such that for all points (p'_i, p'_j) in the neighborhood, we have $a'_{i_l} \in \{a_{i_1}, \cdots, a_{i_{n_{ij}}}\}$ for any $l \in \{1, \ldots, n'_{ij}\}$ where the notation is as in the proof of Proposition 4.4.2 (iv). Let $a'_{i_l} = a_{i_s}$. Since $[p', p'']$ belongs to the interior of $H_{Q_\epsilon}(v_{i,j,s}(p_i, p_j))$, which in turn varies continuously as (p_i, p_j) for fixed a_{i_s}, there exists a neighborhood around (p_i, p_j) such that for all (p'_i, p'_j), we have $[p', p''] \subset H_{Q_\epsilon}(v_{i,j,l})(p'_i, p'_j)$. It then follows that there exists a neighborhood $\mathcal{B}_3$ of Y_{G_1,G_2} around P such that for all $P' = (p'_1, \cdots, p'_n) \in \mathcal{B}_3$, $[p', p''] \subset \left(\bigcap_{p_j \in \mathcal{N}_{i,G_2}(P')} \bigcap_{k=1}^{n'_{ij}} H_{Q_\epsilon}(v_{i,j,k}(p'_i, p'_j))\right)$. Thus, for any $P' \in \mathcal{B}_1 \cap \mathcal{B}_2 \cap \mathcal{B}_3$, we have that $[p', p''] \subseteq X_i(P')$ or that $\text{diam}(X_i(P')) \geq d$ where d is equal to the length of the segment $[p', p'']$.

Second, let $X_i(P)$ have empty interior. Therefore, $X_i(P)$ is a line segment and since $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$ we have that p_i is one of the end points of $X_i(P)$. Also, since $X_i(P)$ is a line segment, it follows from equation (4.9) that $X_i(P)$ must belong to the intersection of the boundaries of at least two of the sets $\text{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$ and the half-planes $H_{Q_\epsilon}(v_{i,j,k})$. The set $X_i(P)$ cannot belong to the boundary of a ball $B(\frac{p_i+p_j}{2}, \frac{r}{2})$ since its boundary is curved. Therefore, $X_i(P)$ belongs to the intersection of the boundaries of either (i) $\text{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon)$ and a half-plane $H_{Q_\epsilon}(v_{i,j,k})$ or (ii) two half-planes $H_{Q_\epsilon}(v_{i,j,k})$ and $H_{Q_\epsilon}(v_{i,l,m})$. Let us assume without loss of generality that $X_i(P)$ belongs to the interior of the remaining sets. Then, as shown earlier, there exists a neighborhood, $\mathcal{B}_4$, of Y_{G_1,G_2} around P such that for all P' in the neighborhood, $X_i(P)$ belongs to the interior of the remaining

sets. Now cases (i) and (ii) correspond to Figure 4.28 top and bottom respectively. It follows

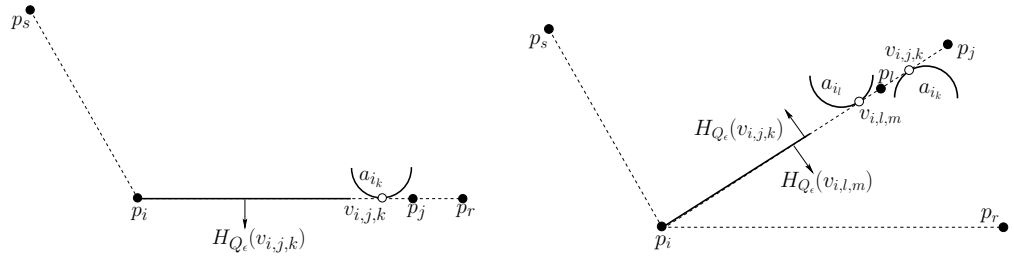


Figure 4.28: Illustration of cases (i) and (ii) in the proof of Lemma 4.9.4. The curved arcs represent strict concavities. The solid line segment represents $X_i(P)$. The dashed lines represent the boundary of $\text{rco}(P, Q_\epsilon)$. Note that there must exist robots p_r, $p_s \in \mathcal{N}_{i,G_1}$ for the dashed lines to be the boundary of $\text{rco}(P, Q_\epsilon)$. Note that $p_i \in \text{Ve}(RCH(P, Q_\epsilon))$ and also p_i is an end point of the segment $X_i(P)$. In the top figure $v_{i,j,k}$, denoted by the white disc, is the point on the strict concavity a_{i_k} nearest to the segment $[p_i, p_j]$. Here $p_j \in \mathcal{N}_{i,G_2}$. The half-plane $H_{Q_\epsilon}(v_{i,j,k})$ has interior in the direction of the arrow. In the bottom figure, $v_{i,l,m}$ and $v_{i,j,k}$ are points on a_{i_l} and a_{i_k} nearest to the segments $[p_i, p_l]$ and $[p_i, p_j]$ respectively. The direction of the arrows points towards the interior of the half-planes $H_{Q_\epsilon}(v_{i,l,m})$ and $H_{Q_\epsilon}(v_{i,j,k})$. Here p_j, $p_l \in \mathcal{N}_{i,G_2}$.

immediately from the figure that we can choose a neighborhood, $\mathcal{B}_5$, of P in Y_{G_1,G_2} such that for P' in the neighborhood, we have that the $\text{rco}(\mathcal{N}_{i,G_1}, Q_\epsilon) \cap H_{Q_\epsilon}(v_{i,j,k})$ and $H_{Q_\epsilon}(v_{i,j,k}) \cap H_{Q_\epsilon}(v_{i,l,m})$ contain a segment of length arbitrarily close to the length of $X_i(P)$. Therefore, for all $P' \in \mathcal{B}_4 \cap \mathcal{B}_5$, we have that $X_i(P')$ contains a segment of length d where $d > 0$.

