Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Navegação e Controle de Robôs Móveis Cooperativos: Uma Abordagem baseada em Conectividade de Grafos

Guilherme Augusto Silva Pereira

Tese apresentada ao Curso de Pós-Graduação em Ciência Computação da Universidade Federal de Minas Gerais como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Mário Fernando Montenegro Campos

Co-orientador: Prof. Vijay Kumar

Belo Horizonte, Novembro de 2003

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Motion Planning and Control of Cooperating Mobile Robots: A Graph Connectivity Approach

Guilherme Augusto Silva Pereira

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Prof. Mário Fernando Montenegro Campos

Co-Advisor: Prof. Vijay Kumar

Belo Horizonte, November, 2003

To Cinthia,
the love of my life.

Resumo

Esta tese aborda o problema de navegação e controle de grupos de robôs móveis cooperativos. É proposta uma metodologia geral baseada em conectividade de grafos que fornece uma única solução para várias tarefas cooperativas. A metodologia, que se baseia na redução de várias tarefas cooperativas em um problema básico único, possibilita um mesmo time de robôs móveis executar diversas tarefas com um único conjunto de algoritmos parametrizados pelas características de cada tarefa. Assim, mesmo uma tarefa totalmente nova e desconhecida poderia ser realizada se esta pudesse ser transformada em uma tarefa já conhecida pelos robôs. A chave é a transformação de tarefas cooperativas em múltiplos problemas individuais de planejamento de movimento restrito. Para tal, um grupo de robôs é modelado como um grafo, onde cada robô é um vértice e cada aresta representa uma restrição de movimento imposta por outros robôs do grupo e que variam de acordo com a tarefa. Em geral, estas restrições podem ser utilizadas em simples e bem conhecidas técnicas de navegação e controle para um único robô. A técnica explorada neste texto é totalmente descentralizada e seus algoritmos são baseados na modificação em tempo real de funções de navegação especificadas a priori. Para validar a metodologia, são mostrados exemplos práticos em sensoriamento colaborativo, comunicação e manipulação, todos avaliados experimentalmente com grupos de robôs holonomicos e não-holonomicos.

Abstract

This thesis addresses the problem of motion planning and control of groups of autonomous mobile robots during cooperative tasks execution. A general framework that transforms several cooperative tasks into the same basic problem is developed thus providing a feasible solution for all of them. The approach enables using a single team of robots to perform numerous different tasks by providing each robot in the team with a single suite of algorithms which are parameterized by the specificities of the tasks. Therefore, even a totally new and unknown task can be executed by the group of robots if this particular task can be transformed to one of the tasks the team is able to execute. The key is the transformation of cooperative problems to individual constrained motion planning problems. In order to do so, the group of robots is modeled as a graph where each robot is a vertex and each edge represents a motion constraint to be satisfied. Motion constraints are imposed by the task and by the other robots in the group. These constraints may be used in simple and very well known motion planning techniques in order to plan and control the motion of each individual robot. We present a decentralized solution for the problem, which algorithms are based on the online modification of pre-specified navigation functions. Examples in sensing, communication and manipulation tasks are presented, eliciting the elegance of the solutions. Finally, experimental results with groups of both non-holonomic and holonomic robots are presented.

Agradecimentos

Esta tese não teria sido concluída sem a ajuda de pessoas muito importantes e queridas.

Gostaria de agradecer à toda minha familia pelo apoio incondicional durante toda a minha vida. Sem eles eu não teria chegado até aqui.

Sou imensamente grato a minha querida esposa Cinthia pelo companheirismo, amizade e amor. Esta tese certamente não seria possível sem ela.

Agradeço ao Professor Mario Campos pelo apoio, amizade e confiança demonstrados em todos estes anos de convívio. Muitas vezes ele apostava mais em mim do que eu mesmo o faria. Serei eternamente grato pelas diversas oportunidades que ele me proporcionou.

O Professor Vijay Kumar foi de extrema importância, não só para a elaboração deste trabalho, mas principalmente para o meu amadurecimento como pesquisador. A nossa curta convivência certamente influenciará toda a minha vida científica. Agradeço-o imensamente pela paciência ao ler minhas muito mal traçadas linhas e ouvir minhas não tão claras idéias.

Gostaria também de agradecer aos demais membros da minha banca, professores Luis Antônio Aguirre, Renato Cardoso Mesquita, Teodiano Bastos Filho e José Reginaldo Hughes Carvalho, pelas opiniões e sugestões de melhoria do trabalho. Gostaria ainda de ressaltar a participação dos professores Luis Aguirre e Renato Mesquita na minha formação profissional, já que como professores de graduação e pós-graduação, não só me forneceram

embasamento teórico e prático, como também me inspiraram a ingressar na carreira científica e acadêmica.

Agradeço aos colegas Aveek Das, John Spletzer, Dan Gomez-Ibanez, e Dan Walker pela ajuda com os robôs, software e experimentos no GRASP Lab. Além destes gostaria de lembrar o amigo Rahul Rao, por tornar a vida longe de casa menos difícil.

Meu muito obrigado a todos os colegas do VERLab pelo ótimo ambiente de trabalho, em especial Bruno, Samuel, Denilson, Andréa, Pedro, Flávio, João e Pio. Um agradecimento muito especial aos colegas Fernanda e Sancho, pela ajuda com os robôs holonômicos, Pinheiro, pela grande ajuda com as "burocracias acadêmicas", e Wilton, que foi o grande responsável pela "telepresença" do Prof. Kumar durante a defesa da tese.

Um agradecimento especial a Luciana e Chaimo pela grande ajuda na transição BH/Philly. Sua amizade foi ainda fundamental para amenizar a saudade de casa.

Merecem ainda meus agradecimentos todos os funcionários do DCC e do GRASP, em especial Renata e Emilia, que sempre resolveram com prontidão toda a burocracia associada ao processo de doutoramento, e Dawn Kelly e Genny Prechtel, pela ajuda burocrática na UPENN.

Agradeço ao CNPq pelo suporte financeiro.

Finalmente, agradeço a todos aqueles com quem tive a oportunidade de conviver e trabalhar e que participaram direta ou indiretamente do processo de conclusão deste trabalho. Muito obrigado!

Acknowledgments

This thesis would not be concluded without the support of very important and special people.

I would like to thank my family for the unconditional support during my whole life. Without them I would never get this far.

I'm very grateful to my beloved wife Cinthia for her support, friendship and love. This thesis would not be possible without her.

Thanks to Professor Mario Campos for his support, friendship and trustiness in all these years we have been working together. Many times he had more faith in me than I did myself. I will be eternally grateful for the various opportunities he has presented me.

Professor Vijay Kumar was extremely important, not only for the conclusion of this thesis, but also for my development as a researcher. The short time we have worked together will certainly influence my whole scientific life. I thank him very much for his patience in reading my poorly written words and listening to my not so clear ideas.

I would like to thank the other members of my thesis committee, professors Luis Antônio Aguirre, Renato Cardoso Mesquita, Teodiano Bastos Filho and José Reginaldo Hughes Carvalho, for their opinions and suggestions to improve this work. Also, I would like to mention the influence professors Luis Aguirre and Renato Mesquita had in my professional life. As undergraduate and graduate professors they not only gave me theoretical and practical

background, but also inspired me to choose the scientific and academic career.

I thank Aveek Das, John Spletzer, Dan Gomez-Ibanez, and Dan Walker for helping with robots, software and experiments at the GRASP Lab. Also, I would like to thank my officemate and friend Rahul Rao for making life far from home less difficult.

Many thanks to my fellow students at VERLab for creating an enjoyable working environment and in special to Bruno, Samuel, Denilson, Andréa, Pedro, Flávio, João and Pio. Special thanks to Fernanda and Sancho, for helping with the holonomic robots, to Pinheiro, for helping with the "academic bureaucracy", and to Wilton, the main responsible for the "virtual presence" of Prof. Kumar during the thesis defense.

Special thanks to Luciana and Chaimo for the enormous help in the transition BH/Philly. Their friendship was also essential to ease our homesick.

Also deserve my gratitude, the staff of DCC and GRASP, in special Renata and Emilia, for solving promptly all bureaucracy associated with the doctorate program, and Dawn Kelly and Genny Prechtel, for helping with the paperwork in GRASP.

I thank CNPq for the financial support.

Finally, I thank all people with whom I've had the opportunity to work with, and who participated directly or indirectly in this work. Thank you very much!

Contents

List of Figures

List of Tables

List of Symbols

$\mathcal{C}$	The configuration space, page 25
$\mathcal{C}_{obs\_i}$	Representation of an obstacle in the configuration space of the i^{th} robot, page 26
δ	Parameter that determines the activation of a constraint, page 47
γ_{ij}	Configuration space for R_i where the constraints induced by R_j are satisfied, page 45
Γ_i	Configuration space where R_i satisfies its formation constraints, page 46
$\mathcal{A}_i$	Description of the i^{th} robot in the workspace, page 22
$\mathcal{B}_j$	Description of the j^{th} obstacle in the workspace, page 22
$\mathcal{C}_{cls}$	Closure configuration space, page 85
$\mathcal{C}_{obj\_i}$	Intersection between the i^{th} robot and a movable object in the object's configuration space, page 84
$\mathcal{C}_{obj}$	Collision region for the object in the object's configuration space, page 85
$\mathcal{C}_{R_i}$	Valid configuration space for the i^{th} robot, page 42
$\mathcal{E}$	Edge set of a graph, page 37
$\mathcal{F}$	Free configuration space, page 26
$\mathcal{G}_{ij}$	Set of constraints between R_i and R_j, page 44
$\mathcal{I}_i$	Intersection between $\mathcal{C}_{obj\_i}$ and $\mathcal{C}_{obj\_k}$, relative to R_i and R_k, in R_k's configuration space, page 87

$\mathcal{O}_k$	Description of the k^{th} object in the workspace, page 22
$\mathcal{R}$	Set of mobile robots, page 21
$\mathcal{V}$	Vertex set of a graph, page 37
$\mathcal{W}$	Robots' workspace, page 21
∇g^x	Unit vector along the gradient of constraint g_x, page 51
$\nabla\Phi(q)$	Gradient of the potential function, page 30
$\nabla\phi_i$	Normalized gradient of the navigation function, page 51
$\ominus$	Minkowski subtraction operator, page 27
$\oplus$	Minkowski sum operator, page 27
$\Phi(q)$	Potential function, page 30
$\{A\}$	Reference frame fixed in $\mathcal{A}$, page 23
${}^A\mathbf{R}_B$	Rotation matrix that transforms the components of vectors in $\{A\}$ into components in $\{B\}$, page 23
${}^A\mathbf{T}_B$	Rotation matrix that transforms the components of vectors in $\{A\}$ into components in $\{B\}$, page 25
e_{ij}	An edge between vertices i an j of a graph, page 37
G	A graph, page 37
$g(\cdot)$	A motion constraint, page 33
G_c	Constraint graph, page 105
G_f	Formation control graph, page 105
g_k^x	The k^{th} constraint induced by R_x in R_i, page 51
G_{cn}	Communication network graph, page 63
G_{sn}	Sensing network graph, page 63
H	Formation graph, page 109
HC	Hamiltonian Cycle, page 38

HG	Hamiltonian Graph, page 38
HP	Hamiltonian Path, page 38
q	A configuration, page 25
q^0	Initial configuration, page 29
q^d	Desired configuration, page 29
R_i	The i^{th} mobile robot, page 21
$SE(n)$	The special Euclidean group in n dimensions, page 24
$SO(n)$	The special subgroup of rotations in n dimensions, page 24
$T(n)$	The special subgroup of translations in n dimensions, page 24
v_i	The i^{th} vertex of a graph, page 37

Chapter 1

Introduction

> If every instrument could accomplish its own work, obeying or anticipating the will of others...if shuttle could weave, and the pick touch the lyre, without a hand to guide them, chief workmen would not need servants nor master slaves.
>
> *Aristotle (384 BC–322 BC)*

The continuous advances in technology have made possible the use of several robotic agents in order to carry out a large variety of cooperative tasks. While one could design a robot for every imaginable task, it would be intuitively more efficient, and perhaps more effective, to assign a team of cooperative mobile robots to perform different tasks, with the possibility that some of the tasks being executed concurrently. Following this idea, we are interested in effectively using groups of existing mobile robots in order to execute various distinct cooperative tasks without being modified to perform them. In other words, we are not interested in engineering the problem by changing the robots' hardware to satisfy the requirements of the task, but engineering the solution by developing robust and reliable software that take into account both the constraints of the robots and of the task. Thus, our problem can be defined as:

Given a team $\mathcal{R}$ of mobile robots and a set $\mathcal{T}$ of tasks to be performed, generate for each $\tau \in \mathcal{T}$ a solution defined by the tuple $< r, a_i >$ where $r \subset \mathcal{R}$

is the subgroup of robots to be used during the task and a_i is a specification of each robot action.

Specifically we want to consider the execution of all tasks as a multi-robot motion planning problem and develop decentralized algorithms for controlling the robots.

1.1 Motivation

The main motivation behind this work is the large number of tasks which are dangerous, inaccessible or even impossible to be effectively carried out by a group of human beings. It might be very interesting, for example, to equip fire-brigades with teams of mobile robots that could be used in situations such as rescuing human beings trapped under piles of debris, inaccessible to human fire fighters. The same team of robots could be also used to provide a mobile communication infrastructure in an earthquake disaster or be commanded to search for victims of a flood. In rare accidents such as the disintegration of the NASA Columbia Space Shuttle during its reentry in Earth atmosphere in January 2003, robots could be used not only for finding, but also for retrieving the debris scattered over a very broad area. Had such a robotic system been available on September 11th, 2001, many fire-fighters could probably have their lives spared during the rescue operation in the twin towers of the World Trade Center. Unfortunately those men did not abandon the buildings because, due to difficulties of communication in the area, they had never received such a command from their superiors. In that situation a network of robots would be useful for providing the necessary communication infrastructure by forming a connected ad hoc network where the fire-fighters and the command base would be communication nodes. If

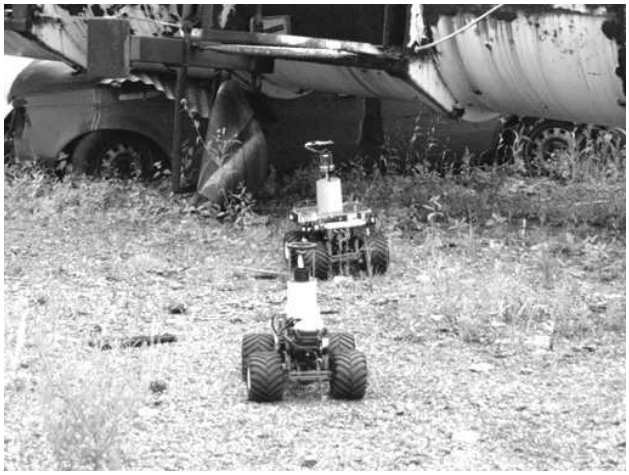

Figure 1.1: Two robots in a disaster situation.

large teams of small and agile robots were part of the search operation, it could be the case that more survivors would have been found. In a foreseeable future, a group of robots could establish a mobile sensor network in inhabited mountainous areas and continuously monitor the risk of avalanche. Several deaths could have been avoided in Belo Horizonte in the Summer of 2002/2003, had such a technology existed. Figure 1.1 shows a picture of our first effort in the direction of enabling such a technology. In this picture two camera-equipped teleoperated robots are sending real time images of a car accident to a remotely located operator.

Less stringent applications for versatile teams of robots can be found in domestic, industrial, and war environments. A common characteristic of these applications is that the team of robots is always near to the human user, interacting with the user, and augmenting his/her skills, providing a natural synergism. In order to have this important requirement, the robots need to have high degree of autonomy so as to execute tasks without close supervision of the operator, who may be executing a different task. The ability to use multiple robots allows the increasing of system autonomy without increasing

the complexity of individual robot behaviors but instead by exploiting the distributed execution of these behaviors.

1.2 Approach

Examples in the previous section are typical instances of cooperation among humans and robots or only among robots that can be classified as *tightly coupled cooperation*. This nomenclature was first introduced by [Brown and Jennings, 1995] in order to describe the forms of cooperation used when the task cannot be serialized. In general, this kind of cooperation is necessary when cooperating entities relay on their teammates's sensors, processing and communication capabilities, actuators and even physical presence. In contrast to *loosely coupled cooperation*, where the task could be divided in parts and executed independently by each agent, in tightly coupled cooperation, tasks can only be completed with the interaction of a minimum number of agents. A typical example of this kind of cooperation is pictured in Figure 1.2. We consider situations where large or clumsy objects need to be transported, such that they could not be carried by a single robot alone. Hence, with the help of at least another robot the task could be completed.