This completes the proof of the fact that if $q \in \text{Ve}(\text{rco}(P, Q_\epsilon))$, then $\text{diam}(X_i(P)) = 0$. But then this implies that $q = p_i$. Thus, this completes the proof of statement (ii).

To prove statement (iii), note that in proving statement (ii) we have shown that if $\text{diam}(X_i(P)) > 0$, then $q = \lim_{k_m \to \infty} \text{CC}(X_i(P_{k_m})) \notin \text{Ve}(\text{rco}(P, Q_\epsilon))$. From Lemma 4.9.3, there exists at least one $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$ such that $\text{diam}(X_i(P)) > 0$. Statement (iii) now follows directly.

We now prove statement (iv). It follows from Statement (ii), that $T^c_{G_1,G_2}(P) \subseteq \text{rco}(P, Q_\epsilon)$. This implies that $\text{rco}(T^c_{G_1,G_2}(P), Q_\epsilon) \subseteq \text{rco}(P, Q_\epsilon)$. □

From Lemma 4.9.4(i), we know that, in particular, if the graphs $\mathcal{G}_{\text{sens}}, \mathcal{G}_{\text{constr}}$ have fixed topology, then the closed perimeter minimizing algorithm is a closed map. However, during the evolution of the system we expect the graphs $\mathcal{G}_{\text{sens}}, \mathcal{G}_{\text{constr}}$ to switch. To study the convergence properties of $T_{\mathcal{G}_{\text{sens}},\mathcal{G}_{\text{constr}}}$ over such a dynamic topology, we define a set-valued algorithm which essentially embeds all possible evolutions that can happen due to switching. Given any allowable Q, Q_ϵ, define

the *perimeter minimizing algorithm over all allowable graphs* to be the set-valued function T that maps points in Q_ϵ^n to all possible subsets of Q_ϵ by:

$$T(P) = \{P' \in T^c_{G_1,G_2}(P) \mid G_2 \subseteq G_1 \subseteq \mathcal{G}_{\text{sens}}(P), G_1, G_2 \text{ have fixed topologies and same connected components as } \mathcal{G}_{\text{sens}}\}.$$

Also, let $\mathcal{Y} = \{P \in Q_\epsilon^n \mid \mathcal{G}_{\text{sens}}(P) \text{ is connected}\}$ denote the set of all configurations where $\mathcal{G}_{\text{sens}}$ is connected.

Proposition 4.9.5. (Perimeter minimizing algorithm over all allowable graphs) *The set $\mathcal{Y}$ and set-valued map T have the following properties:*

(i) $\mathcal{Y}$ is compact, $T(\mathcal{Y}) \subseteq \mathcal{Y}$, and T is closed on $\mathcal{Y}$;

(ii) $\text{rco}(T(P), Q_\epsilon) \subseteq \text{rco}(P, Q_\epsilon)$*, for $P \in Q_\epsilon^n$; and*

(iii) if $P' \in T(P)$ and $P \in Q_\epsilon^n$, then $V_{\text{perim},Q_\epsilon}(P') \leq V_{\text{perim},Q_\epsilon}(P)$.

Proof. Since $\mathcal{Y} \subseteq Q_\epsilon^n$, we have that $\mathcal{Y}$ is bounded. Now, let $\{P_k\}$ be a sequence in $\mathcal{Y}$ such that $P_k \to P$. Then, from the proof of Lemma 4.9.4 (i), we know that there exists k_0 such that for all $k \geq k_0$, (p_i^k, p_j^k) is an edge of $\mathcal{G}_{\text{sens}}(P_k)$ implies that (p_i, p_j) is an edge of $\mathcal{G}_{\text{sens}}(P)$. But $\mathcal{G}_{\text{sens}}(P_k)$ is connected. Therefore, $\mathcal{G}_{\text{sens}}(P)$ is also connected. Hence, $P \in \mathcal{Y}$ and, therefore, $\mathcal{Y}$ is closed. This proves that $\mathcal{Y}$ is compact.

Now let $P' \in T(P)$ where $P \in \mathcal{Y}$. Then $P' \in T^c_{G_1,G_2}(P)$ for some G_1 and G_2 with fixed topologies having the same connected components as $\mathcal{G}_{\text{sens}}(P)$. Since $P \in \mathcal{Y}$, $\mathcal{G}_{\text{sens}}(P)$ is connected and hence, so are G_1, G_2. The fact that $P' \in \mathcal{Y}$ or that $\mathcal{G}_{\text{sens}}(P')$ has one connected component can be verified exactly along the lines of the proof of Theorem 4.5.4 (i). This proves that $T(\mathcal{Y}) \subseteq \mathcal{Y}$.

Now, let $\{\overline{P}_k\}$ be any sequence with $\overline{P}_k \in T(P_k)$ such that $\overline{P}_k \to \overline{P}$. Since $\overline{P}_k \to \overline{P}$, and the number of distinct graphs possible over n nodes is finite, there exists a subsequence $\overline{P}_{k_m} \to \overline{P}$ such that $\overline{P}_{k_m} \in T^c_{G_1,G_2}(P_{k_m})$ where $G_1(P_{k_m})$ and $G_2(P_{k_m})$ have fixed topologies for all m. Now, since $P_{k_m} \to P$, $\overline{P}_{k_m} \to \overline{P}$ and $T^c_{G_1,G_2}$ is closed over Y_{G_1,G_2} from Lemma 4.9.4 (i), we have that $\overline{P} \in T^c_{G_1,G_2}(P)$. Now, to finish the proof, we have to show that $G_2(P) \subseteq G_1(P) \subseteq \mathcal{G}_{\text{sens}}(P)$. The first inclusion follows from the fact the by definition of $T(P)$ where $G_2(P) \subseteq G_1(P)$. Now, notice

that we have shown in proof of Lemma 4.9.4 (i) that there exists k_0 such that for all $k \geq k_0$, (p_i^k, p_j^k) being an edge of $\mathcal{G}_{\text{sens}}(P_k)$ implies that (p_i, p_j) is an edge of $\mathcal{G}_{\text{sens}}(P)$. Also, by definition, $G_1(P_{k_m}) \subseteq \mathcal{G}_{\text{sens}}(P_{k_m})$. Therefore, for all $k_m \geq k_0$, we have that $(p_i^{k_m}, p_j^{k_m})$ being an edge of $G_1(P_{k_m})$ implies that (p_i, p_j) is an edge of $\mathcal{G}_{\text{sens}}(P)$. Therefore, $G_1(P) \subseteq \mathcal{G}_{\text{sens}}(P)$. This completes the proof of statement (i).

Statement (ii) follows directly from Lemma 4.9.4 (ii). Finally, statement (iii) is a consequence of statement (ii) and Lemma 4.9.1 (ii). □

We are now ready to present the proof of the main result in this chapter. The proof uses the analysis results presented in this section and the discrete time LaSalle Invariance Principle for set-valued maps [21].

Proof of Theorem 4.5.4:

We begin by proving statement (i). Let $p_i[t_0], p_j[t_0]$ be two robots belonging to the same connected component of $\mathcal{G}_{\text{sens}}(P[t_0])$. Then from Corollary 4.4.6(ii), they must belong to the same connected component of $\mathcal{G}_{\text{constr}}(P[t_0])$. Therefore, there exists a path between $p_i[t_0]$ and $p_j[t_0]$ in $\mathcal{G}_{\text{constr}}(P[t_0])$. Let $(p_k[t_0], p_l[t_0])$ be any edge of $\mathcal{G}_{\text{constr}}(P[t_0])$ that belongs to this path. Note that $u_k[t_0]$ and $u_l[t_0]$ in the `Perimeter Minimizing Algorithm` algorithm are constrained to belong to the sets $C_{p_k,Q}(\mathcal{N}_{j,\mathcal{G}_{\text{constr}}}) - p_k[t_0]$ and $C_{p_l,Q}(\mathcal{N}_{j,\mathcal{G}_{\text{constr}}}) - p_l[t_0]$, respectively. Therefore, $p_k[t_0+1], p_l[t_0+1] \in \mathcal{C}_Q(p_k[t_0], p_l[t_0]) \subseteq Q_\epsilon \cap B(\frac{p_k[t_0]+p_l[t_0]}{2}, \frac{r}{2})$. Hence, the edge $(p_j[t_0+1], p_k[t_0+1])$ is an edge of $\mathcal{G}_{\text{sens}}(P[t_0+1])$. Thus, there exists a path between $p_i[t_0+1]$ and $p_j[t_0+1]$ in $\mathcal{G}_{\text{sens}}(P[t_0+1])$, which proves statement (i).

For statements (ii) and (iii), it suffices to prove the results for the system evolving under the action of T, since the evolution under $T_{\mathcal{G}_{\text{sens}},\mathcal{G}_{\text{constr}}}$ is just one of the possible evolutions under T. Statement (ii), therefore, follows from Proposition 4.9.5(iii).

We now prove statement (iii). From statement (i), it follows that the number of connected components of $\mathcal{G}_{\text{sens}}(P[t])$ is nondecreasing. Since the number of possible connected components of $\mathcal{G}_{\text{sens}}$ is finite, this implies that, after a finite time $\bar{t}$, the number of connected components of $\mathcal{G}_{\text{sens}}$ does not change and robots belonging to different connected components never detect each other's presence. To prove statement (iii), it now suffices to show that any two robots belonging to the same connected component of $\mathcal{G}_{\text{sens}}$ converge to the same point. Since the evolution of any group of

connected robots from time $\bar{t}$ on is independent from the existence of other connected components, we can then assume, without loss of generality, that $\mathcal{G}_{\text{sens}}(P[\bar{t}])$ has just one connected component.

According to Lemma 4.5.3 and Proposition 4.9.5(iii), $V_{\text{perim},Q_\epsilon}$ is continuous and non-increasing along T on Q_ϵ^n. From Proposition 4.9.5(i), T is a closed map on the compact set $\mathcal{Y} \subset Q_\epsilon^n$. Then, by the LaSalle Invariance Principle for closed set-valued maps [21], $P[t] \to \mathcal{M}$, where $\mathcal{M}$ is the largest weakly positively invariant[7] set contained in

$$\{P \in \mathcal{Y} \mid \exists P' \in T(P) \text{ s.t. } V_{\text{perim},Q_\epsilon}(P') = V_{\text{perim},Q_\epsilon}(P)\}.$$

Let us define the set $\text{diag}(Q_\epsilon^n) = \{(p, \ldots, p) \in Q_\epsilon^n \mid p \in Q_\epsilon\} \subset \mathcal{Y}$, and show that $\mathcal{M} = \text{diag}(Q_\epsilon^n)$. Clearly, $\text{diag}(Q_\epsilon^n) \subseteq \mathcal{M}$. To prove the other inclusion, we reason by contradiction. Let $P \in \mathcal{M} \setminus \text{diag}(Q_\epsilon^n)$. Then, Lemma 4.9.4 (iii) implies that for any connected $G_2(P)$, $G_1(P)$ satisfying $G_2(P) \subseteq G_1(P) \subseteq \mathcal{G}_{\text{sens}}(P)$, there exists $p_i \in \text{Ve}(\text{rco}(P, Q_\epsilon))$ such that $T^c_{\mathcal{G}_1,\mathcal{G}_2}(P)_i \subseteq \text{rco}(P, Q_\epsilon) \setminus \text{Ve}(\text{rco}(P, Q_\epsilon))$. Therefore, if $P' \in T(P)$, then P' contains a number of robots $N(P')$ belonging to $\text{Ve}(\text{rco}(P, Q_\epsilon))$ that is strictly smaller than $N(P)$. In turn, this implies that after at most $N(P)$ steps, at least one vertex of $\text{rco}(P, Q_\epsilon)$ will contain any robot. By the definition of a vertex of $\text{rco}(P, Q_\epsilon)$, it follows that the relative convex hull of the configuration after at most $N(P)$ steps is a strict subset of $\text{rco}(P, Q_\epsilon)$. From Lemma 4.9.1(ii) in Appendix 4.9, $V_{\text{perim},Q_\epsilon}$ has strictly decreased. This is a contradiction with the fact that $\mathcal{M}$ is weakly positively invariant.