Another example, where the task do not demand physical contact, is when multiple robots must maintain communication with a human operator while executing their tasks. Communication is normally accomplished using wireless ad hoc networks, having as nodes each robot and the human operator. If in order to accomplish the task it is required that some of the robots move outside the operator's communication range, one may consider two solutions: (i) the operator move closer to the robots or, more interesting, (ii) "relay" robots position themselves so as to guarantee uninterrupted communication

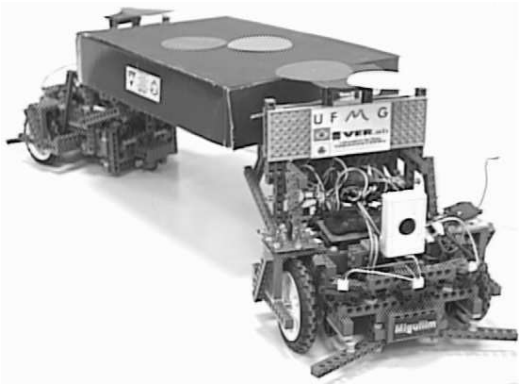

Figure 1.2: Two robots in tightly coupled cooperation carrying a relative large box (about double the length of each robot) [Pereira et al., 2002b].

between the operator and the moving robots. Since communication depends on the relay robots and a failure of one of these robots could compromise the task execution, it can be seen that the cooperation (in this case involving a human operator) is of tightly coupled type.

Our approach for specifying the motion of a group of cooperating mobile robots is based on the fact that the dependence among the agents, created by the tightly coupled nature of the tasks, introduces constraints to their motions.

Consider, for instance, the box transportation task in Figure 1.2. Each robot must move while satisfying some constraints, which are imposed by the box and by the motion of the other robot. If the leading robot moves much faster than the following one, for example, the box will eventually fall. Thus, it is clear that the velocity of the leading robot is constrained by the follower's velocity. A similar constraint can also be derived to the follower robot. Hence, in this thesis we provide a framework that solves tightly coupled cooperation problems when they can be reduced to constrained motion planning and control problems. The only difference among the problems would be the type and the nature of the constraints. Once a cooperative task

is solved, it is straightforward to adopt the solution to another cooperative task. This idea will be explored in depth in the rest of this text.

1.3 Contributions

This work has ushered in four major contributions to the area of cooperative robotics:

1. A generalized framework for solving the problem of planning and controlling the motion of cooperating mobile robots. The framework considers all kinds of interactions among robots and the requirements of the task as individual temporal constraints. The framework consists of decentralized algorithms that rely only on each robot's ability to estimate the positions of its neighbors. Therefore, our methodology is potentially scalable to larger groups of robots operating in unstructured environments. Conditions are derived under which all requirements of the task are satisfied (Chapter 4);

2. A decentralized approach for multi-robot manipulation. The methodology requires the less stringent condition that the object be trapped or caged by the robots. Our algorithms do not rely on exact estimates of either position or orientation of the manipulated object and therefore are potentially more robust than previous approaches relying on force closure (Chapter 6);

3. A decentralized leader-following framework where, besides following their leaders, each robot must maintain proximity constraints with the other robots in the formation they are maintaining. With this framework it is possible to introduce useful behaviors such as the one where

a leader waits for its followers (Chapter 7);

4. A graph based cooperative sensing algorithm for multi-robot localization and tracking. Both global and relative localization are captured by our algorithm. Decentralization of the algorithm is discussed (Apendix B).

All of the above contributions resulted from theoretical development and were experimentally evaluated using two distinct groups of mobile robots (Apendix A).

1.4 Organization

This document is organized in eight chapters (including this one) and two appendices.

Chapter 2 reviews the relevant works in cooperative robotics and gives a general background of the area. We focus on the works related to motion planning, coordination and control of multiple cooperative robots, which are the main subjects of the thesis. Other references related to specific chapters of the document are surveyed in the beginning of each chapter.

Chapter 3 is a review of some basic tools used in this thesis.

Chapter 4 presents the main contribution of the thesis. It contains a generic in-depth discussion of the methodology for multi-robot motion planning and control.

Chapter 5 shows an example of the framework presented in Chapter 4 in applications where each robot has to move while maintaining either communication or sensing visibility with its teammates. Experimental results are presented.

A second example of the motion control framework is presented in Chapter 6. We model an object transportation task as set of motion constraints for a group of robots and present experimental results with groups of three and four robots.

As a third example of the methodology, we present in Chapter 7 a leader-following approach where, besides following a robot leader, each robot is supposed to maintain proximity and avoidance constraints with other robots in the group.

Concluding remarks and suggestions of continuity are discussed in Chapter 8.

Apendix A describes relevant details of the multi-robot platform used to validate the theory presented in the previous chapters.

Apendix B presents our methodology for localization and object tracking. This methodology was implemented on one of the platforms presented in Apendix A and used in the experiments presented in this thesis.

Chapter 2

Related Work

> Seeing that I cannot choose a particulary useful or pleasant subject, since the men born before me have taken for themselves all the useful and necessary themes, I shall do the same as the poor man who arrives last at the fair, and being unable to choose what he wants, has to be content with what others have already seen and rejected because of its small worth. I will load my humble bags with this scorned and disdained merchandise, rejected by many buyers, and go to distribute it not in the great cities, but in poor villages, receiving the price for what I have to offer.
>
> *Leonardo DaVinci (1452–1519)*

This chapter presents a survey of the current state of the art in the field of distributed and cooperative robotics and, more specifically, in the area of planning and control of multiple mobile robots. The chapter is divided in four sections. Section 2.1 presents a general overview of the area of multi-robot motion coordination and locates this work among the works found in literature. The same procedure is adopted in Section 2.2 where we focus on cooperative manipulation. Sections 2.3 and 2.4 present some works related to communication and sensing in teams of mobile robots, respectively. These topics, together with manipulation, are applications of our multi-robot control approach. Related works necessary to support more specific topics of the thesis will be surveyed wherever necessary.

2.1 Multi-Robot motion coordination

Since late 1980's, when several researchers began to investigate some fundamental issues in multi-robot systems, one of the topics of particular interest was multi-robot motion coordination. After the famous seminal papers on reconfigurable robots [Fukuda and Nakagawa, 1987, Beni, 1988] were published, papers on motion control [Arai et al., 1989, Wang, 1989] and coordination [Asama et al., 1989] followed up. More recent research in this field include applications such as automated highway systems [Varaiya, 1993], formation flight control [Mesbahi and Hadaegh, 2001], unmanned underwater vehicles [Smith et al., 2001], satellite clustering [McInnes, 1995], exploration [Burgard et al., 2000], surveillance [Rybski et al., 2000], search and rescue [Jennings et al., 1997], mapping [Thrun, 2001] and manipulation [Matarić et al., 1995].

There are several approaches to multi-robot motion coordination and control reported in the literature. Most of these approaches may be classified as deliberative or reactive, and centralized or decentralized. However, several others exist, combining the main characteristics of the previous ones.

Some deliberative approaches for motion planning consider a group of n robots as a single system with n degrees of freedom and use centralized planners to compute paths in the combined configuration space [Aronov et al., 1998]. In general, these techniques guarantee completeness but with exponential complexity in the dimension of the composite configuration space [Hopcroft et al., 1984]. Other groups have pursued decentralized approaches to path planning. This generally involves two steps: (i) individual paths are planned independently for each robot; and (ii) the paths are merged or combined in a way collisions are avoided. Some authors call these approaches *coordinated path planning* [LaValle and Hutchinson, 1998,

Simeon et al., 2002, Guo and Parker, 2002].

Totally decentralized approaches are in general behavior-based [Balch and Arkin, 1998]. In *behavior-based* control [Arkin, 1998], several desired behaviors are prescribed for each agent and the final control is derived by weighting the relative importance of each behavior. Typically, each agent has a main behavior that guides it to the goal and secondary behaviors that are used in order to avoid obstacles and other robots in the team. These behaviors are generally based on artificial potential fields [Khatib, 1986], such as in [Howard et al., 2002], where this technique was used to deploy robots in unknown environments. Artificial potential fields as a model for robots repelling each other were also used in [Reif and Wang, 1995]. The main problem with these approaches is the lack of formal proofs and guarantees of completion and stability.

A totally decentralized methodology for coordination of large scale agent systems is proposed in [Vicsek et al., 1995]. This methodology is basically a behavior-based approach but with guaranteed convergence for the group behavior. Each agent's heading is updated using a local rule based on the average value of its own heading and the headings of its neighbors. The authors call this the *nearest neighbor rule*. A similar model was developed in [Reynolds, 1987] for computer graphics simulation of flocking and schooling behaviors for the animation industry. In both works, it has been shown by simulation that the nearest neighbor rule may cause all agents to eventually move in the same direction in spite of the absence of centralized coordination. A theoretical explanation is given in [Jadbabaie et al., 2002].

Some research groups consider a set of robots in formation as a rigid structure, that is called *virtual structure* [Eren et al., 2002, Olfati-Saber and Murray, 2002b]. In [Olfati-Saber and Murray, 2002a], the

authors propose potential functions to initially stabilize a group of robots in a rigid formation. A second step is to assign a desired motion to the virtual structure, which is decomposed into trajectories that each member of the formation will follow. Virtual structures have been also proposed in [Tan and Lewis, 1997] and used for motion planning, coordination and control of space-crafts [Gentili and Martinelli, 2000]. Virtual structures are, in general, centralized and deliberative methods but some approaches use reactive *leader-following* in order to maintain formation. In such a framework, each robot has a designated leader, which may be other robots in the group or a virtual robot that represents a pre-computed trajectory supplied by a higher level planner. Thus, each robot is a follower that tries to maintain a specified relative configuration (a fixed separation and bearing, for example) with respect to its leader(s) [Desai et al., 1998, Leonard and Fiorelli, 2001, Das et al., 2002a, Pereira et al., 2003b].

The approach we propose is inspired in some of the ideas mentioned earlier. We propose totally decentralized, reactive controllers for each robot based only on the relative position of its nearest neighbors. However, we require centralized planners to define the relationship between a robot and its neighbors and also deliberative (but decentralized) motion planners in order to take into account static, known obstacles in the environment. We also consider formation of robots, but differently from rigid structures, we are mostly interested in flexible formations, where the robots are not required to maintain fixed relative positions but bounds on these positions. Furthermore, our methodology is based on well known theories, which enables proofs and guaranteed results for controller design.

2.2 Cooperative Manipulation

Transport and object manipulation with mobile robots have been extensively discussed in the literature. Most approaches use the notions of force and form closure to perform the manipulation of relative large objects [Ota et al., 1995, Kosuge and Oosumi, 1996, Rus, 1997, Sugar and Kumar, 1998, Chaimowicz et al., 2001b]. *Force closure* is a condition that implies that the grasp can resist any external force applied to the object. *Form closure* can be viewed as the condition guaranteeing force closure, without requiring the contacts to be frictional. In general, robots are the agents that induce contacts with the object, and are the only source of grasping forces. But, when external forces acting on the object, such as gravity and friction, are used together with the contact forces to produce force closure, we have a situation of *conditional force closure*. Several research groups have used conditional closure to transport an object by pushing it from an initial position to a goal [Rus et al., 1995, Matarić et al., 1995, Lynch, 1996].

In this thesis we use the concept of *object closure*. In contrast to approaches derived from force or form closure, as shown in Figure 2.1, object closure requires the less stringent condition that the object is trapped or caged by the robots. Observe in Figure 2.1(d) that the robots surround the object but not necessarily touch it or press it. In other words, although it may have some freedom to move, it cannot be completely removed [Davidson and Blake, 1998, Wang and Kumar, 2002b]. Because a caging operation requires a relatively low degree of precision in relative positions and orientations, manipulation strategies based on caging operations are potentially more robust than, for example, approaches relying on force closure [Sugar and Kumar, 1998].

Caging was first introduced in [Rimon and Blake, 1996] for non-convex

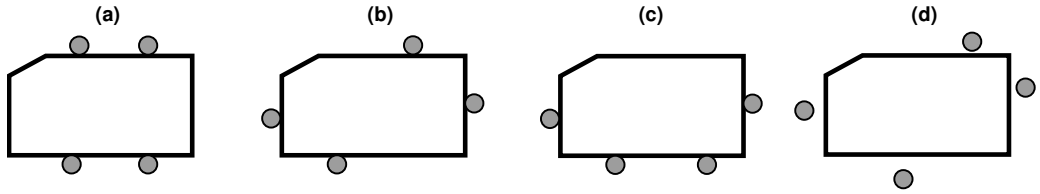

Figure 2.1: Four technics of manipulation: (a) force closure (robots pressing the object); (b) form closure; (c) conditional closure (robots pushing the object up); (d) object closure or caging.

objects and two fingered gripers (Our use of the concept of caging is slightly different from the definition in [Rimon and Blake, 1996], and we call it object closure.) Other papers addressing variation on this basic theme are [Davidson and Blake, 1998, Sudsang and Ponce, 1998, Sudsang et al., 1999, Sudsang and Ponce, 2000, Wang and Kumar, 2002b, Wang and Kumar, 2002a]. Of all these, our approach (presented in Chapter 6) is closest in spirit to the work [Sudsang and Ponce, 2000]. However, while they have developed a centralized algorithm for moving three robots with circular geometry in an object manipulation task, we focus on decentralized algorithms to verify object closure and coordinated a group of n generic-shaped robots [Pereira et al., 2003d].

2.3 Communication in Robot Teams

Cooperating mobile robots must be able to interact with each other using either *explicit* or *implicit* communication and frequently both. Explicit communication corresponds to a deliberate exchange of messages, typically through a wireless network designed solely to convey information to other robots on the team [Parker, 2000]. Examples of cooperative tasks using this type of communication can be found in several papers [Rus et al., 1995, Chaimowicz et al., 2001b, Sugar and Kumar, 1998] as a method for coordi-

nating and controlling teams of mobile robots. On the other hand, implicit communication is derived through sensor observations that enable each robot to estimate the state and trajectories of its teammates. For example, each robot can observe relative state (position and orientation) of its neighbors (implicit communication), and through explicit communication exchange this information with the whole team in order to construct a complete configuration of the team. This may be very useful for multi-robot sensing, as presented in the next section.

When communication uses explicit exchange of messages, the robots, in general, establish among themselves a *Mobile Ad hoc Network* (MANet) [Manet, 2003]. This network is defined as an autonomous system of mobile nodes connected by wireless links. An ad hoc topology is in general represented by a graph where each node is a robot and the edges are communication links established between any two robots. Therefore, in MANets the nodes are free to move, which changes the network topology either by creating or breaking links (edges). Although, ad hoc networks have been the center of attention of various researches, the focus has been on routing algorithms to enable efficient, low-cost computation keeping network connectivity [Kumar et al., 2000, Toh et al., 2002]. Works on mantaining connectivity in MANets by moving the nodes are scarce in the literature. One of the first works in this field is [Pimentel, 2002], which presents algorithms for maintaining communication in a mobile ad hoc network where each node is a robot performing a search and rescue task.

Implicit communication offers several immediate advantages over the explicit form, which include simplicity, robustness to faulty and noisy communication environments, lower power consumption, and stealthiness. In [Balch and Arkin, 1994], the authors show that although communication

significantly improves the performance of a robotic team, explicit communication is not essential when the implicit form is available. Still, more complex communication strategies offer little or no benefit over low-level communication. The paper [Khatib et al., 1995] shows a cooperative manipulation task in which inter-robot communication is achieved through the interaction forces sensed by each manipulator. In our previous work [Pereira et al., 2002b] we have shown that almost no performance increase is obtained when implicity communication is replaced by the explicit form in an object carrying task. As in [Khatib et al., 1995], communication is performed through the object being manipulated. Another example of implicit communication is given in [Stiwell and Bishop, 2001], which presents a theoretic approach showing that the amount of explicit communication can be reduced by using the implicit form. Their approach is validated in a group of simulated underwater vehicles which maintain tight formation by inferring distances from each other through acoustic vibrations produced by their thrusters.

It is important to notice that, in most cases, both implicit and explicit communication require each robot to respect bounds in their positions in order to maintain communication. One of the objectives of this thesis is providing planning and control strategies that enable the robots to move while preserving communication. In order to do so, we model communication range as a motion constraint for each individual robot and create a group behavior which guarantees inter-robot communication [Pereira et al., 2003a]. For explicit communication we assume an ad hoc wireless network and rely on maintaining graph connectivity in order to guarantee communication. We are also concerned with the amount of information interchanged by the robots and thus present totally decentralized controllers. We adopt similar criteria for implicit communication by sensing, which is discussed next.

2.4 Multi-Robot Sensing

A robotic sensing system may be as simple as a single sensor or a highly complex configuration composed by multiple sensors. In the last scenario, data are processed and combined to provide information that is more reliable and complete when compared to the single sensor approach. We are interested in situations where sensors are placed on networked mobile robots programmed to perform a large variety of cooperative tasks such as search and rescue, surveillance and manipulation. Therefore, we focus in robot localization and tracking as the fundamental tasks to be achieved. Actually, it is possible to consider that targets to be tracked are passive team members that do not contribute with sensor measurements in the localization process, but that must be still localized.

Sensor fusion and robot localization yield significant improvements to methods used in single and multiple mobile robot navigation, localization and mapping [Majumder et al., 2001, Thrun, 2001, Thrun et al., 2000]. Most works in this field have emphasized probabilistic techniques for data fusion such as Kalman [Balakrishnan, 1987] and Information Filters [Durrant-Whyte and Manyika, 1993], with a recent focus on particle filtering methods [Arulampalam et al., 2002]. In [Fox et al., 2000] two robots are localized via Monte Carlo Localization. Each robot maintains a particle set approximating the probability distribution of its pose with respect to a global reference frame. These estimates are then refined whenever one robot detects another so that each robots' beliefs (sensor measurements and particle sets) are made consistent. As consequence, the uncertainty in estimation of each robot is reduced.

In [Roumeliotis and Bekey, 2000], the authors present a Kalman filter based approach for addressing interdependencies in uncertainty propagation

for multi-robot localization. Each robot computes its own Extended Kalman Filter (EKF) for state estimation. There is a main advantage in this approach. Although the uncertainty representation is centralized — each robot maintains state estimates for the entire formation — the process of updating the covariance matrix is computationally distributed. This is accomplished by dividing the updates into two phases: intra-robot updates based on local sensors, and inter-robot updates resulting from robot detection and communication. More recently, [Howard et al., 2003] presents a Bayesian approach to cooperative localization. As in [Fox et al., 2000], each robot maintains an estimate for the relative position of each of its teammates as a particle set. However unlike [Fox et al., 2000], attempts were made to address interdependencies in uncertainty estimates.

The aforementioned methods rely on system dynamics in order to maintain uncertainty estimates for the system states estimates. In contrast, the *direct* or reactive approaches for sensing relies on the high level of accuracy of unaltered sensor measurements to estimate parameters of interest at each time step. Estimates for uncertainty in pose are not maintained.

The first application of cooperative localization in robotics employed a direct approach [Kurazume and Hirose, 1998]. The authors addressed odometry problems by employing a cooperative strategy using teams of three or more robots. In their model, the robots themselves served as moving landmarks, and the team conducted itself through the environment using a leapfrog approach. This approach relied upon a robust sensor suite for accomplishing the localization task. A very similar idea was later presented in [Spletzer et al., 2001]. Another direct approach is presented in the work [Stroupe et al., 2001]. The authors use a static Kalman Filter for tracking a target using information from multiple robots. Similarly, the authors of

[Das et al., 2002a] performed cooperative localization using a least squares method in order to fuse and optimize the robot's position and orientation.

The localization approach we present in Apendix B is direct since no estimates are used from one step to another. However, in order to improve object tracking, our approach has the possibility of including EKFs [Pereira et al., 2003f].

An important observation is that the larger the number of inter-robot information, the more precise are the localization estimates. With this in mind, in this thesis we model the presence of measurements as constraints for robots motion and perform multi-robot planning and control under such constraints in order to guarantee a minimum number of required measurements, and move toward increasing this number.

Chapter 3

Background

> If I have seen further, it is by standing on the shoulders of giants.
>
> *Isaac Newton (1642–1727)*

This chapter presents a short review of some basic tools used in this thesis. It is also intended to introduce and explain notations used in the rest of the text, thus facilitating its understanding.

3.1 World Modeling

Consider a workspace, $\mathcal{W}$, which can be either two-dimensional ($2D$) or three-dimensional ($3D$). Mathematically, $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$ respectively. The entities of the world are a set, $\mathcal{R} = \{R_1, \ldots, R_n\}$, of n robots, which behave according to a motion strategy, objects and obstacles. Objects are bodies that can be manipulated (pushed, pulled or carried) by the robots such as boxes, carts, etc. There are two kinds of obstacles:

- Static obstacles: portions of the world permanently occupied, such as walls[1], stairs, bookshelves, etc.

[1] Here we consider walls inside the workspace. Walls that bound $\mathcal{W}$ are not considered obstacles, but limits to the robots' and objects' positions.

- Dynamic obstacles: portions of the world that move in an uncontrolled way, for example, people, other robots, etc.

We use a boundary representation in order to geometrically model the entities of the world. We first assume that entities are (or can be approximated by) rigid polygonal or polyhedral subsets of $\mathcal{W}$. Also, for the sake of simplicity, assume that entities are at first convex and remember that non-convex entities can be easily represented by a union of convex ones. Thus, the i^{th} robot R_i is described by the convex and compact (i.e closed and bounded) subset $\mathcal{A}_i$ of $\mathcal{W}$. In addition, the obstacles $\mathcal{B}_1, \ldots, \mathcal{B}_k$ and the movable objects $\mathcal{O}_1, \ldots, \mathcal{O}_l$ are also compact subsets of $\mathcal{W}$.

In a $2D$ world, convex entities are represented by an intersection of m half planes derived from the line equations of each edge. Then, let the vertices of such entities be given in counterclockwise order and, $f_j(x, y) = a_j x + b_j y + c_j$, where a, b and c are real numbers, be the line equation that corresponds to the edge from (x_j, y_j) to (x_{j+1}, y_{j+1})[1] and $f_j(x, y) < 0$ for all points to the left of the edge (Figure 3.1(a)). The convex, polygonal entity, $\mathcal{P}$, can be expressed as:

$$\mathcal{P} = H_1 \cap H_2 \cap \cdots \cap H_m \ , \tag{3.1}$$

where H_j is the half plane defined as:

$$H_j = \{(x, y) \in \mathbb{R}^2 \mid f_j(x, y) \leq 0\}, \qquad 1 \leq j \leq m \ . \tag{3.2}$$

An example of such a representation for a triangular entity is shown in Figure 3.1.

[1] From now on, consider we are using a circular notation or, in other words, $j + 1 = 1$ for $j = m$ and $j - 1 = m$ for $j = 1$.

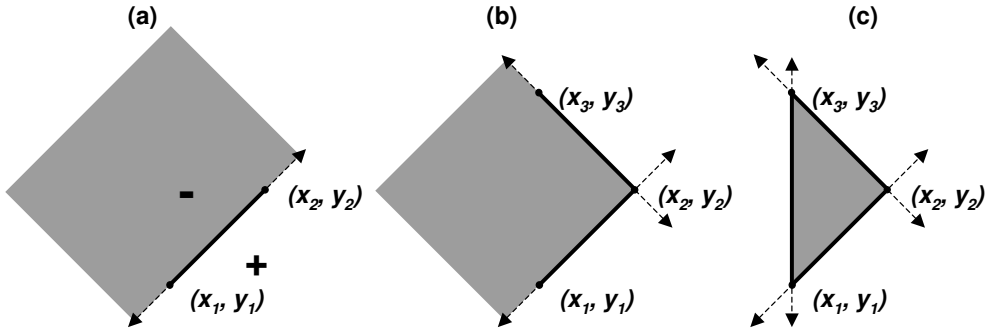

Figure 3.1: A convex polygonal entity can be represented by the intersection of planes. (a) – Each edge equation divides the plane into two half planes. (b) and (c) – a triangular region is formed by the intersection of three half planes.

Eventually, we may extend our representation to allow non-polygonal, convex sets of $\mathcal{W}$, in which, H_j in Equation (3.2), would be delimited by linear and non-linear functions.

3.2 Transformations and Lie Groups

In general, the spacial displacement of an entity $\mathcal{P}$ can be described with respect to the world reference frame $\{W\}$, by establishing a reference frame $\{P\}$ on $\mathcal{P}$ and describing its pose (position and orientation) with respect to $\{W\}$ using a homogeneous transformation matrix [Murray et al., 1994]. This transformation matrix can written as:

$$ {}^{W}\mathbf{T}_{P} = \begin{bmatrix} {}^{W}\mathbf{R}_{P} & {}^{W}\mathbf{r}_{O} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} , \tag{3.3} $$

where ${}^{W}\mathbf{R}_{P}$ is a rotation matrix that transforms the components of vectors in $\{P\}$ into components in $\{W\}$, and ${}^{W}\mathbf{r}_{O}$ is the position vector of the origin of $\{P\}$ with respect to $\{W\}$.

The set of all transformation matrices in $\mathbb{R}^3$ is the Lie Group $SE(3)$, the special Euclidean group of rigid body displacements in three dimensions [Murray et al., 1994]. Thus,

$$SE(3) = \left\{ \mathbf{T} | \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{r} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{r} \in \mathbb{R}^3, \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I} \right\} . \tag{3.4}$$

The group $SE(3)$ has many subgroups of interest. Among them, is the Lie group in two dimensions, $SE(2)$, that is defined by:

$$SE(2) = \left\{ \mathbf{T} | \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{r} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix}, \mathbf{R} \in \mathbb{R}^{2\times 2}, \mathbf{r} \in \mathbb{R}^2, \mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I} \right\} . \tag{3.5}$$

Practically, $SE(2)$ is defined by the set of all matrices of the form:

$$\mathbf{T} = \begin{bmatrix} \cos\theta & \sin\theta & x \\ -\sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.6}$$

where θ, x and $y \in \mathbb{R}$. Other Lie groups are the subgroups of rotations, $SO(n)$, $n = 2, 3$, and translations, $T(n)$, $n = 1, 2, 3$. As an example, the groups of rotation and translation in two dimensions, $SO(2)$ and $T(2)$, are defined, respectively, as the set of all homogeneous transformation matrices of the form:

$$\mathbf{T}_{SO(2)} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} , \quad \mathbf{T}_{T(2)} = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} . \tag{3.7}$$

3.3 Configuration Spaces

A configuration, q, for an entity, $\mathcal{P}$, is a specification of the position of every point in this entity relative to the world reference frame, $\{W\}$. The minimum number of variables (also called coordinates) to completely specify the configuration of a entity is called the *number of degrees of freedom* for that entity. In a $2D$ world, a configuration $q = (x, y, \theta)$, indicates that the entity has three degrees of freedom. Each q corresponds to a transformation, ${}^{W}\mathbf{T}_{P}$, applied to the entity. The configuration $q = (x, y, \theta)$, for instance, corresponds to a transformation matrix with the form of Equation (3.6). We will write $q \in SE(2)$ to indicate that q is a configuration that corresponds to translations and rotations in $2D$. In the same way, $q \in T(3)$ indicate that q corresponds to translations in a $3D$ world or, in other words, $q = (x, y, z)$. Extending our notation, we will write $\mathcal{P}(q)$ to represent the entity $\mathcal{P}$ at a configuration q. Thus, $\mathcal{A}_i(q)$ will represent the set description of R_i at the configuration q.

The configuration space or C-space, $\mathcal{C}$, is the set of all possible configurations, q, for a given entity. Thus, any entity can be represented by a point in the configuration space, with dimension equal to the number of degrees of freedom of the system. This idea was introduced by [Udapa, 1977] in order to represent physical robots as points and thus simplify the motion planning problem. In the configuration space, an obstacle maps to a set of configurations, $\mathcal{C}_{obs}$, which the robot touches or into which the robot penetrates. Formally, the representation of the obstacle $\mathcal{B}$ in the configuration space of the i^{th} robot is defined as:

$$\mathcal{C}_{obs\_i} = \{q \in \mathcal{C} \mid \mathcal{A}_i(q) \cap \mathcal{B} \neq \emptyset\} \ . \tag{3.8}$$

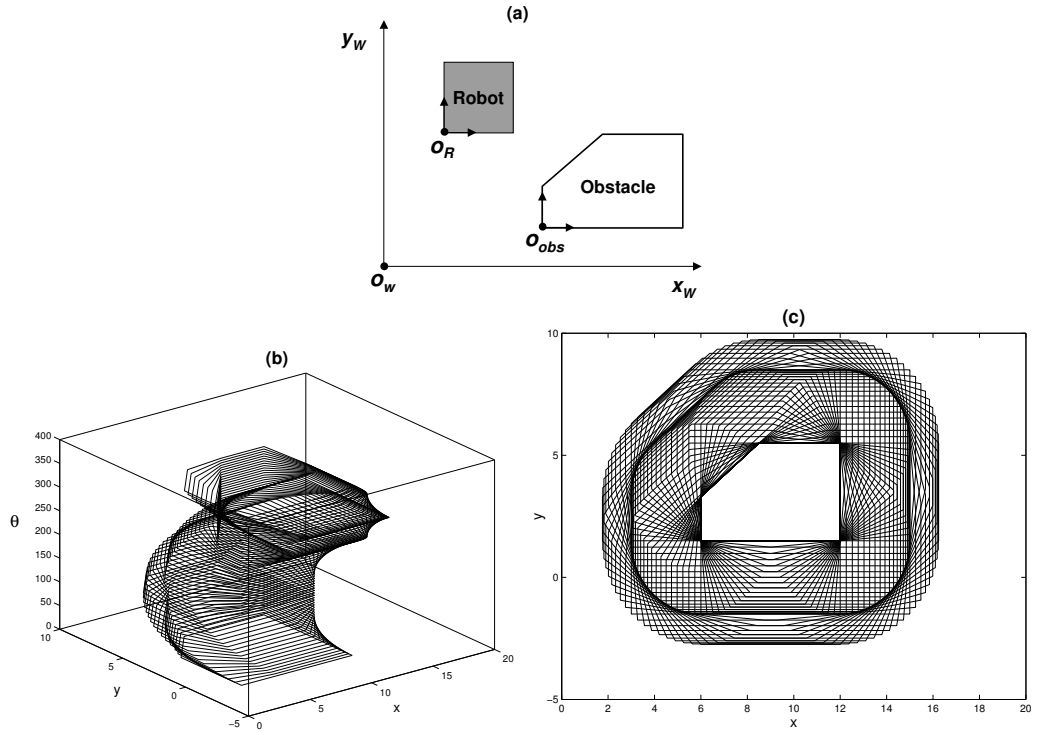

Figure 3.2: (a) – A polygonal robot and an obstacle; (b) – The representation of the obstacle in the configuration space ($\mathcal{C}_{obs\_i}$); (c) – The top view of $\mathcal{C}_{obs\_i}$

Figure 3.2 shows an example in $SE(2)$ of the representation of a 5-side polygonal obstacle in the configuration space of a square robot. The x and y axes in Figure 3.2(b) represent the possible positions of the reference point, O_R, and the third axis represents rotations in the plane. Each point in the three dimensional plot represents a possible position and orientation of the robot. The space outside the solid is a free configuration, $\mathcal{F}$, for the robot while the solid, $\mathcal{C}_{obs}$, represents intersections with the obstacle. Finding a collision-free path for a robot thus corresponds to finding a trajectory (a sequence of configurations) in the configuration space that does not intersect any $\mathcal{C}_{obs}$ [Latombe, 1991].

3.4 Sets

In previous sections we have used sets to model robots, objects and obstacles in the workspace and also their representation in the configuration space. We now make some definitions of set operations used in other chapters of this thesis.

Definition 3.1 (Minkowski sum) *The Minkowski sum of two sets $\mathcal{A}$ and $\mathcal{B}$ is defined as:*

$$\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A},\ b \in \mathcal{B}\} .$$

The Minkowski sum is an operation that preserves set convexity, *i.e.* the Minkowski sum of two convex sets is also a convex set. Also, it is well known that the Minkowski sum of two polygonal sets, $\mathcal{A}$ and $\mathcal{B}$, can be computed in linear time $O(l + m)$, where l and m are the number of edges of $\mathcal{A}$ and $\mathcal{B}$ respectively. This two properties turns out to be very useful to prove efficacy and efficiency of motion planning algorithms. We now define the Minkowski subtraction as:

Definition 3.2 (Minkowski subtraction) *The Minkowski subtraction of two sets $\mathcal{A}$ and $\mathcal{B}$ is defined as:*

$$\mathcal{A} \ominus \mathcal{B} = \{a - b | a \in \mathcal{A},\ b \in \mathbf{B}\} .$$

As a consequence of Definition 3.2 it can be observed that if $\mathcal{A} = \emptyset$ then $\mathcal{A} \ominus \mathcal{B} = \ominus\mathcal{B} = \{-b | b \in \mathcal{B}\}$. It also can be noticed that: $\mathcal{A} \ominus \mathcal{B} = A \oplus (\ominus B)$. Therefore, the Minkowski subtraction inherits the same properties of the Minkowski sum.

It can be noticed that a transformation of a set in $T(n)$, $n = 1, 2, 3$, can be seen as the Minkowski sum between the set being transformed and a set with

a single element representing the transformation. Thus, if a set $\mathcal{A}$ must be translated in $\mathbb{R}^2$, this transformation can be represented by a configuration $q \in T(n)$ and is expressed as:

$$\mathcal{C} = \{q\} \oplus \mathcal{A}\,. \tag{3.9}$$

Given Equation (3.9) we have the following proposition:

Proposition 3.1 *If* $\mathcal{I} = \{x | \mathcal{A} \cap (\mathcal{B} \oplus \{x\}) \neq \emptyset\}$ *then* $\mathcal{I} = \mathcal{A} \ominus \mathcal{B}$.