To finish the proof, we need to show that every trajectory converges to an individual point, i.e., $P[t] \to P^* \in \mathcal{M}$ for all initial conditions $P[0] \in \mathcal{Y}$. Note that $\{P[t]\}$ is a sequence in a compact set. Hence, it has a convergent subsequence $\{P[t_k]\}$, say to P^*. Since $P[t] \to \mathcal{M}$, it follows that $P^* \in \mathcal{M}$. Using again Lemma 4.5.3, we know that $V_{\text{perim},Q_\epsilon}$ is continuous and $V_{\text{perim},Q_\epsilon}(P^*) = 0$. Hence, $V_{\text{perim},Q_\epsilon}(P[t_k]) \to V_{\text{perim},Q_\epsilon}(P^*) = 0$. This implies that, given any $\delta > 0$, there exists k_0 such that for all $k \geq k_0$, $|V_{\text{perim},Q_\epsilon}(P[t_k]) - V_{\text{perim},Q_\epsilon}(P^*)| = V_{\text{perim},Q_\epsilon}(P[t_k]) < \delta$. From Proposition 4.9.5(iii), we know that $V_{\text{perim},Q_\epsilon}(P[t]) \leq V_{\text{perim},Q_\epsilon}(P[t_{k_0}])$ for all $t > t_{k_0}$. In turn this implies that $V_{\text{perim},Q_\epsilon}(P[t]) \leq V_{\text{perim},Q_\epsilon}(P[t_{k_0}]) < \delta$ for all $t > t_{k_0}$. However, $\|P[t] - P^*\| \leq n \max\{\|p_i[t] - p_i^*\| \mid i \in \{1, \ldots, n\}\} \leq n V_{\text{perim},Q_\epsilon}(P[t])$, where the second inequality holds because

[7] A set $\mathcal{M}$ is weakly positively invariant with respect to a map T on W if, for any $w_0 \in W$, there exists $w \in T(w_0)$ such that $w \in W$.

$p_i^* \in \text{rco}(P[t], Q_\epsilon)$, for $i \in \{1, \ldots, n\}$, and because the length of the shortest measurable curve enclosing a set of points is greater than the distance between any two of the points. Thus, for all $t \geq t_{k_0}$, we have $\|P[t] - P^*\| \leq n\delta$. Hence $P[t] \to P^* \in \mathcal{M}$. ■

4.10 Proofs of results in Section 4.6

Proof of Proposition 4.6.1: We begin by proving statement (i). Note that step `1:` of the `Constraint Set Generator Algorithm` is in fact the sensing operation that takes $O(M)$ time and the result is a star-shaped polygon with at most M vertices. It is easy to see that the intersection of a star-shaped polygon with a half-plane takes time linear in the number of vertices of the polygon. Therefore, step `4:` of the algorithm takes at most M time. If Q has at most κ concavities, then the computational complexity of `Constraint Set Generator Algorithm` is $O(M) + \kappa M$, which is $O(\kappa M)$.

We now prove statement (ii). Step `1:` of the `Perimeter Minimizing Algorithm` is the result of the sensing operation taking $O(M)$ time. Since the sensor takes at most M readings, we can assume that the cardinality of $\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}$ is at most M. In step `2:`, $\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}$ is the set of all robot positions obtained in step `1:`. Let the computation of $\mathcal{N}_{i,\mathcal{G}_{\text{constr}}}$ given the set $\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}$ take $\tau(M)$ time. Here $\tau(M)$ depends on the specific heuristic used to compute the maximal cliques of a graph. In step `3:`, $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}}) = \bigcap_{p_j \in \mathcal{N}_{i,\mathcal{G}_{\text{lc, }\mathcal{G}}}}$ is the intersection of at most M convex constraint sets. It is clear from the proof of statement (i) that each of the convex sets $\mathcal{C}_Q(p_i, p_j)$ has at most M vertices and can be computed in $O(\kappa M)$ time. The intersection of M convex polygons of M vertices takes $O(M^2 \log M)$ time [114]. Thus, the computation of $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}})$ takes $MO(\kappa M) + O(M^2 \log M)$ time and $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}})$ has at most M^2 vertices. The computation of $\text{rco}(\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}, \mathcal{V}(p_i, \epsilon))$ takes $O(M \log M)$ time (cf. [115]), since the cardinality of both $\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}$ and $\text{Ve}(\mathcal{V}(p_i, \epsilon))$ is at most M. Now, finally step `3:` involves the intersection of two star-shaped polygons $C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}_{\text{constr}}})$ and $\text{rco}(\mathcal{N}_{i,\mathcal{G}_{\text{sens}}}, \mathcal{V}(p_i, \epsilon))$ having at most M^2 and M vertices respectively. Therefore the total number of vertices of the two polygons is at most $M^2 + M$. The computational complexity of computing the intersection of the two star-shaped polygons is $O((M^2 + M)\log(M^2 + M) + M^3)$ or $O(M^3)$, where M^3 is due to the fact that the maximum number of possible intersection points is M^3 [116].