Proof:

($\subseteq$) Let y be a generic element of $\mathcal{I}$. By the definition of intersection there is a z such that $z \in \mathcal{A}$ and $z \in (\mathcal{B} \oplus \{y\})$. Thus, $z = b + y$, $b \in \mathcal{B}$, or, in other form, $y = z - b$. Since $z \in \mathcal{A}$, $y \in (\mathcal{A} \ominus \mathcal{B})$. Hence, $\mathcal{I} \subseteq (\mathcal{A} \ominus \mathcal{B})$.

($\supseteq$) Let y be a generic element of $\mathcal{A} \ominus \mathcal{B}$. Thus, $y = a - b$, $a \in \mathcal{A}$, $b \in \mathcal{B}$ or $a = b + y$. Since for all $z \in (\mathcal{B} \oplus \{y\})$, $z = b + y$, we can affirm that $a \in (\mathcal{B} \oplus \{y\})$ and therefore $\mathcal{A} \cap (\mathcal{B} \oplus \{y\}) \neq \emptyset$. In other words $y \in \mathcal{I}$. Hence, $(\mathcal{A} \ominus \mathcal{B}) \subseteq \mathcal{I}$.

Therefore, $\mathcal{I} = \mathcal{A} \ominus \mathcal{B}$. ∎

Using Proposition 3.1 it is easy to verify that, for each specific orientation, the representation of an obstacle $\mathcal{B}$ in the configuration space of robot R_i, as given by Equation (3.8), can be written as:

$$\mathcal{C}_{obs\_i} = \mathcal{B} \ominus \mathcal{A}_i\,. \tag{3.10}$$

This is an important result because it provides well known, efficient algorithms and tools to compute and work with $\mathcal{C}_{obs\_i}$.

Another important set operation is the Cartesian product, also know as cross product, and defined as:

Definition 3.3 (Cartesian product) *The Cartesian product of two sets $\mathcal{A}$ and $\mathcal{B}$ is defined as:*

$$\mathcal{A} \times \mathcal{B} = \{(a, b) | a \in \mathcal{A},\ b \in \mathcal{B}\} \ .$$

The cross product is associative, *i.e.* $(\mathcal{A} \times \mathcal{B}) \times \mathcal{C} = \mathcal{A} \times (\mathcal{B} \times \mathcal{C})$. Therefore, we can extend this definition to a set $\mathcal{A} \times \mathcal{B} \times \cdots \times \mathcal{Z}$ of ordered n-tuples for any positive integer n. Also, as the Minkowski sum, the cross product preserves set convexity — the cross product of two convex sets is also convex.

3.5 Robot Motion Planning

The robot motion planning problem can be defined as [Latombe, 1991]:

Definition 3.4 (Robot motion planning problem) *Let a single robot R in the world $\mathcal{W}$ be represented by the configuration $q \in \mathcal{C}$, and consider $\mathcal{F} \subseteq \mathcal{C}$ to be the free configuration space for R. Steer the robot from its initial configuration $q^0 \in \mathcal{F}$ at time $t = t_0$ to the desired configuration $q^d \in \mathcal{F}$ at some time $t = t_f > t_0$, such that $q \in \mathcal{F}\ \forall t \in (t_0, t_f]$.*

This problem consists of three basic subproblems: (i) – computing the free configuration space, $\mathcal{F}$, by considering the obstacles in $\mathcal{W}$ as shown in Section 3.3; (ii) – generating a path τ, which is a continuous sequence of configurations in $\mathcal{F}$ for R; and (iii) – controlling the robot to follow τ. Several methodologies have been developed to solve these problems. Some methodologies solve each problem independently and others solve them si-

multaneously. A review of methodologies to solve the robot motion planning problem can be found in [Latombe, 1991].

A very known and practical solution for the previous problem is called *Potential Field Method*. In this approach a scalar field $\Phi(q)$, called potential function, is defined over the configuration space $\mathcal{C}$. The negated gradient of the potential function, $-\nabla\Phi(q)$, is then treated as an artificial force acting on the robot, and the resultant force may be used to control the robot. The most basic instance of this approach is to assign an *attractive potential* to the goal and a *repulsive potential* to the obstacles and add them together in order to compose $\Phi(q)$. The integral curves of the vector field formed by $-\nabla\Phi(q)$, define implicit paths from every start configuration in $\mathcal{C}$ to q^d.

Typically, $\Phi(q)$ is constructed in order to define a control law for the dynamical system $\dot{q} = u$ by defining $u = -\nabla\Phi(q)$. Thus, potential field methods may be considered open loop approaches to motion planning, because they are based on a model of the environment, but they also may be viewed as closed loop approaches that rather than simply generating a geometric path, effectively assign forces or velocities to the robot. The closed loop nature is very important because it makes the planning more robust to actuator and sensor noise and external disturbances.

The main drawback of most potential field approaches is that, due to the presence of local minima in the potential function, convergence is not guaranteed. However, as proposed in [Rimon and Koditschek, 1992], in a large number of situations it is possible to construct a special potential function for which the only minimum lies at q^d. This function is called *Navigation Function* and is defined as follows [Rimon and Koditschek, 1992].

Definition 3.5 (Navigation Function) *A scalar map* $\Phi : \mathcal{F} \rightarrow [0, 1]$ *is a navigation function if it is:*

1. *smooth on* $\mathcal{F}$, i.e., *it has continuous second-order derivatives;*
2. *polar at* q^d, i.e., *has a unique minimum on the connected subset of* $\mathcal{F}$ *containing* q^d*;*
3. *admissible on* $\mathcal{F}$, i.e., *uniformly maximal on the boundary of* $\mathcal{F}$*;*
4. *a Morse function,* i.e., *its Hessian is nonsingular at the critical points.*

Requirement 1 is in place because the robot is typically commanded to track the negated gradient of the potential function and therefore smoothness is desirable. The other requirements can be used to prove convergence from almost everywhere in the environment. In spite of this, a few initial states will not converge to q^d. It can be shown that in environments with m obstacles, a navigation function will have at least m saddle-like critical points, that are unstable equilibria. At these points $\nabla\Phi(q) = 0$, and consequently the robots following $u = -\nabla\Phi(q)$ would be stuck once they reach a saddle point. However, due to sensor and actuator noise, these points generally do not cause problems. A disturbance in the estimation of q would cause the robot to compute a gradient $\nabla\Phi(q) \neq 0$ and scape this local minimum, since the Hessian of Φ is nonsingular at the critical points (Requirement 4).

Figure 3.3 shows an example of navigation function on a circular workspace with circular obstacles, $\mathcal{M}$. It can be shown that if there is a map between a generic environment $\mathcal{N}$ to this specific workspace, then there exists a navigation function on $\mathcal{N}$, and this navigation function can be obtained by a transformation of the navigation function computed on $\mathcal{M}$ [Rimon and Koditschek, 1992]. The navigation function on the trivial

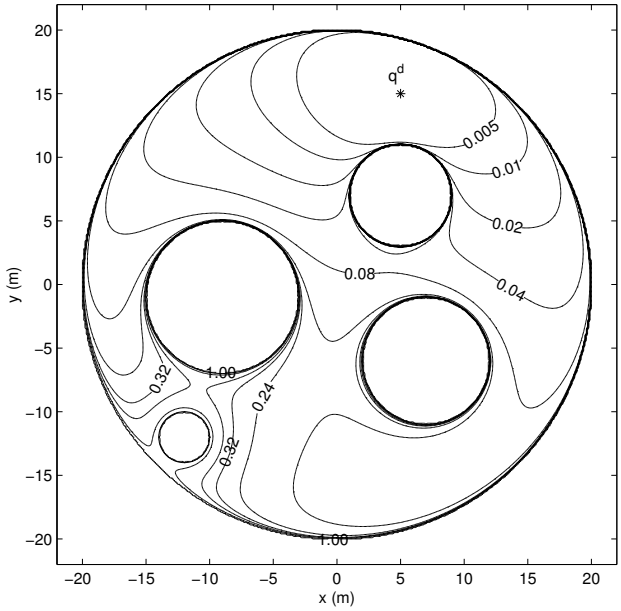

Figure 3.3: Equipotential contours of a navigation function in a circular environment with four circular obstacles.

circular workspace with m obstacles can be computed as:

$$\Phi_\kappa(q) = \frac{\|q - q^d\|^2}{[\|q - q^d\|^\kappa + \beta(q)]^\kappa}\,, \tag{3.11}$$

$$\beta(q) = \prod_{j=0}^{m} \beta_j(q)\,, \tag{3.12}$$

where β_0 is a function representing the circular boundary of the workspace with radius ρ_0 and center q_0 given by:

$$\beta_0(q) = -\|q - q_0\|^2 + \rho_0^2\,, \tag{3.13}$$

and β_j, $1 \leq j \leq m$ are functions representing the obstacles with radius ρ_j and center q_j given by:

$$\beta_j(q) = \|q - q_j\|^2 - \rho_j^2\,. \tag{3.14}$$

The parameter κ in Equation (3.11) controls the smoothness of the function and its Hessian. It can be show that, for a given workspace, all undesirable local minima disappear as the parameter κ increases [Koditschek and Rimon, 1990]. In Figure 3.3, $\kappa = 3$ was used.

3.6 Motion Constraints

We consider that the motion of an entity is described by a simple kinematic model of the form:

$$\dot{q} = f(q, u) \tag{3.15}$$

where u is the input vector of the system, which exists only if the entity is a robot.

In real world, the motion of both robots and objects are generaly constrained. In this work we will make a distinction between two groups of motion constraints. The first group, is composed of *equality constraints*, which are represented by equations of the form $g(\dot{q}, q, t) = 0$, where t is time. The second is the group of *inequality constraints*, represented by $g(\dot{q}, q, t) \leq 0$.

Equality constraints may still be further divided into two groups: *holonomic* and *non-holonomic*. Holonomic constraints are those that constrain the entity's configuration, q, while non-holonomic are those that constrain the configuration's time derivative, $\dot{q}$, but do not constrain q itself. Holonomic constraints reduce, while non-holonomic constraints preserve, the number of degrees of freedom of the system. On the other hand, non-holonomic constraints reduce the number of independent velocities for the system.

For an example of holonomic constraint, consider a car moving in a road with several slopes. Initially, the car may be represented by a configuration $q \in SO(3)$ in relation to a fixed reference frame on the earth, meaning that it

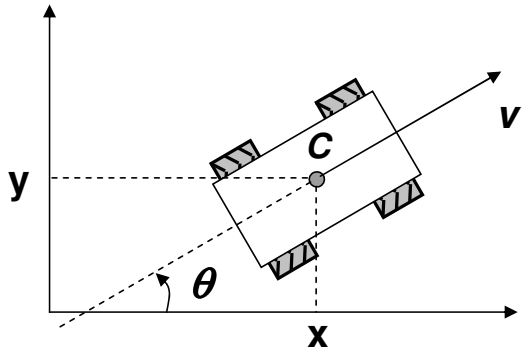

Figure 3.4: Non-holonomic constraint: The position of the car in the plane is describe by three coordinates, but its velocity can only be along the vector v. Thus, the changing rate of the three position coordinates is constrained.

has six degrees of freedom. However, if we consider that the car is constrained to be at the ground level and ignore suspension's movements (assume the car is a rigid body), we can reduce the number of degrees of freedom to three and represent the car's configuration by latitude, longitude and heading, for example. This operation is possible because there is a function $g(q)$, which is related to the topological map of the road, from which — given latitude, longitude and heading — it is possible to fully recover $q \in SO(3)$. In this case $g(q) = 0$ is a holonomic constraint for the system.

As an example of non-holonomic constraints, consider again a car, but now on a flat surface. The car is represented by a configuration $q = (x, y, \theta) \in SO(2)$ as shown in Figure 3.4. Since the car cannot instantaneously move in a lateral direction, the velocity of its center of mass, C, is a vector that lies along the longitudinal axis (shown by the vector v). In other words, the velocity component in a direction perpendicular to the motion must be zero:

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0\,, \tag{3.16}$$

which is a non-holonomic constraint for the car.

In this work we will call *non-holonomic systems* all movable entities in $\mathcal{W}$ subject to non-holonomic constraints. Also, we will call all entities that are not subject to non-holonomic constraints, by *holonomic systems*.

Besides equality constraints, dynamic systems may be subject to inequalities constraints. Consider, again, a real car. Since the linear velocity, V, of the car is limited, one constraint is clearly $V \leq V_{max}$, where V_{max} is a constant that depends on the car's maximum torque, road conditions, weather, etc. If buildings are modelled as m-sided polygons as shown in Section 3.1, another set of constraints would be $f_j(q) \geq 0$, $1 \leq j \leq m$, where q is the car configuration. This set of constraints is related to collisions between the car and the buildings. Simply put, they express the fact that building and car cannot simutaneously the same position in the space.

3.7 Lyapunov Stability

When dealing with motion planning and control algorithms, one is usually required to show the completeness of the algorithms used. By completeness we mean that a solution for the problem exists and it is correct. When a robot with a configuration q is being driven to a configuration q^d, for example, we need to show that within a finite time, t_f, $q \rightarrow q^d$. One way of discussing and verifying completeness of motion control methods, especially the reactive ones, is by checking the stability of the resulting closed loop system.

Intuitively, a system is considered unstable if a "small" change in its input creates a "large" effect in the system's output. If the resulting effect is also small, we call it a stable system. In order to provide a precise definition of stability we first should define two simple concepts:

Definition 3.6 (class K functions) *A continuous function* $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$

is of class K if it is strictly increasing and $f(0) = 0$.

Definition 3.7 (locally positive definite functions) *A continuous function* $g : \mathbb{R}^N \rightarrow \mathbb{R}$ *is locally positive definite if:*

- *g(0)=0*
- *there exists a constant* $r > 0$ *and a function* f *of class K such that:*

$$f(\|x\|) \leq g(x), \forall x \in B_r \tag{3.17}$$

where B_r *is the N dimensional ball with radius r and center in* $x = 0$.

Thus, we can define stability in the sense of Lyapunov using the following theorem:

Theorem 3.1 (Lyapunov stability theorem) *The equilibrium* $x = 0$ *of the differential equation* $\dot{x} = f(x)$ *is stable if there exists a locally positive definite function* $V : \mathbb{R}^N \rightarrow \mathbb{R}$ *and a constant* $r > 0$ *such that:*

$$\dot{V}(x) \leq 0, \forall t > t_0, \forall x \in B_r \tag{3.18}$$

where t_0 *is the initial instant of time.*

The proof for the Lyapunov's theorem is well known and can be found, for instance, in [Sastry, 1999].

An immediate application for Lyapunov's theorem is related to the navigation functions introduced in Section 3.5. It turns out that a navigation function $\Phi(q)$ is a Lyapunov function candidate because it is positive definite with equilibrium point at the goal configuration, q^d. By showing that $\Phi(q)$ is a Lyapunov function for a specific robot, we show that the robot will reach and stay at the goal configuration within a finite amount of time.

Theorem 3.2 ([Esposito and Kumar, 2002]) *A navigation function $\Phi(q)$ with goal in q^d is a Lyapunov function for the holonomic robot R with dynamics $\dot{q} = u$ controlled by $u = -\nabla\Phi(q)$.*

Proof: $\Phi(q)$ is positive definite by the definition of navigation function. We need to show that $\dot{\Phi}(q) \leq 0$.

$$\dot{\Phi}(q) = \frac{d\Phi}{dq} \cdot \frac{dq}{dt} = \nabla\Phi(q) \cdot \dot{q} = \nabla\Phi(q) \cdot u = -\nabla\Phi(q) \cdot \nabla\Phi(q) = -\|\nabla\Phi(q)\|^2 \leq 0$$

Hence, $\dot{\Phi}(q)$ always decreases along the robot's trajectory and vanishes at the goal position where $\|\nabla\Phi(q)\| = 0$. Therefore $\Phi(q)$ is a Lyapunov function for robot R. ∎

3.8 Graphs

A graph G consists of a *vertex* set $\mathcal{V} = \{v_1, v_2, \ldots v_n\}$ and a *edge* set $\mathcal{E}$, where an edge is defined by a relation between a pair of vertices of $\mathcal{V}$. In this text we will either use e_{ij} or (v_i, v_j) to denote an edge between vertices v_i and v_j. In this case we say that v_i is a *neighbor* of v_j. The *valency* of a vertex is the number of neighbors of this vertex. An edge of the type (v_i, v_i) indicates a *loop*.

Two types of graph are possible: (i) *simple graph*, when each edge is an unordered pair (*i.e.* $(v_i, v_j) = (v_j, v_i)$), and (ii) *directed graph*, when each edge is an ordered pair (*i.e.* $(v_i, v_j) \neq (v_j, v_i)$). In the later one an edge is called *arc* or simply *directed edge*. In a simple graph, in opposition to the directed one, if v_i is a neighbor of v_j then v_j is a neighbor of v_i. For directed graphs the term valency is extended: *in-valency* is applied to indicate the number of edges that points to a vertex, while *out-valency* indicates the

number of edges that points from a vertex. Throughout this thesis we will explicitly mention when we are considering directed graphs, and otherwise the term "graph" will refer to a simple graph.

A *subgraph* of a generic graph G is a graph which vertex and edge sets are subsets of vertex and edge sets of G, respectively. Then, the *spanning subgraph* of a generic graph, G, is the graph which vertex set is equal to the vertex set of G.

A *path* from v_i to v_j is a sequence of consecutive vertices starting with v_i and ending with v_j such that consecutive vertices are neighbors. We also say that a graph, G, is *connected* if there is a path between any two vertices of the graph, and *disconnected* otherwise. A *cycle* is a connected graph where every vertex has two neighbors. However, "a cycle in a graph G" refers to a subgraph of G that is a cycle. An *acyclic graph*, or *tree*, is a graph with no cycles. A spanning subgraph with no cycles is called a *spanning tree*.

We define now a *Hamiltonian Path* (HP) as a path through a graph that visits each vertex exactly once. One can observe that this subgraph can have one or none cycles. When all vertices have valency two, it has a cycle and is called *Hamiltonian Cycle* (HC). It also can be noticed that for the same graph G, there may exist several HPs and HCs. In this work we will refer to either subgraphs as *Hamiltonian Graphs* (HG). Then, by definition, each HG is a common spanning subgraph of a set of generic graphs with $\mathcal{V}$ as vertices. Figure 3.5 shows a particular HC for a set of nine vertices as a spanning subgraph of two different graphs.

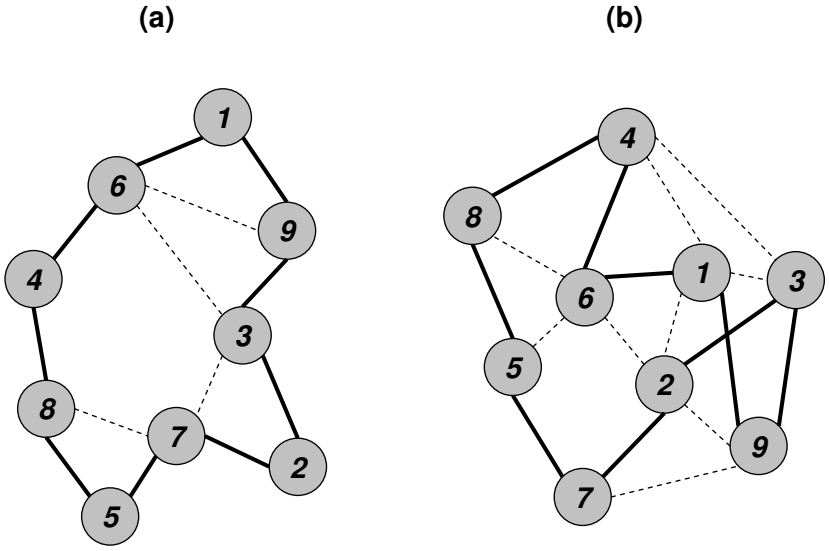

Figure 3.5: *Hamiltonian Graph* as a common spanning subgraph of two different graphs. The continuous edges represent the edges of the HG while the dashed ones are the other edges of the graph.

Chapter 4

Motion Planning with Cooperative Constraints

> Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius — and a lot of courage — to move in the opposite direction.
>
> *Albert Einstein (1879–1955)*

This chapter presents our framework for motion planning and control of cooperating robots. The problem definition is presented in Section 4.1 and our decentralized solution for a specific theoretical robot is developed in Section 4.2. Generalization of the approach for generic, real robots is presented in Section 4.3.

4.1 Problem Definition

The previous chapter introduced the necessary background for a clear understanding of the problem considered in this thesis. This problem will be formally defined in this section. We start by extending the motion planning problem presented in Section 3.5 in order to account for multiple cooperating mobile robots. The basic multi-robot motion planning problem is to find a motion plan for all the robots in a group such that each robot reaches its goal while avoiding collisions with other robots and with the obstacles

in the environment. We define the *coordinated motion planning problem*, where besides avoiding collisions, robots need to cooperate and maintain constraints imposed by the task and other robots in order to reach their goals [Pereira et al., 2003a, Pereira et al., 2003e].

Definition 4.1 (Coordinated motion planning problem) *Consider a world, $\mathcal{W}$, occupied by a set, $\mathcal{R}$, of n robots. The i^{th} robot R_i can be represented by a configuration q_i in the configuration space $\mathcal{C}$. Let $\mathcal{F}_i \subseteq \mathcal{C}$ denote the free configuration space for R_i. Additionally, let $\mathcal{C}_{R_i}(\mathcal{R} \backslash R_i, t) \subseteq \mathcal{F}_i$ denote R_i's valid configuration space imposed by its* formation constraints*. Steer each robot, R_i, $1 \leq i \leq n$, from a initial configuration q_i^0 at time $t = t_0$ to the goal configuration $q_i^d \in \mathcal{F}_i$ at some time $t = t_f > t_0$ such that $q_i \in \mathcal{C}_{R_i}(\mathcal{R} \backslash R_i, t)$ $\forall t \in (t_0, t_f]$.*

Formation constraints are constraints on individual robots induced by the other robots in the team. Thus, $\mathcal{C}_{R_i}(\mathcal{R} \backslash R_i, t)$, $i = 1, 2, \ldots, n$, depend on the robots' characteristics, their configurations, and also on the nature of the task. Notice that our problem statement differs from the previous definition of multi-robot motion planing problem in the sense that, besides inter-robot collisions, we consider other kinds of constraints that include, for example, sensor field-of-view constraints and communication range constraints. It is interesting to note that our definition of multi-robot motion planning is not too different from the definition of robot motion planning for a single robot. While the traditional definition considers the problem of moving the robot in a limited free space, where the constraints are induced by the (non-controlled) obstacles in the environment, here part of the valid (free) configuration space for the robots is also induced by the position of the other robots.

We will show by examples in later chapters that our problem definition is quite generic, since it captures several cooperative tasks. The basic difference

among the tasks is the shape of the formation constraints. Consequently, if an approach to solve a given cooperative problem is conceived, it can be probably used to solve other problems with minor modifications. In the chapters which follow we will show how some apparently very different problems can be solved using the same motion planning approach. This approach will be discussed in the rest of this chapter.

4.2 Motion Planning Approach

We will now present a methodology to solve the cooperative motion planning problem defined in Section 4.1 for a planar world $\mathcal{W} = \mathbb{R}^2$. Instead of developing one centralized algorithm for coordinating the motion of all the robots, we develop a decentralized algorithm which allows each robot to choose its own motion based on the available free space and formation constraints. We assume a two-level motion planner. The higher level specifies a deliberative plan [Arkin, 1998] in terms of previously computed navigation functions for each robot, and desired *neighborhood relationships*. Navigation functions were introduced in Section 3.5. Neighborhood relationships are pairwise formation constraints that are formalized in Section 4.2.1. The lower level of the planner is the main focus of this work. We address real time modification of pre-planned functions computed by the deliberative controller in order to accommodate formation constraints. Before we proceed further, we will make a few assumptions. Then, in Section 4.3 we will discuss relaxing these assumptions.

Assumption 4.1 All robots are identical in terms of geometry, and in terms of capabilities and constraints related to sensing, communication, control, and mobility.

Assumption 4.2 The robots are point robots: $q_i = (x_i, y_i)$.

Assumption 4.3 The robots are holonomic. For the i^{th} robot, the dynamical model is then given by: $\dot{q}_i = u_i$.

4.2.1 Neighborhood Relationships

The robot's physical locations coupled with the characteristics of the hardware and the requirements of the task dictate the *neighborhood relationship* for a group of robots. This relationship can be represented by a graph G, where the robots themselves are the vertices and the relationship itself is represented by directed edges or arcs. A graph is represented by a tuple $(\mathcal{R}, \mathcal{E}, \mathcal{G})$, where $\mathcal{R}$ is the set of robots, $\mathcal{E} \subseteq \mathcal{R} \times \mathcal{R}$ is the edge set representing the relationship among the robots, and $\mathcal{G}$ is the set of constraint functions that describe the conditions under which each edge is maintained. For each element of $\mathcal{E}$ there is, at least, one corresponding element in $\mathcal{G}$. When, for example, the task requires interaction between robots R_i and R_j, this interaction is represented by the edge $e_{ij} = (R_i, R_j) \in \mathcal{E}$ and a set of functions $\mathcal{G}_{ij} \subseteq \mathcal{G}$, where $\mathcal{G}_{ij} = \{g_1(q_i, q_j) \dots g_m(q_i, q_j)\}$. In this case we also say that R_j is a *neighbor* of R_i. Under the assumption that robots are identical, the graphs are in general bidirectional and therefore undirected. In this case, $(R_i, R_j) \in \mathcal{E}$ is equivalent to $(R_j, R_i) \in \mathcal{E}$ and $\mathcal{G}_{ij} = \{g_1(q_i, q_j) \dots g_m(q_i, q_j)\}$ is identical to $\mathcal{G}_{ji} = \{g_1(q_j, q_i) \dots g_m(q_j, q_i)\}$. We assume that there are no loops or, in other words, there are no edges of the form (R_i, R_i).

Formation Constraints and Valid Configuration Spaces

For each constraint function in $\mathcal{G}$ we associate at least one inequality of the form $g_k(q_i, q_j) \leq 0$, which describes a *formation constraint*. The existence

of an edge between R_i and R_j is conditioned to the satisfaction of these inequalities. Hence, the edge between R_i and R_j is removed from the graph as soon as, at least one constraint in $\mathcal{G}_{ij}$, say $g_k(q_i, q_j)$, is strictly greater than zero.

If we use the boundary representation presented in Section 3.1, the set of functions $\mathcal{G}_{ij}$ can be seen as boundaries for configuration spaces. Then, $\mathcal{G}_{ij}$, is a representation of the region γ_{ij} in R_i's configuration space defined by the intersection of m planar regions as:

$$\gamma_{ij} = \bigcap_{k=1}^{m} H_k \,, \tag{4.1}$$

where,

$$H_k = \{(q_i, q_j) \in \mathbb{R}^2 \times \mathbb{R}^2 \mid g_k(q_i, q_j) \leq 0\}, \qquad 1 \leq k \leq m \,. \tag{4.2}$$

In order to better understand the physical meaning of the constraint set $\mathcal{G}_{ij}$, suppose R_j is static with fixed position $q_j = (x_j, y_j)$. In this case each $g_k(q_i, q_j)$ can be written as $g_k(q_i)$, a function of q_i only. Since $g_k(q_i) \leq 0$ describes a condition under which the edge e_{ij} is maintained, we can observe that in order to preserve this edge, q_i must belong to the set, γ_{ij}, represented by $\mathcal{G}_{ij}$. In this case $g_k(q_i) \leq 0$, $1 \leq k \leq m$ is always satisfied.

For example, consider a group of robots communicating through a wireless ad hoc network, and that the only task the robots are suppose to perform is to drive towards their goals while maintaining communication with the others (this specific task will be addressed in detail in Chapter 5). The group is represented by its communication graph G_{cn}. The existence of a communication link between R_i and R_j is represented by the edge e_{ij} and a single function $g(q_i, q_j)$, which is directly related to the maximum range of

their antennas. The communication link between the robots is maintained only if constraint $g(q_i, q_j) \leq 0$ is satisfied or, in other words, if R_i is within the communication range of R_j and *vice-versa*. If we consider identical robots and omnidirectional antennas, we restrict our attention to undirected graphs and represent the constraint function $g(q_i, q_j)$ by a circle with center in q_j and radius r, representing the communication range. Because our graph is undirect, the constraint induced in R_j, $g(q_j, q_i)$ can be thought of as another circle with center q_i and radius r. Both circles define regions for q_i and q_j where communication between R_i and R_j is guaranteed.

The previous formulation defines a configuration space for a robot R_i in function of one of its neighbors, R_j, where formation constrains are satisfied. The intersection of the configuration spaces induced by all neighbors of R_i defines the region for q_i where all neighborhood relationships are maintained. This is represented as:

$$\Gamma_i = \bigcap_k \gamma_{ik}, \quad \forall (R_i, R_k) \in \mathcal{E}\,. \tag{4.3}$$

Analogously to the problem in Definition 4.1, one can see that the region in the configuration space defined by the constraints in $\mathcal{G}$ and represented by Γ_i, directly determines the valid configuration space for R_i:

$$\mathcal{C}_{R_i}(\mathcal{R}\backslash R_i, t) = \mathcal{F}_i \cap \Gamma_i\,. \tag{4.4}$$

Section 4.2.2 will show a reactive approach to control the shape of $\mathcal{C}_{R_i}(\mathcal{R}\backslash R_i, t)$, $1 \leq i \leq n$, in order to guarantee that the formation constraints are satisfied.

Constraint Activation

A specific constraint $g_k(q_i, q_j) \leq 0$ is active when $g_k(q_i, q_j) = 0$. Then, a constraint activation means that q_i is on the border of Γ_i and, since $g_k(q_i, q_j)$ is identical to $g_k(q_j, q_i)$, q_j is on the border of Γ_j. Thus, observe that, regardless the number of constraints and the number of neighboring robots, when one constraint is active for robot R_i another identical constraint is active for one of its neighbors. This observation will be very important in our control design.

Our objective is to control the robots in order to maintain q_i inside Γ_i. Hence, when a given constraint is active, our control laws must act in order to prevent q_i from moving outside Γ_i. Thus, in order to account for sensors and actuators dynamics and uncertainty, instead of considering the border of Γ_i as the limit for the activation of the controllers, it would be adequate to consider a constraint to be activated whenever $g_k(q_i, q_j) = \delta$, where δ is a negative number. The value of δ is highly dependent on the task and on the hardware characteristics. Consider, for example, that we are working with holonomic robots with perfect actuators, but real sensors, which provide relative positioning to other robots with zero error mean and standard deviation σ. In this case, $\delta = -2\sigma$ would be a good choice since this would guarantee that the robots would be aware of constraint activation in approximately 95% of the cases, by the definition of standard deviation. In another example, δ could be chosen to take into account different robots' dynamics. If, for instance, the robots cannot instantaneously respond to the controller's output, one suggestion is to choose δ proportional to the robots' time response.

In this thesis, formation constraints define three regions in the robot's configuration space. These regions are hierarchicaly represented as seen in

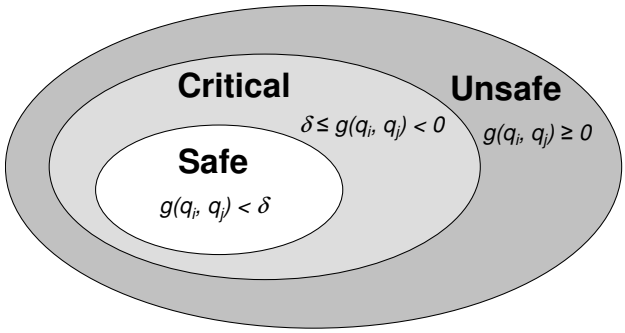

Figure 4.1: The activation of the constraints define three regions in the robots' configuration space.

Figure 4.1. In the *safe region*, each constraint satisfies $g(q_i, q_k) < \delta$. The region defined by $\delta \leq g(q_i, q_k) < 0$ is the *critical region* for the robot. If $g(q_i, q_k) \geq 0$ the robot is outside the valid configuration space that we call *unsafe region*. Depending on the nature of the constraints, the robots may not be able to return to the safe region of the configuration space if they reach the unsafe region. Our decentralized controllers are designed with the objective of keeping the robots inside the safe configuration space.

Essential and Desirable Relationships

The set Γ_i is the region of the configuration space where R_i preserves all current neighbors. However, depending on the nature of the task, R_i does not need to keep all of its neighbors. We can thus divide the neighboring robots of R_i into three groups: (i) *essential neighbors* — robots which relationship with R_i is necessary for task completion; (ii) *desirable neighbors* — robots which relationship with R_i improve the execution of the task; and (iii) *non-essential neighbors* — robots which relationship with R_i does not change the execution of the task.

In our communication example, if each robot keeps communication constraints with two other robots, forming a network similar to a ring, we may

say that all neighbors of a generic robot R_i are essential, because otherwise, the network will be disconected[1]. On the other hand, in order to improve bandwidth and decrease communication delay, it would be good to increase the number of neighbors of each robot[2]. In this case, neighbors other than the essential ones are desirable. If the number of neighbors is further increased, a situation may happen where adding new robots will not increase the performance of the communication, and therefore they will be non-essential.