Thus the computational complexity of step `3:` is $MO(\kappa M) + O(M^2 \log M) + O(M) + O((M^2 + M)\log(M^2+M) + M^3)$. Since $\kappa \in O(M)$, we have that the computational complexity of step `3:` is $O(M^3)$. In step `4:`, the computation of the circumcenter of X_i, which is a convex polygon of at most M^3 vertices, takes $O(M^3 \log M^3)$ or $O(M^3 \log M)$ time [114]. Step `5:` can be performed in constant time. Thus, the complexity of the entire algorithm is $O(M) + \tau(M) + O(M^3) + O(M^3 \log M)$ or $\tau(M) + O(M^3 \log M)$.

To prove statement (iii), note that if $\mathcal{G} = \mathcal{G}_{\text{constr}} = \mathcal{G}_{\text{sens}}$, then $\tau(M) \in O(1)$. Also, from Lemma 4.9.2 (ii), we have that $X_i = C_{p_i,Q}(\mathcal{N}_{i,\mathcal{G}}) \cap \text{co}(\mathcal{N}_{i,\mathcal{G}})$. The convex hull of M points can be computed in $O(M \log M)$ time (cf. [114]) and hence $\text{co}(\mathcal{N}_{i,\mathcal{G}})$ can be computed in $O(M \log M)$ time. The set X_i is an intersection of convex sets and can be computed in $O(M^2 + M)$ or $O(M^2)$ time, and contains at most $O(M^2)$ vertices. Step `4:` now takes $O(M^2 \log M^2)$ or $O(M^2 \log M)$ time. Thus, the complexity of the entire algorithm is $O(M^2 \log M)$. ■

CHAPTER 5

Conclusion

In this dissertation, we have considered some fundamental problems related to motion coordination of mobile robotic agents with visibility sensors.

The first problem considered is that of deploying a group of such agents over an unknown nonconvex environment to achieve complete visibility. To begin with we consider the problem of optimally locating a single agent in an unknown environment. We look at the area of the environment that is visible to the agent from a point and design an algorithm that drives the agent on a trajectory along which the the area is nondecreasing. The algorithm is gradient-based, i.e., the agent moves along the gradient of the area of the region visible to it. However, the area of the visible region is not differentiable everywhere; this leads to discontinuities in the right hand side of the differential equation governing the motion of the agent. We resort to nonsmooth analysis to characterize the points to which the agent is driven to. We propose a novel form of the LaSalle Invariance Principle and conclude that the agent is driven to a critical point of the area of the visibility region. We simulated the algorithm in `MATLAB`. The results illustrate that, in the presence of noise, the observer reaches a local maximum of the visible area. In a "highly nonconvex" environment, a single observer may not be able to see a large fraction of the environment. In such a case, a team of robotic agents can be deployed to achieve the same task, which is the problem we consider next.

We now revisit the first problem with the objective of designing a distributed deployment algorithm for a group of robotic agents. We assume that all members of the group are initially collocated at a given entry point to the environment. Using a top-down incremental partition algorithm together with bottom-up deployment scheme, we design the `Depth-first Deployment` algorithm which solves the above problem. The number of agents required to guarantee that the task is achieved is $\lfloor n/3 \rfloor$, where n is the number of vertices of the environment. Remarkably,

this number is the identical to that even if the environment were known a priori; see the famous Art Gallery Theorem [76]. A second problem that we consider is the deployment problem but under the additional constraint that the visibility graph of the final configuration of the agents is connected. Using an approach similar to that in the `Depth-first Deployment` algorithm, we propose the `Connected Depth-first Deployment` algorithm that solves the new problem. The number of agents required to guarantee that the task is achieved is $(n-1)/2$. Again, remarkably, this number differs by at most one from the number that is obtained even if the environment is known a priori; see [77].

There are various interesting directions in which the above work can be extended : (a) considering the special case of orthogonal environments which are often used to represent indoor and urban scenarios; see [117] for some preliminary results; (b) handling the case when the initial position of the agents is not collocated; (c) designing distributed algorithms that give better bounds for time taken for the completion of the task; (d) handling changing environments, for example, the sudden opening of a door in an indoor environment; and (e) designing algorithms for 3D environments. Also, the deployment problem is just one of the large and very rich class of "illumination" problems in computational geometry for which distributed sensor-based approaches might be devised. One possible example is the searchlight scheduling problem. A preliminary distributed solution to the problem is provided in [118].

The second problem that we consider is that of designing a distributed algorithm for gathering a group of robotic agents scattered over the environment at a single location. We present a provably correct discrete-time synchronous distributed algorithm, termed as the `Perimeter Minimizing Algorithm` algorithm, which builds on a novel solution to the connectivity maintenance problem also proposed in this work. The name `Perimeter Minimizing Algorithm` derives from the novel Lyapunov function that is chosen to prove the correctness of the algorithm. In other words, the rendezvous objective is achieved if the perimeter of the relative convex hull of the agents' position inside a nonconvex environment is decreased to zero. The performance of the algorithm under asynchronous operation of the robots, presence of noise in sensing and control, and nontrivial robot dimension is investigated and found to be quite satisfactory. The computational complexity of the algorithm under the assumption of finite sensing resolution is also investigated.

To conclude, we have presented distributed algorithms for two fundamental problems pertaining to groups of mobile robotic agents equipped with visibility sensors. It is hoped that this work contributes some measure of progress to the research on motion coordination of mobile sensor networks.

APPENDIX A

Nonsmooth analysis and discontinuous vector fields

In this appendix we review some basic facts and standard notations from nonsmooth analysis [66].

Given a locally Lipschitz function $f : \mathbb{R}^N \to \mathbb{R}$, a point $x \in \mathbb{R}^N$ which verifies that $0 \in \partial f(x)$ is called a *critical point of* f. The extrema of Lipschitz functions are characterized by the following result.

Proposition A.0.1. *Let f be a locally Lipschitz function at $x \in \mathbb{R}^N$. If f attains a local minimum or maximum at x, then $0 \in \partial f(x)$, i.e., x is a critical point.*

Let $\mathrm{Ln} : 2^{\mathbb{R}^N} \to 2^{\mathbb{R}^N}$ be the set-valued map that associates to each closed subset S of $\mathbb{R}^N$ the set of its least-norm elements $\mathrm{Ln}(S)$. For a locally Lipschitz function f, we consider the *generalized gradient vector field* $\mathrm{Ln}(\partial f) : \mathbb{R}^N \to \mathbb{R}^N$ given by $x \mapsto \mathrm{Ln}(\partial f)(x) = \mathrm{Ln}(\partial f(x))$.

Theorem A.0.2. *Let f be a locally Lipschitz function at x. Assume that $0 \notin \partial f(x)$. Then, there exists $T > 0$ such that $f(x - t\,\mathrm{Ln}(\partial f)(x)) \leq f(x) - \frac{t}{2}\|\mathrm{Ln}(\partial f)(x)\|^2$, $0 < t < T$. The vector $-\mathrm{Ln}(\partial f)(x)$ is called a* direction of descent*.*

For differential equations with discontinuous right-hand sides we understand the solutions in terms of differential inclusions following [88]. Let $F : \mathbb{R}^N \to 2^{\mathbb{R}^N}$ be a set-valued map. Consider the differential inclusion

$$\dot{x} \in F(x)\,. \tag{A.1}$$

A solution to this equation on an interval $[t_0, t_1] \subset \mathbb{R}$ is defined as an absolutely continuous function $x : [t_0, t_1] \to \mathbb{R}^N$ such that $\dot{x}(t) \in F(x(t))$ for almost all $t \in [t_0, t_1]$. Given $x_0 \in \mathbb{R}^N$, the existence of at least a solution with initial condition x_0 is guaranteed by the following lemma.

Lemma A.0.3. *Let the map F be upper semicontinuous with nonempty, compact and convex values. Then, given $x_0 \in \mathbb{R}^N$, there exists at least a solution of* (A.1) *with initial condition x_0.*

REFERENCES

[1] Committee on Networked Systems of Embedded Computers. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. National Academy Press, 2001.

[2] Special report on sensor nation. IEEE Spectrum, July 2004.

[3] K. Sugihara and I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems*, 13(3):127–39, 1996.

[4] T. Balch and R. Arkin. Behavior-based formation control for multirobot systems. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[5] P. Tabuada, G. J. Pappas, and P. Lima. Motion feasibility of multi-agent formations. *IEEE Transactions on Robotics*, 21(3), 2005.

[6] J. H. Reif and H. Wang. Social potential fields: a distributed behavioral control for autonomous robots. *Robotics & Autonomous Systems*, 27(3):171–194, 1999.

[7] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951, 2001.

[8] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[9] S. M. LaValle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6):912–925, 1998.

[10] M. Savchenko and E. Frazzoli. On the time complexity of the multiple-vehicle coordination problem. In *American Control Conference*, pages 3536–3541, Portland, OR, June 2005.

[11] A. M. Bruckstein, N. Cohen, and A. Efrat. Ants, crickets, and frogs in cyclic pursuit. Technical Report CIS 9105, Center for Intelligent Systems, Technion, Haifa, Israel, July 1991. Available electronically at `http://www.cs.technion.ac.il/tech-reports`.

[12] J. A. Marshall, M. E. Broucke, and B. A. Francis. Formations of vehicles in cyclic pursuit. *IEEE Transactions on Automatic Control*, 49(11):1963–1974, 2004.

[13] V. Gazi and K. M. Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48(4):692–697, 2003.

[14] C. Tomlin, G. J. Pappas, and S. S. Sastry. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–21, 1998.

[15] O. Jae-Hyuk and E. Feron. Safety certification of air traffic conflict resolution algorithms involving more than two aircraft. In *American Control Conference*, pages 2807–11, Philadelphia, PA, June 1998.

[16] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[17] H. Tanner, A. Jadbabaie, and G. J. Pappas. Stability of flocking motion. Technical report, Department of Computer and Information Science, University of Pennsylvania, January 2003.

[18] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[19] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem - the asynchronous case. In *IEEE Conf. on Decision and Control*, pages 1926–1931, Paradise Island, Bahamas, December 2004.

[20] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[21] J. Cortés, S. Martínez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, 2006.

[22] S. Martínez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

[23] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli. On synchronous robotic networks – Part I: Models, tasks and complexity. *IEEE Transactions on Automatic Control*, 2007. To appear.

[24] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli. On synchronous robotic networks – Part II: Time complexity of rendezvous and deployment algorithms. *IEEE Transactions on Automatic Control*, 2007. To appear.

[25] S. Martínez, J. Cortés, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 2007. To appear.

[26] E. Klavins and R. M. Murray. Distributed algorithms for cooperative control. *IEEE Pervasive Computing*, 3(1):56–65, 2004.

[27] R. Olfati-Saber and R. M. Murray. Consensus protocols for networks of dynamic agents. In *American Control Conference*, pages 951–956, Denver, CO, June 2003.

[28] W. Ren and R. W. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

[29] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

[30] Y. Hatano and M. Mesbahi. Agreement over random networks. In *IEEE Conf. on Decision and Control*, pages 2010–2015, Paradise Island, Bahamas, December 2004.

[31] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[32] M. Mesbahi. On state-dependent dynamic graphs and their controllability properties. *IEEE Transactions on Automatic Control*, 50(3):387–392, 2005.

[33] A. Ganguli, S. Susca, S. Martínez, F. Bullo, and J. Cortés. On collective motion in sensor networks: sample problems and distributed algorithms. In *IEEE Conf. on Decision and Control and European Control Conference*, pages 4239–4244, Seville, Spain, December 2005.

[34] J. Urrutia. Art gallery and illumination problems. In J. R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, Amsterdam, the Netherlands, 2000.

[35] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Oxford, UK, 1987.