The specification of the neighborhood relationships and the form of the constraints is highly dependent on the task and will be discussed in the chapters that follow of this thesis.

4.2.2 Decentralized Controllers

Our control system is decentralized and implemented using a set of reactive controllers, one for each space defined by the formation constraints (Figure 4.1). Each robot switches between these controllers as shown in Figure 4.2. Since switching is governed by constraints activation, it depends on the relative positioning of a robot with respect to its neighbors. Although the controller has basically three modes, each mode contain other switched controllers. The basic modes represent the three basic operations necessary for the completion of the cooperative task: (i) satisfy the formation constraints; (ii) maintain the constraints while navigating to the goal; and (iii) go to the goal without any reference to the other robots.

As introduced at the beginning of Section 4.2, a navigation function for

[1]Actually two robots in the team may have only one essential robot. In this situation the network will not have a cycle.

[2]Bandwidth would be increased because more paths would be used between two communicating robots. Delays would be reduced because, on average, the number of nodes between two robots would decrease.

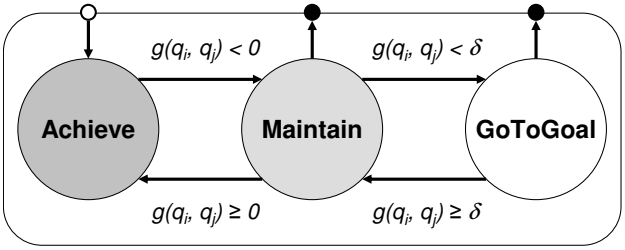

Figure 4.2: Switched control system with three modes.

solving the non-cooperative problem of steering each individual robot towards the goal while avoiding static obstacles in the environment is assumed to be available from a higher level planner.

When robots are cooperating, the formation constraints will force them to navigate near each to other. Also because they are performing tightly coupled tasks, their final goals must be reasonably close. Therefore, the gradients of the navigation functions for neighboring robots are similar. Based on this observation, our fourth assumption is:

Assumption 4.4 For any pair of cooperating robots, R_i and R_j,$\nabla\Phi_i(q_i) \cong \nabla\Phi_j(q_j)$, where Φ_x is the navigation function for R_x.

This assumption is required for the proof of Proposition 4.2. Obviously, if the robots have goals that are not close to each other and the robots are near to their destinations, this assumption is not valid since their navigation functions might be very different.

We will develop a controller that allows each robot to have up to two assigned essential neighbors and several desirable ones. In other words it guarantees connection with up to two robots and attempts to maintain several other links. The specification of the neighbors is also available from a higher level planner. The two essential neighbors of R_i will be denoted by R_a and R_b. Denote the constraints due to R_a, which have the form $g_k(q_a, q_i) \leq 0$,

by g_k^a and those due to R_b, which have the form $g_k(q_i, q_b) \leq 0$ by g_k^b. In general, constraints induced by R_x will be denoted by g_k^x.

In the ACHIEVE mode each robot tries to move in order to satisfy the constraints induced by both R_a and R_b, without using the navigation function. In other words, the constraints themselves act as potential fields attracting the robots to each other and forcing them into a feasible configuration that satisfies all the constraints. The control input in this mode is:

$$u_i = -k_1 \left(\alpha \nabla g^a + \beta \nabla g^b \right) , \tag{4.5}$$

where ∇g^x is a unit vector along the gradient of the constraint defined by:

$$\nabla g^x = \frac{\partial g^x / \partial q_i}{\| \partial g^x / \partial q_i \|} .$$

∇g^a is due to R_a and ∇g^b is due to robot R_b. The variables α and β assume value 1 or 0 depending whether the constraints are active or not, respectively and k_1 is a positive constant.

In the GOTOGOAL mode, the robots move towards the goal with the following input:

$$u_i = -k_2 \nabla \phi_i , \tag{4.6}$$

where $\nabla \phi_i = \nabla \Phi_i(q_i) / \| \nabla \Phi_i(q_i) \|$ is the normalized gradient vector of the navigation function $\Phi_i(q_i)$. As mentioned before, it is a deliberative controller with a pre-planned navigation function that guides the robots toward the goal. In this mode, no information of other robots is used.

In the MAINTAIN mode, a robot attempts to navigate toward the goal while maintaining the formation constraints. The control input for this state

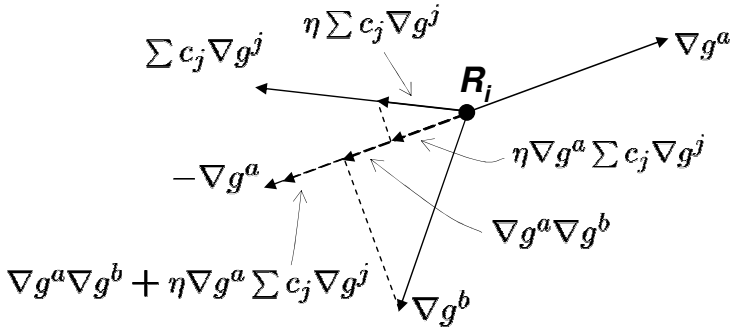

Figure 4.3: η is chosen in order to avoid R_i to move in a direction contrary to ∇g^a and ∇g^b. Only the constraint relative to ∇g^a is illustrated by simplicity. Observe that a small η was chosen such that $\nabla g^a \nabla g^b + \eta \nabla g^a \sum c_j \nabla h^j < -\nabla g^a$

is given by:

$$u_i = -k_1 \left(\alpha \nabla g^a + \beta \nabla g^b + \eta \sum_j c_j \nabla g^j \right) - k_2 \nabla \phi_i \ , \tag{4.7}$$

where $k_2 > 3k_1$ in order to guarantee convergence to the goal, as will be shown later in the proof of Proposition 4.2. In this equation α, β, and c_j can each be 0 or 1. When $g \leq \delta$, the value 0 is assigned, and when $g > \delta$ the value 1 is assigned. The summation $\sum c_j \nabla g^j$ is the part of the control action due to the desirable edges in the graph and corresponds to the summation of the active constraints induced by the desirable neighbors of R_i. The non-negative gain η must be properly chosen in order to satisfy the following constraints:

$$\begin{aligned} -1 < \alpha\beta \nabla g^a \nabla g^b + \eta\beta \nabla g^b \sum c_j \nabla g^j \leq 2 \\ -1 < \alpha\beta \nabla g^a \nabla g^b + \eta\alpha \nabla g^a \sum c_j \nabla g^j \leq 2 \end{aligned} \tag{4.8}$$

As shown in Figure 4.3, these two constraints for η are used in order to guarantee, by controlling the magnitude of $\eta \sum c_j \nabla g^j$, that (i) the robots

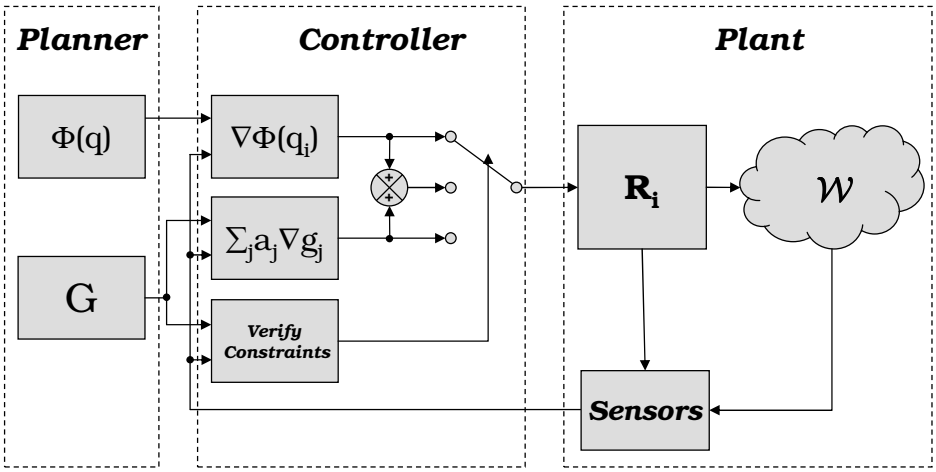

Figure 4.4: Block diagram of the system.

will never go in directions contrary to ∇g^a and ∇g^b, and (ii) the robots will never go in directions contrary to $\nabla \phi_i$. The value of η can then be computed as the maximum non-negative value that satisfies the constraints using an optimization algorithm. Observe that η is zero when ∇g^a and ∇g^b are anti-parallel vectors. In this case, the robots give up keeping the desirable links and only worry about the necessary ones.

Observe that Equation (4.7) includes the two other control modes. If $k_2 = \eta = 0$ it becomes Equation (4.5) and if $k_1 = 0$ it becomes Equation (4.6). Hence, the previous control system can be represented by a single block diagram such as the one shown in Figure 4.4. In this figure the block *Verify Constraints* is responsible for choosing one of three controllers to be used to control de robot. Notice that this block is in the part of the system that we call *Controller*. This is the reactive part of the robot control system while *Planner* is the deliberative one. The planner includes navigation function and graph design.

Control laws (4.5), (4.6) and (4.7) solve the n problems of individually leading the robots to their goals while guaranteeing the formation constraints are satisfied.

Proposition 4.1 *If the robots start in a feasible configuration,* i.e.*, a configuration which satisfies all formation constraints, the switched control law represented by (4.5), (4.6) and (4.7) guarantees that those constraints are satisfied during the motion of the robots.*

Proof: We consider a generic constraint involving a generic pair of robots R_i and R_j, $g(q_i, q_j) \leq \delta$, and show when the constraint is active, the control input makes $\dot{g}(q_i, q_j) \leq 0$. The time derivative of $g(q_i, q_j)$ is given by:

$$\dot{g}(q_i, q_j) = \frac{\partial g}{\partial q_i}\dot{q}_i + \frac{\partial g}{\partial q_j}\dot{q}_j \ . \tag{4.9}$$

For the i^{th} robot, if $g(q_i, q_j)$ is active, then for the j^{th} robot, $g(q_j, q_i)$ is also active. In the control law (4.7), $\nabla g^a = \nabla g^j$ for R_i and $\nabla g^b = \nabla g^i = -\nabla g^j$ for R_j. Let ∇g^α be the term associated with the constraint induced by the other necessary neighbor of R_i and ∇g^β be the term associated with the constraint induced by the other necessary neighbor of R_j. Substituting for $\dot{q}_i$ and $\dot{q}_j$ in (4.9) from (4.7), the time derivative of $g(q_i, q_j)$ is given by:

$$\begin{aligned}
\dot{g}_j(q_i, q_k) =& \frac{\partial g}{\partial q_i} \cdot \left[-k_1 \left(a\nabla g^\alpha + \nabla g^j + \eta_i \sum c_k \nabla g^k\right) - k_2 \nabla \phi_i\right] + \\
&+ \frac{\partial g}{\partial q_j} \cdot \left[-k_1 \left(-\nabla g^j + b\nabla g^\beta + \eta_j \sum c_l \nabla g^l\right) - k_2 \nabla \phi_j\right] \\
&= \left\|\frac{\partial g}{\partial q_i}\right\| \nabla g_j^k \cdot \left[-k_1 \left(a\nabla g^\alpha + \nabla g^j + \eta_i \sum c_k \nabla g^k\right) - k_2 \nabla \phi_i\right] + \\
&- \left\|\frac{\partial g}{\partial q_i}\right\| \nabla g_j^k \cdot \left[-k_1 \left(-\nabla g^j + b\nabla g^\beta + \eta_j \sum c_l \nabla g^l\right) - k_2 \nabla \phi_j\right] \\
&= \left\|\frac{\partial g}{\partial q_i}\right\| \Big[-k_1 \Big(\nabla g^j \cdot \nabla g^j + a\nabla g^j \cdot \nabla g^\alpha + \eta_i \ g^j \cdot \sum c_k \nabla g^k\Big) + \\
&- k_2 \nabla g^j \cdot \nabla \phi_i + \\
&- k_1 \Big((-\nabla g^j) \cdot (-\nabla g^j) + b(-\nabla g^j) \cdot \nabla g^\beta + \eta_j (-\nabla g^j) \cdot \sum c_l \nabla g^l\Big) + \\
&- k_2 \left(-\nabla g^j\right) \cdot \nabla \phi_j\Big] \ .
\end{aligned}$$

Under the assumption that $\nabla\phi_i = \nabla\phi_j$ we have:

$$\dot{g}(q_i, q_k j) = -\left\|\frac{\partial g}{\partial q_i}\right\| \Big[k_1\left(1 + a\nabla g^j \cdot \nabla g^\alpha + \eta_i \nabla g^j \cdot \sum c_k \nabla g^k\right) + \\ + k_1\left(1 + b(-\nabla g^j) \cdot \nabla g^\beta + \eta_k(-\nabla g^j) \cdot \sum c_l \nabla g^l\right)\Big] \leq 0\,,$$

since η_i and η_k were chosen in order to satisfy constraints (4.8).

If $\nabla\phi_i \neq \nabla\phi_j$ eventually, $\dot{g}(q_i, q_k) > 0$. In these cases, at a certain configuration, we can have the activation of the constraint $g(q_i, q_j) \leq \delta$ and consequently the control will switch to the ACHIEVE mode. The control law in this mode assumes that $\nabla v_i = \nabla v_j = 0$, what results, by the previous conclusion, that the constraints are preserved. Therefore, given the initial conditions, $g(q_i, q_k) \leq 0$, for all $1 \leq i \leq n$ and $k \in \{l, r\}$, and the fact that the derivatives $\dot{g}(q_i, q_j) \leq 0$ when the this constraint is active, the proposition is proved. ∎

Proposition 4.2 *If the robots' start and goal positions are valid configurations and during the motion the gradient of the navigation function of two neighboring robots can be considered the same, the switched control law represented by (4.5), (4.6) and (4.7) leads the robots to their respective goals.*

Proof: Observe by the proof of Proposition 4.1 that if the robots' initial configuration satisfies $g(q_i, q_r) \leq \delta$ and $\nabla\phi_k = \nabla\phi_i$, then they will never switch to the ACHIEVE mode of the control law. Then, only equations (4.7) and (4.6) will be used as control inputs. Thus, observe that $\Phi_i(q_i)$, which is locally positive definite, is a common Lyapunov function for both modes:

$$\frac{d\Phi_i(q_i)}{dt} = \dot{\Phi}_i(q_i) = \nabla\Phi_i \cdot \dot{q}_i = \nabla\Phi \cdot u_i\,, \tag{4.10}$$

GoToGoal Mode:

$$\dot{\Phi}_i = \nabla\Phi_i \cdot u_i = -k_2\|\nabla\Phi_i\| \cdot \nabla\phi_i \cdot \nabla\phi_i = -k_2\|\nabla\Phi_i\|^2 \leq 0\,,$$

Maintain Mode:

$$\begin{aligned}\dot{\Phi}_i &= -\nabla\Phi_i \left[k_1 \left(a\nabla g^r + b\nabla g^l + \eta_i \sum c_j \nabla g^j\right) + k_2\nabla\phi_i\right] \\ &= -\|\nabla\Phi_i\| \cdot \nabla\phi_i \left[k_1 \left(a\nabla g^r + b\nabla g^l + \eta_i \sum c_j \nabla g^j\right) + k_2\nabla\phi_i\right] \\ &= -\|\nabla\Phi_i\| \left[k_1 \left(a\nabla\phi_i \cdot \nabla g^r + b\nabla\phi_i \cdot \nabla g^l + \eta_i\nabla\phi_i \sum c_j \nabla g^j\right) + \right. \\ &\qquad \left. + k_2\nabla\phi_i \cdot \nabla\phi_i\right] \leq 0\,,\end{aligned}$$

since constraints (4.8) guarantee η_i such that the term in parenthesis is non-negative and smaller than or equal to 3, and $k_2 > 3k_1$. Therefore, in these two modes the control law are free of local minima, since $\nabla\Phi_i = 0$ if and only if $q_i = q_i^d$ by the definition of the navigation function. ∎

One may question the assumption that $\Phi_i(q_i) = \Phi_k(q_k)$, which underlies the validity of Proposition 4.2. Although the validity of this assumption may be questioned, it is a good assumption when the robots are far away from their respective destinations, as discussed in the beginning of this subsection. Also, it is important to notice that convergence to the goal will only be guaranteed if the constraints are satisfied when the robots are at the goal position. More specifically:

$$\left(q_1^d, q_2^d, \ldots, q_n^d\right) \in \mathcal{C}_{R_1}(t) \times \mathcal{C}_{R_2}(t) \times \cdots \times \mathcal{C}_{R_n}(t)\,, \tag{4.11}$$

for $t \to \infty$. If this constraint is not satisfied, the robots would need to eventually violate the formation constraints in order to reach their goals. Since this is not allowed by our controllers, as shown in the proof of Proposition 4.1, the robots will be stuck in a local minima introduced by the Achieve mode of

the controller. If this behavior is not acceptable, a high level planner should act and change the graph, G, which define the neighborhood relationship, during task execution.