[36] T. C. Shermer. Recent results in art galleries. *IEEE Proceedings*, 80(9):1384–1399, 1992.

[37] J. Urrutia. Art Gallery and Illumination Problems. Unpublished Lecture Notes. Available electronically at http://www.matem.unam.mx/˜urrutia, 2004.

[38] A. J. Briggs and B. R. Donald. Robust geometric algorithms for sensor planning. In J.-P. Laumond and M. Overmars, editors, *Workshop on Algorithmic Foundations of Robotics*. A. K. Peters, Wellesley, MA, 1996.

[39] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, Boca Raton, FL, 1997.

[40] H. Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[41] H. H. González-Baños, A. Efrat, J.-C. Latombe, E. Mao, and T. M. Murali. Planning robot motion strategies for efficient model construction. In *International Symposium on Robotics Research*, Snowbird, UT, October 1999.

[42] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998.

[43] J. S. B. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 445–466. CRC Press, Boca Raton, FL, 1997.

[44] V. Boltyanski, H. Martini, and V. Soltan. *Geometric methods and optimization problems*, volume 4 of *Combinatorial optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[45] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. John Wiley, New York, 2 edition, 2000.

[46] Z. Drezner, editor. *Facility Location: A Survey of Applications and Methods*. Springer Series in Operations Research. Springer Verlag, New York, 1995.

[47] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Mateo, CA, 1997.

[48] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.

[49] D. P. Bertsekas and J. N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms - a survey. *Automatica*, 27(1):3–21, 1991.

[50] S. H. Low and D. E. Lapsey. Optimization flow control I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–74, 1999.

[51] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[52] L. E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5(1):5–19, 1999.

[53] R. A. Brooks. A robust layered control-system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[54] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[55] M. J. Matarić. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):323–336, 1997. Special issue on Software Architectures for Physical Agents.

[56] M. S. Fontan and M. J. Matarić. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5):815–822, 1998.

[57] A. C. Schultz and L. E. Parker, editors. *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002. Proceedings from the 2002 NRL Workshop on Multi-Robot Systems.

[58] T. Balch and L. E. Parker, editors. *Robot Teams: From Diversity to Polymorphism*. A K Peters Ltd., Natick, MA, 2002.

[59] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–55, 2002.

[60] R. W. Brockett. Hybrid models for motion control systems. In *Essays in Control: Perspectives in the Theory and its Applications*, pages 29–53. Birkhäuser Verlag, Boston, MA, 1993.

[61] V. Manikonda, P. S. Krishnaprasad, and J. Hendler. Languages, behaviors, hybrid architectures and motion control. In J. Baillieul and J. C. Willems, editors, *Mathematical Control Theory*. Springer Verlag, New York, 1998.

[62] M. Egerstedt. Behavior based robotics using hybrid automata. In *Lecture Notes in Computer Science: Hybrid Systems III: Computation and Control*, pages 103–116, Pittsburgh, PA, March 2000. Springer Verlag.

[63] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[64] U. Helmke and J. B. Moore. *Optimization and Dynamical Systems*. Springer Verlag, New York, 1994.

[65] E. W. Justh and P. S. Krishnaprasad. Pattern-forming systems for control of large arrays of actuators. *Journal of Nonlinear Science*, 11(4):239–277, 2001.

[66] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley, 1983.

[67] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[68] H. H. González-Baños and J.-C. Latombe. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 21(10):829–848, 2002.

[69] K. Kakusho, T. Kitahashi, K. Kondo, and J.-C. Latombe. Continuous purposive sensing for 2D map building. In *Proceedings of the IEEE International Conference of Systems, Man and Cybernetics*, pages 1472–1477, Vancouver, BC, Canada, 1995.

[70] Li-Tien Cheng and Yen-Hsi Tsai. Visibility optimization using variational approaches. *Communications in Mathematical Sciences*, 3(3):425–451, 2005.

[71] A. Ganguli, J. Cortés, and F. Bullo. Maximizing visibility in nonconvex polygons: Nonsmooth analysis and gradient algorithm design. In *American Control Conference*, pages 792–797, Portland, OR, June 2005.

[72] A. Ganguli, J. Cortés, and F. Bullo. Maximizing visibility in nonconvex polygons: Nonsmooth analysis and gradient algorithm design. *SIAM Journal on Control and Optimization*, 45(5):1657–1679, 2006.

[73] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Topological exploration with multiple robots. In *International Symposium on Robotics and Applications*, Anchorage, Alaska, May 1998.

[74] P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, and A. Pelc. Collective tree exploration. In M. Farach-Colton, editor, *LATIN 2004*, volume 2976 of *Lecture Notes in Computer Science*, pages 141–151. Springer Verlag, 2004.

[75] A. Howard, M. J. Matarić, and G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.

[76] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory. Series B*, 18:39–41, 1975.

[77] G. Hernández-Peñalver. Controlling guards. In *Canadian Conference on Computational Geometry*, pages 387–392, Saskatoon, Canada, 1994.

[78] V. Pinciu. A coloring algorithm for finding connected guards in art galleries. In *Discrete Mathematical and Theoretical Computer Science*, volume 2731/2003 of *Lecture Notes in Computer Science*, pages 257–264. Springer Verlag, 2003.

[79] A. Ganguli, J. Cortés, and F. Bullo. Distributed deployment of asynchronous guards in art galleries. In *American Control Conference*, pages 1416–1421, Minneapolis, MN, June 2006.

[80] A. Ganguli, J. Cortés, and F. Bullo. Distributed coverage of nonconvex environments. In V. Saligrama, editor, *Proceedings of the NSF Workshop on Future Directions in Systems Research for Networked Sensing, May 25-26, Boston, MA*, Lecture Notes in Control and Information Sciences. Springer Verlag, 2006. To appear.

[81] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[82] A. Ganguli, J. Cortés, and F. Bullo. On rendezvous for visually-guided agents in a nonconvex polygon. In *IEEE Conf. on Decision and Control and European Control Conference*, pages 5686–5691, Seville, Spain, December 2005.

[83] A. Ganguli, J. Cortés, and F. Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, November 2006. Submitted.

[84] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer Verlag, New York, 2 edition, 2000.

[85] D. S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics. An International Journal*, 45(6):615–627, 1998.

[86] P. Valtr. Guarding galleries where no point sees a small area. *Israel Journal of Mathematics*, 104:1–16, 1998.

[87] O. Cheong, A. Efrat, and S. Har-Peled. On finding a guard that sees most and a shop that sells most. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1091–1100, New Orleans, LA, January 2004.

[88] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides*, volume 18 of *Mathematics and Its Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1988.

[89] D. Shevitz and B. Paden. Lyapunov stability theory of nonsmooth systems. *IEEE Transactions on Automatic Control*, 39(9):1910–1914, 1994.

[90] A. Bacciotti and F. Ceragioli. Stability and stabilization of discontinuous systems and nonsmooth Lyapunov functions. *ESAIM. Control, Optimisation & Calculus of Variations*, 4:361–376, 1999.

[91] P. K. Eason and J. A. Stamps. The effect of visibility on space use by territorial red-capped cardinals. *Behaviour*, 138(1):19–30, 2001.

[92] J. E. Goodman. and R. Pollack. Foundations of a theory of convexity on affine Grassmann manifolds. *Mathematika. A Journal of Pure and Applied Mathematics*, 42(2):305–328, 1995.

[93] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, UK, 2006.

[94] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, November 1990.

[95] J. Cortés and F. Bullo. Coordination and geometric optimization via distributed dynamical systems. *SIAM Journal on Control and Optimization*, 44(5):1543–1574, 2005.

[96] B. Paden and S. S. Sastry. A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators. *IEEE Transactions on Circuits and Systems*, 34(1):73–82, 1987.

[97] R. Simmons, D. Apfelbaum, D. Fox, R. Goldman, K. Haigh, D. Musliner, M. Pelican, and S. Thrun. Coordinated deployment of multiple heterogenous robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 2254–2260, Takamatsu, Japan, 2000.

[98] I.M. Rekleitis and V. Dujmović. Efficient topological exploration. In *IEEE Int. Conf. on Robotics and Automation*, pages 676–681, Detroit, MI, May 1999.

[99] M. Dynia, J. Kutylowski, F. Meyer auf der Heide, and C. Schindelhauer. Smart robot teams exploring sparse trees. In *International Symposium of Matematical Foundations of Computer Science*, Stará Lesná, Slovakia, August 2006.

[100] S. Fisk. A short proof of Chvátal's watchman theorem. *Journal of Combinatorial Theory. Series B*, 24:374, 1978.

[101] Y. Uny Cao, A. S. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[102] T. Arai, E. Pagello, and L. E. Parker. Guest editorial: Advances in multirobot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[103] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[104] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous oblivious robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005.

[105] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem: An extended summary. In V. Kumar, N. E. Leonard, and A. S. Morse, editors, *Proceedings of the 2003 Block Island Workshop on Cooperative Control*, volume 309 of *Lecture Notes in Control and Information Sciences*, pages 257–282. Springer Verlag, New York, 2004.

[106] Z. Lin, M. Broucke, and B. Francis. Local control strategies for groups of mobile autonomous agents. *IEEE Transactions on Automatic Control*, 49(4):622–629, 2004.

[107] R. Orghidan, J. Salvi, and E. Mouaddib. Modelling and accuracy estimation of a new omnidirectional depth computation sensor. *Pattern Recognition Letters*, 27(7):843–853, 2006.

[108] D. Sack and W. Burgard. A comparison of methods for line extraction from range data. In *IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, Lisbon, Portugal, July 2004.

[109] F. Duguet and G. Drettakis. Robust epsilon visibility. In *ACM Annual Conference on Computer graphics and Interactive Techniques (SIGGRAPH '02)*, pages 567–575, San Antonio, Texas, July 2002.

[110] J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517, 1992.

[111] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[112] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization - Supplement Volume A*, pages 1–74. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[113] W. W. Hogan. Point-to-set maps in mathematical programming. *SIAM Review*, 15(3):591–603, 1973.

[114] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer Verlag, New York, 1993.

[115] G. T. Toussaint. Computing geodesic properties inside a simple polygon. *Revue D'Intelligence Artificielle*, 3(2):9–42, 1989.

[116] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 2000.

[117] A. Ganguli, J. Cortés, and F. Bullo. Visibility-based multi-agent deployment in orthogonal environments. In *American Control Conference*, New York, July 2007. To appear.

[118] K. J. Obermeyer, A. Ganguli, and F. Bullo. Asynchronous distributed searchlight scheduling. In *IEEE Conf. on Decision and Control*, New Orleans, LA, December 2007. Submitted.

VITA

Anurag Ganguli was born in 1979 in Jabalpur, Madhya Pradesh, India. He received his preliminary education in Cuttack, Orissa and in Jabalpur, Madhya Pradesh. He received the Bachelor of Technology and the Master of Technology degrees in Mechanical Engineering from the Indian Institute of Technology Bombay, Mumbai in 2002. He received the Master of Science in General Engineering from the University of Illinois at Urbana-Champaign in 2004.

Anurag is a recipient of the Carver Fellowship from the department of General Engineering at the University of Illinois at the Urbana-Champaign, the winner of the Best Student Paper Award at the American Control Conference 2006 and a finalist for the Best Student Paper Award at the American Control Conference 2005. His research has focused on distributed motion coordination algorithms for robotic agents equipped with visibility sensors. After completing his PhD, Anurag will join as a Senior Research and Development Scientist the UtopiaCompression Corporation in Los Angeles, California.