It is more difficult to prove that the control law (4.5) in the ACHIVE mode leads to a condition where all constraints are satisfied. The main difficulty come from the fact that no potential function is used and therefore obstacles are not avoided. However, this problem can be avoided if the robots start in favorable initial conditions. The specification of a controller to initially position the robots is a direction of future research.

4.3 Extension to Real Robots

Our control laws were derived under Assumptions 4.1 – 4.3. However, at least one of these assumptions does not hold for most real robots, and therefore it is natural to ask whether our methodology can be applied for other kinds of robots.

Assumption 4.1, which is relative to identical neighboring robots, cannot be entirely relaxed in applications where formation constraints depend on the physical characteristics of the robots. It happens because our proofs need the constraints for both neighboring robots to be simultaneously active and therefore, they need to be identical. However, a simple way to avoid this problem is to choose common constraints for each pair of robots. For example, consider a group of robots maintaining communication constraints. In this case, it is important that the robots' antennas present the same characteristics. If they do not, a very good approximation is to use the constraints related to the lower performance antenna of the two neighboring robots. For this specific situation, other differences such as mobility, size of

the robots, and so on, do not affect the algorithm.

Assumption 4.2 , which considers point robots, is easily relaxed if the obstacles are dilated by the size of the robots before the navigation function construction, as shown in Section 3.3. Some constraints may also need to be changed depending on the robot's shape. We will address this issue in the next chapters, using specific examples.

Assumption 4.3 is more difficult to relax. For non-holonomic robots, u_i, which is a two dimensional vector, can be used as set-point for controllers that account for non-holonomic constraints. In the next subsection we derive one of these controllers.

4.3.1 Non-Holonomic Robots

As mentioned before, because these systems are underactuated, we cannot directly apply our methodology for non-holonomic robots. Our proposal is simply apply some form of feedback linearization [Sastry, 1999] in order to use the ideal holonomic input to obtain an input for the non-holonomic robot. We cannot prove that constraints will always be satisfied but, from a practical standpoint, we still have the activation values δ that can be varied in order to compensate for differences in the robots dynamics.

In order to derive a controller for a non-holonomic robot, consider that its kinematic model is given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.12}$$

where x and y are described with respect to a fixed reference frame $\{U\}$ and θ is the orientation of the robot frame $\{R\}$ in relation to $\{U\}$ (see Figure 4.5).

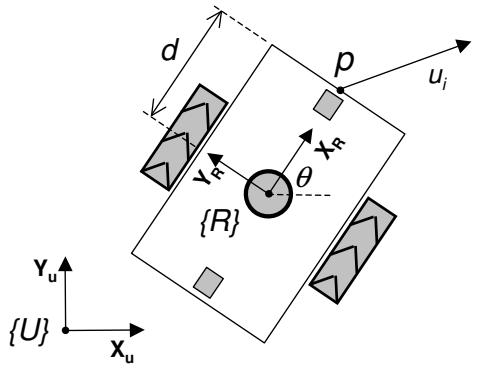

Figure 4.5: A non-holonomic robot.

The linear and angular velocities, v and ω respectively, are the robot's input. A natural output, z, for the system is the robot's position (x, y) in relation to $\{U\}$. The time derivative of the output is then:

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{A} \begin{bmatrix} v \\ \omega \end{bmatrix} . \tag{4.13}$$

We can not use Equation (4.13) to solve our control problem using feedback linearization because one of the inputs (angular velocity) does not appear explicitly. In other words, the decoupling matrix of the system, $\mathbf{A}$, is singular. A possible way to circumvent this problem, which is shown in Figure 4.5, is to redefine the system output to be $(x_p, y_p) = (x + d\cos\theta, y + d\sin\theta)$ which corresponds to the coordinates of a point p located at position $(d, 0)$ in the robot reference frame. Using this point, the derivative of the output can be rewritten as:

$$\dot{z} = \begin{bmatrix} \dot{x} + d\,\dot{\theta}\sin\theta \\ \dot{y} - d\,\dot{\theta}\cos\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & +d\sin\theta \\ \sin\theta & -d\cos\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{A} \begin{bmatrix} v \\ \omega \end{bmatrix} . \tag{4.14}$$

Since the determinant of $\mathbf{A}$ is equal to $-d \neq 0$, we can solve Equation (4.14)

for the inputs as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{A}^{-1}\dot{z} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{d} & \frac{\cos\theta}{d} \end{bmatrix} \dot{z} \ . \tag{4.15}$$

Therefore, if we consider that the original velocity vector u_i is applied to p, and not to the center of the robot, we can control the non-holonomic robot using w and v given by:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{d} & \frac{\cos\theta}{d} \end{bmatrix} u_i \ . \tag{4.16}$$

Since u_i may be calculated in the robot's reference frame, we can assume that the origin of $\{U\}$ is located at the point p fixed to the robot body and the X_u axis is always parallel to u_i. In this way, the y component of u_i can be considered to be null and θ is the angle between u_i and X_R. We can rewrite our control law as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} k\,\dot{x}_u\,\cos\theta \\ -\frac{k\,\dot{x}_u}{d}\sin\theta \end{bmatrix} u_i \ . \tag{4.17}$$

The next three chapters show examples of cooperative tasks using the approach presented in this chapter. Experimental results with cooperating polygonal non-holonomic robots and also with circular holonomic robots are shown.

Chapter 5

Application to Sensing and Communication

This chapter presents the first application of the methodology presented in Chapter 4. Communication and sensing requirements are used to constrain the robots' motion in a simple go-to-goal task. Section 5.1 presents an introduction and a literature review. A discussion about neighborhood relationships for this specific task is found in Section 5.2. The use of the control laws presented in Chapter 4 for maintaining sensing and communication is addressed in Section 5.3. Experiments are presented in Section 5.4, and concluding remarks in Section 5.5.

5.1 Introduction

Cooperating mobile robots interact with each other using either sensor observations or deliberate exchange of messages through a wireless network. A fundamental limitation related to these forms of interaction among robots is the limited field of view of the physical sensors and the limited range of transmitters and receivers. When communication is essential to the completion of the specified task, the robots must move in order to maintain such constraints. In this chapter, we address the problem of controlling the motion of a team of mobile robots subject to such communication and sensing

constraints using the *formation constraints* introduced in Chapter 4.

Just a few works explicitly consider the problem of planning and controlling multiple mobile robots's motion by taking into account sensing and communication constraints. Actually, as discussed in [Ando et al., 1995], the correctness proofs of the algorithms for robots with limited visibility of its teammates can be considerably more complex than those for robots with unlimited visibility. In that paper, the authors propose a reactive algorithm to move a set of robots to a single point while maintaining connectivity of a sensing graph. In [Pimentel, 2002], the author proposes a methodology where each robot has a local planning algorithm that continuously provide local optimal actions based on a prediction model of a near-future communication network topology. Prediction is based on local state information broadcasted by each robot in the team. The algorithm is evaluated for a search and rescue task. More recently, [Lian and Murray, 2003] proposed the use of constraints induced by the range of sensing and communication devices in a real-time trajectory generator. Since trajectory generators may include other constraints such as obstacles in the environments, their approach presents guarantees for task completion. They show, in simulation, results where a group of robots is coordinated in order to perform go-to-goal like tasks.

As mentioned before, we use reactive controllers to "replan" the robots trajectories in real time. Due to the properties of the navigation function discussed in Chapter 3, we basically achieve the same results of [Lian and Murray, 2003] in relation to completeness and stability. Our approach is totally decentralized. Therefore, because no communication is necessary, our system is able to guarantee connectivity even in the presence of communication faults such as delays, interference, etc. Also, in contrast

to [Pimentel, 2002] that requires $O(n^2)$ communication, where n is the number of robots, in our approach the communication channels can be totally dedicated to higher level applications.

5.2 Sensing and Communication Networks

In this chapter, robots relationships are defined by sensing and communication networks. These networks can be represented by two graphs, G_{sn} and G_{cn}, where the robots themselves are the vertices of both graphs and the flow of information between the vertices are represented by the edges. Notice that since sensing and communication devices have different characteristics, the graphs G_{sn} and G_{cn} will have different edge sets. Each graph is represented by a triple $(\mathcal{R}, \mathcal{E}, \mathcal{G})$, where the edge set represents communication (or sensing) links among robots. Because we consider identical robots and omnidirectional devices, we can restrict our attention to undirected graphs. Then, $(R_i, R_j) \in \mathcal{E}$ is equivalent to $(R_j, R_i) \in \mathcal{E}$ and $g(q_i, q_j) \in \mathcal{G}$ is identical to $g(q_j, q_i) \in \mathcal{G}$. Also, there are no edges of the form (R_i, R_i), which physically would make no sense.

For most communication algorithms, vertices in a communication graph may be seen as routers that provide communication between two other vertices even if they are not neighbors. The same kind of behavior can be observed in several sensing algorithms, such as the one presented in Apendix B. Then, we now use the concept of *Hamiltonian Graph* (HG) defined in Chapter 3 as a connected subgraph of a graph G where each vertex has up to two neighbors. Routing algorithms can take advantage of the existence of HGs in order to create channels of communication between any two vertices of the graph, since the existence of a path is guaranteed.

We are interested in moving the robots in such a way they complete their tasks and also maintain communication with their teammates. Thus, based on the definition of HG it is easy to verify that we can guarantee communication in a group of n robots if we constrain the robots movements so that there is always at least one HG of n vertices. Furthermore, it is also desirable that other edges are maintained, not only to increase system robustness to agents failures, but also to avoid communication problems such as traffic, delays, routing, etc. In sensing algorithms, edges in the graph can be thought of as constraints for optimization algorithms. Even though only some of them are necessary for algorithm execution, the more edges are available the more precise will be the optimization, as discussed in Apendix B. Thus, as in Chapter 4, we call the edges of the HG by *essential links* and all the remaining edges in $\mathcal{E}$ by *desirable links*.

5.2.1 Formation Constraints

As discussed earlier, we assume that the graphs G_{cn} and G_{sn}, and therefore $\mathcal{E}$, are specified by a higher-level planner. To each edge $(R_i, R_k) \in \mathcal{E}$, we associate a formation constraint for R_i induced by R_k as a inequality of the form $g(q_i, q_k) \leq 0$, where $g(q_i, q_k) \in \mathcal{G}$. In general $g(q_i, q_k)$ could be any convex, differentiable function, but under the assumptions of omnidirectional devices and circular robots, all elements of $\mathcal{G}$ are represented by circles. Then, $g(q_i, q_k) = (x_i - x_k)^2 + (y_i - y_k)^2 - r_k^2$. A constraint is said to be active when $g(q_i, q_k) = \delta_x$, where δ_x is a negative number that can be thought of as a threshold. The constant δ_x defines the radius of the circular constraint.

We consider that a generic robot R_k induces three constraints in R_i. First, we have a hard sensing or communication constraint given by $g(q_i, q_k) \leq \delta_1$, beyond which connectivity between R_k and R_i is lost. While δ_1 could be set

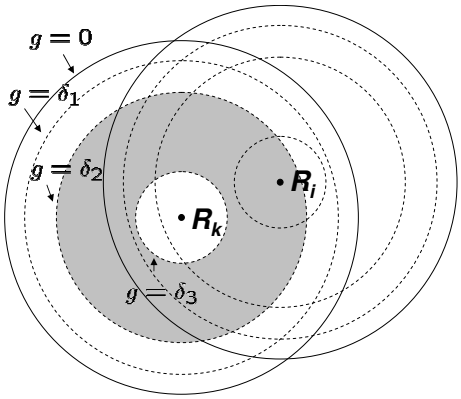

Figure 5.1: Formation constraints: R_k induces constraints on the position of R_i. If R_i is inside the circle defined by $g \leq \delta_1$ (outer dashed circle), connectivity with R_k is guaranteed. The gray area defined by $g > \delta_3$ and $g < \delta_2$ is a *safe* configuration space for R_i, where collisions are avoided and connectivity is maintained.

to zero, it is best, in order to be robust to sensing errors, to keep it at a small, negative value. Second, we have a soft sensing or communication constraint given by $g(q_i, q_k) \leq \delta_2$. This is assumed to delineate a range within which the performance of the communication or sensing link is optimal. Finally, we have the avoidance constraint $g(q_i, q_k) \geq \delta_3$, which prevents a robot to be very close to another[1]. Observe that $\delta_3 < \delta_2 < \delta_1 < 0$. Figure 5.1 shows a picture of R_i constraints induced by R_k. Each neighbor of R_i induces a similar set of constraints.

5.3 Planning and Control

In order to control the robots, the methodology in Chapter 4 can be directly applied. The task planning will consist of two steps (i) – navigation function construction, and (ii) – neighborhood or networking assignment.

[1]For non-point robots, the robot shape itself can be considered a forth constraint, which is a collision constraint.

The construction of the navigation function is performed independently by each robot based on maps of the environment. However, if we want all robots in the group to reach their goals while keeping the constraints, the relationship in Equation (4.11) should be satisfied. This indicates that at least the decision concerning goal positions must be performed in a centralized fashion.

The problem of neighborhood assignment is also a difficult task to be executed in a decentralized manner, since global knowledge about the robots position is necessary. We propose a heuristic that should work for most of cases. This heuristic is based on two steps (i) selecting the Hamiltonian graph and (ii) selecting the desirable links.

HG Selection

Selecting the HG is an operation that involves global knowledge of the group position. Hence, more resources, such as communication bandwidth, processing time and power need to be used. Because of its cost, the algorithm should be used only once, or only when it is necessary during the execution of the task (*e.g.*: only when one of the robots dies or one new robot is incorporated).

In the first step of the algorithm the robots discover the graph topology by "flooding" the network with packages and listening to the answers. Each robot broadcast special packages to its neighbors, which mark the packages with their identifications (IDs) and retransmit it. Eventually, each package will arrive back at its origin containing the information associated with the robots that retransmit it. By examining several packages each robot can estimate the network topology. This is a traditional technique to determine network topologies and may be substituted by more efficient

ones [Lowekamp et al., 2000]. Based on the recovered topology, each robot determines n (or what they think it is) and the IDs of its teammates. One of the robots is then chosen to execute the computation, in case of centralized algorithms, or to coordinate the algorithms, in case of parallel ones. Leader election algorithms, such as the one presented in [Das et al., 2002a], may be used in this process.

Exact algorithms for finding a Hamiltonian Cycle are NP-complete. However, there exist heuristics and parallel algorithms which may be used to improve the computation [MacKenzie and Stout, 1993]. The only necessary condition for these algorithms is the existence of at least one HC. In this case, if no solution is found within a fixed amount time, it is assumed that there is no HC in the graph and the robots will need to move in order to create more connections. Because Hamiltonian Cycles are Hamiltonian Paths with cycles, the problem of finding non-cyclic Hamiltonian Paths can be solved by including "fake" links between vertices with valency 1 (in order to create loops in the graph) and using the same algorithm to solve the problem.

Desirable Links Selection

An easy and obvious way to determine which are desirable links is considering all existing links as such. In this way the robots try to maintain only the current links and, consequently, as soon as a link is broken, it is no longer considered to be desirable. Another simple solution is considering that only the links with the m closest robots are important. In this case, because the gradient of the desirable link constraints are summed together in (4.7), the smaller the value of m, the larger the probability of maintaining the links.

A more complex way of selecting links is based on routing and communication algorithms. In this case desirable links are those which makes

communication simpler and more efficient. In certain tasks, for example, it might be necessary to increase the bandwidth between two specific nodes and, therefore, some specific links could be considered as desirable.

5.4 Experimental Results

In order to validate our methodology we performed experiments with two multi-robot teams: (i) a team with five non-holonomic, car-like robots, equipped with omnidirectional cameras; and (ii) a team of four holonomic robots. Details of both teams are described in Apendix A. Videos of the experiments can be found in [Pereira, 2003].

5.4.1 Non-holonomic robots

Each of the non-holonomic robots in the team use its omnidirectional vision system in order to perform relative localization. Absolute localization in the workspace is obtained by fusing the information from the omnidirectional cameras with data from an overhead camera. These sensor fusion algorithm is described in Apendix B. A limitation of the omnidirectional cameras used by the robots is that their resolution decrease with the distance of the objects. At 2 m, for instance, the projection of an observed robot in the image plane is only one pixel in size. Due to this shortcoming, the robots must keep sensing constraints with respect to their neighbors in order to correctly localize themselves with respect to each other.

Figure 5.3 shows six snapshots of our first experiment. In this experiment G_{sn} was specified such that R_1 and R_3 are neighbors of R_2 but not neighbors of each other as shown in Figure 5.2. The equipotential contours of the navigation function for R_3 is shown in all snapshots. Figure 5.3(a) shows

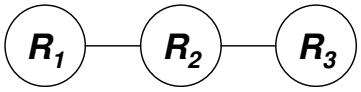

Figure 5.2: Sensing graph for the experiment in Figure 5.3.

that R_1 was initialized outside the sensing region of R_2, which was set to be 1.5 m, and hence in an unsafe configuration space. Observe that this region is smaller than the one where the robots cannot actually detect the others (a circle of approximately 2 m), and therefore immediately outside its critical region of the configuration space they still can perform localization. The next snapshot shows that the robots move to satisfy this constraint. Figure 5.3(c) shows R_2 and R_3 very close two each other. The activation of the avoidance constraints is then followed by a repulsion (Figure 5.3(d)).

In another experiment we consider the deployment of a sensing network of five robots. Figure 5.5 shows four snapshots of the experiment where, all robots follow the same navigation function. This figure illustrates how a local minimum introduced by the ACHIEVE mode of the controller may be used to deploy the network. The robots have to maintain communication constraints with two other neighbors forming a chain. The sensing graph is then a Hamiltonian Path where one of the vertices is a static base as shown in Figure 5.4. In our case the first robot is static and plays the role of base. Consequently, only one of the robots reach the goal but, as a side effect, a communication link is built between the base and the goal.

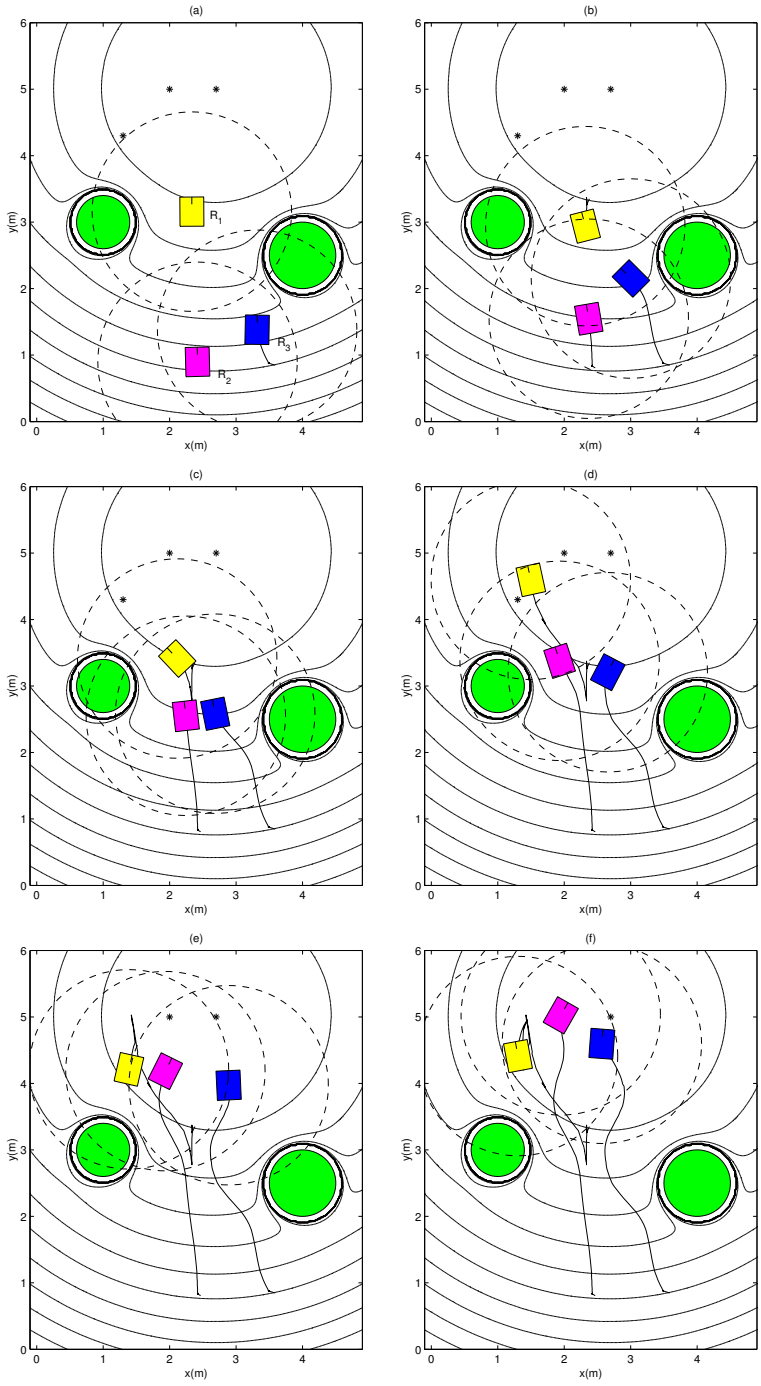

Figure 5.3: Three robots following their navigation functions while maintain sensing constraints with at least another robot. Ground truth data (trailing solid lines behind each robot) is overlaid on the equipotential contours of the navigation function for R_3.

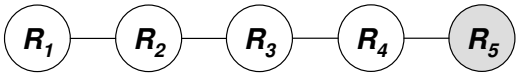

Figure 5.4: Sensing graph for the experiment in Figure 5.5. The gray vertex is a static robot.

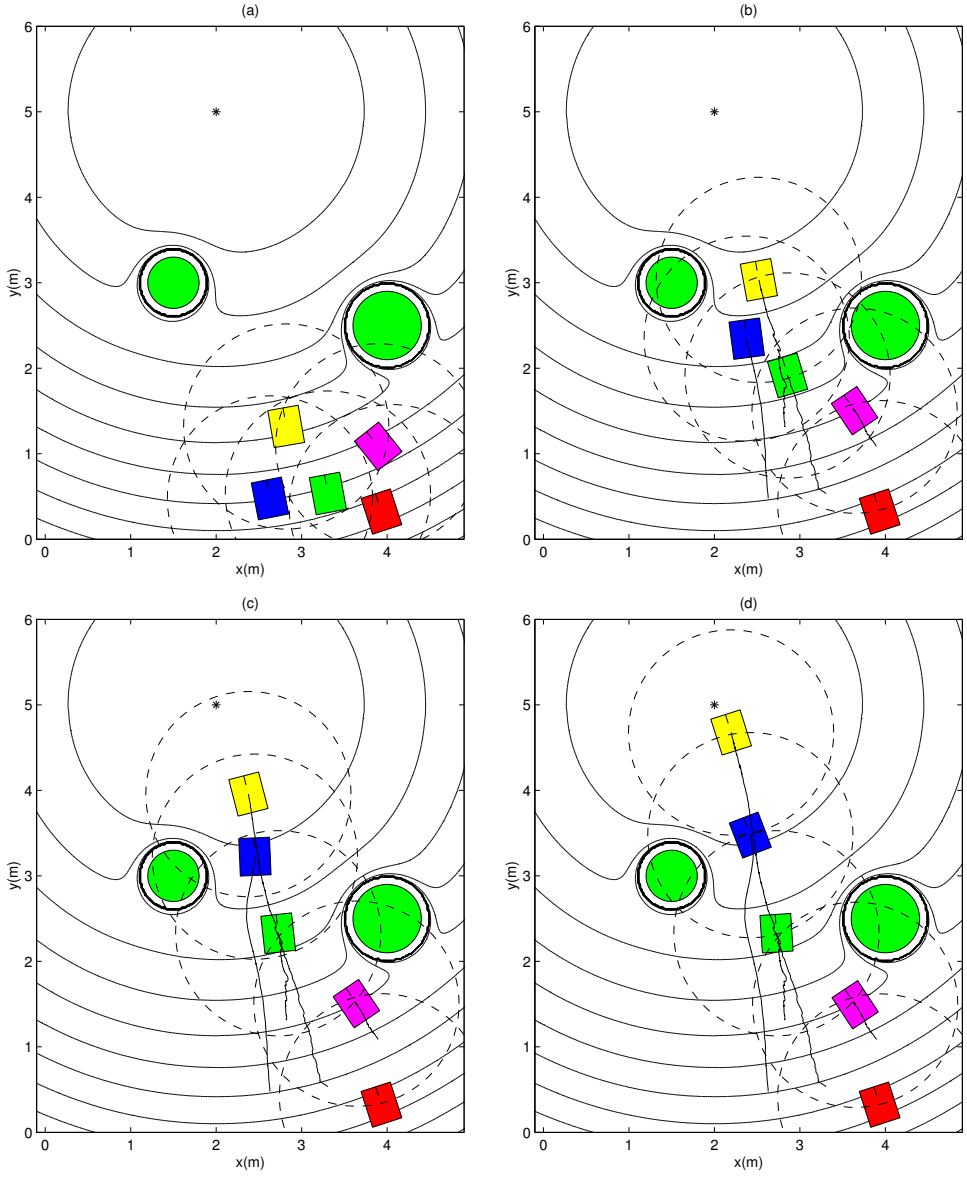

Figure 5.5: Deploying a mobile sensor network with five nodes. Figures (a)–(d) show four snapshots of the same experiment. One of the robots is static and is considered a base. Ground truth data is overlaid on the equipotential contours of the navigation function for the robots.

5.4.2 Holonomic robots

The second set of experiments presented in this chapter were performed with a team of holonomic robots. These robots do not have local sensors or communication devices (see Apendix A) and are controlled remotely by a computer that relies on an overhead camera to localize the robots in the environment. However, a task that involves the same constraints of sensing and communication is *flocking* [Reynolds, 1987]. In this task a group of robots is suppose to move from a initial region of the free configuration space to another region. In order to move as a group the robots must keep a maximum distance from their neighbors and also avoid possible collisions with static obstacles and with other robots. Figure 5.6 shows trajectories of three holonomic robots flocking from a initial position at time equals to t_0, to a final goal at t_f in an environment with a single circular obstacle. As proposed in our methodology, a navigation function was independently constructed for each robot. Because the navigation function defines paths from every free configuration to the desirable configuration, observe in Figure 5.6 that at t_0, R_2 could move towards the goal taking either side of the obstacle. However, due to formation constraints the robots travel together to the goal area.

For the experiment shown in Figure 5.6, graph G was set to be a complete triangular graph (see Figure 5.7) with two constraints per edge: (i) an avoidance constraint represented by a circle with 24 cm of diameter and (ii) a maximum distance constraint represented by a circle with 45 cm of diameter. Inside the first and outside the second circle, which are centered in each of its neighbors, each robot is in the unsafe region of its configuration space and consequently in the ACHIEVE mode of the controller.

In order to evaluate the effect of possible differences in gradients of nav-

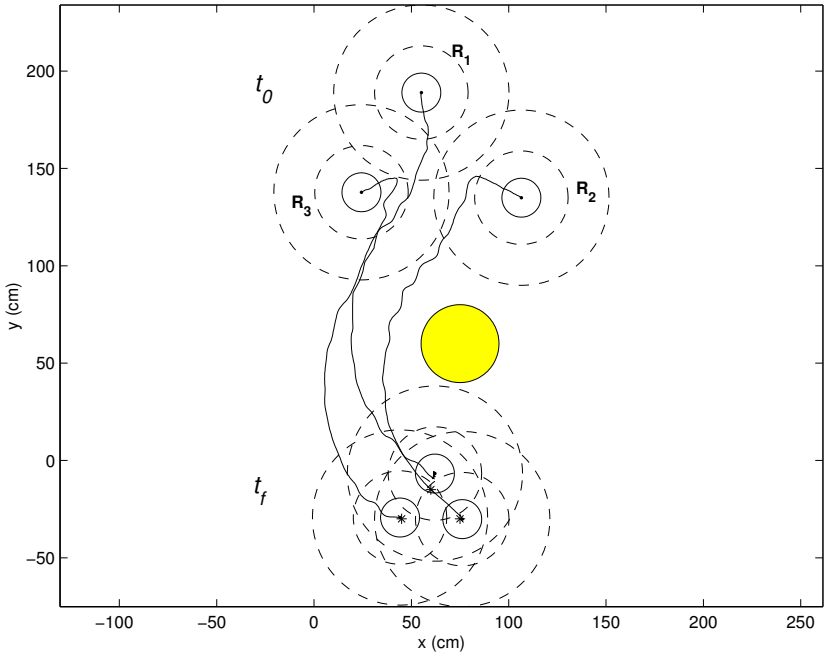

Figure 5.6: Three holonomic circular robots flocking from initial configurations to a target region avoiding a single circular obstacle. The dashed lines represent both the collision and the maximum distance constraints.

igation functions for each robot (what is contrary to Assumption 4.1), and also the effect of different values of the threshold δ, we have executed several trials and obtained quantitative results. We have chosen four sets of values for δ. In the first set, the robots enter in the critical region of the configuration space (and consequently switch to the MAINTAIN mode of the controller) when they are outside a circle with diameter of 37 cm or inside a circle with diameter of 33 cm, centered in their neighbors. In the second set of values these circles were set to have 40 cm and 30 cm of diameter respectively, and in the third set they have diameters 42 cm and 27 cm. In the forth set, the critical region of the configuration space was made very small with circles diameters set be 44 cm and 25 cm respectively. The differences among these four sets of values can be visualized in Figure 5.8, where the regions of the

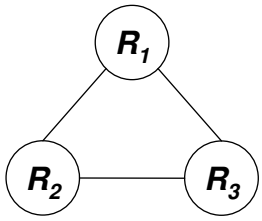

Figure 5.7: Constraint graph for the experiment in Figure 5.6.

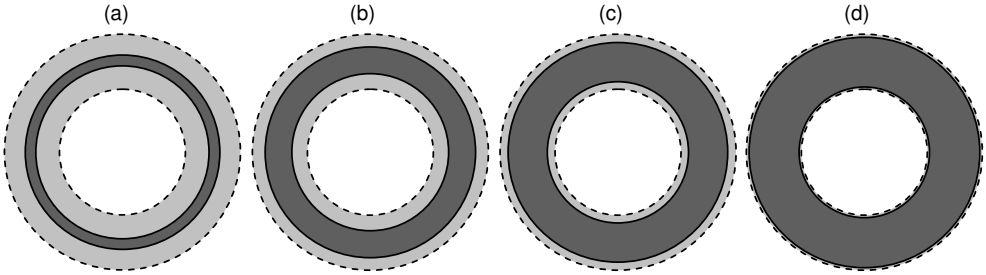

Figure 5.8: The four different sets of values for δ used in the experiments. The dashed lines delimit valid configuration space represented by $g(q_i, q_j) < 0$, and the continuous lines represent $g(q_i, q_j) = \delta$. The shadowed dark areas represent the safe regions of the configuration space and the shadowed light regions are the critical configuration spaces. (a) – Set 1, (b) – Set 2, (c) – Set 3, and (d) – Set 4.

configuration space were plotted to scale.

For each value of δ we have performed 20 runs in a single obstacle environment (Figure 5.6). In all runs the robots were initialized in the same region of the workspace but not in the same exact configuration (due to positioning errors). After entering for the first time in the critical or safe regions of their respective configuration spaces (and therefore satisfying the constraints), the percentage of time each robot spent in each region was computed and is shown in Table 5.1. An average of these values for each experiment is plotted in Figure 5.9. Despite the time the three robots stayed in their unsafe spaces, in all 80 runs (20 for each set) the robots successfully completed their tasks and reached their goals.

Table 5.1: Percentage of time in each region of the configuration space for four different sets of values for δ: Set 1 – big δ, large critical regions; Sets 2 an 3 – intermediate values of δ; Set 4 – small δ, small critical regions.

	Configuration Space	R_1	R_2	R_3
Set 1	Safe (%)	21.16	19.71	16.08
	Critical (%)	77.24	79.80	83.05
	Unsafe (%)	1.60	0.49	0.87
Set 2	Safe (%)	63.77	67.66	60.94
	Critical (%)	33.76	31.78	36.81
	Unsafe (%)	2.47	0.56	2.25
Set 3	Safe (%)	78.10	81.08	77.62
	Critical (%)	20.30	18.39	21.08
	Unsafe (%)	1.60	0.53	1.30
Set 4	Safe (%)	76.81	77.17	78.17
	Critical (%)	10.22	10.56	10.91
	Unsafe (%)	12.98	12.27	10.92

The first observation from Table 5.1 is that Assumption 4.1 is either a good approximation or it is actually not necessary when the robots have enough time to respond to an active constraint. Observe that, except for set 4 where the critical region is very small, the control laws in the MANTAIN mode (relative to the critical region of the configuration space) are sufficient to prevent the system to move to its unsafe space. This conclusion come from the small percentage of time that the robots stay in their unsafe regions of the configuration space.

The results also show that robots behaviors change when the relative sizes of their configuration space regions vary. When the critical region is relatively large (set 1), the robots stay a large percentage of time in the MAINTAIN mode. Apparently it is not a big problem since we have proved that in this mode the robots are still following their navigation functions. However, the time of completion of the whole mission can be compromised, since they are not moving in the best direction, given by the gradient of

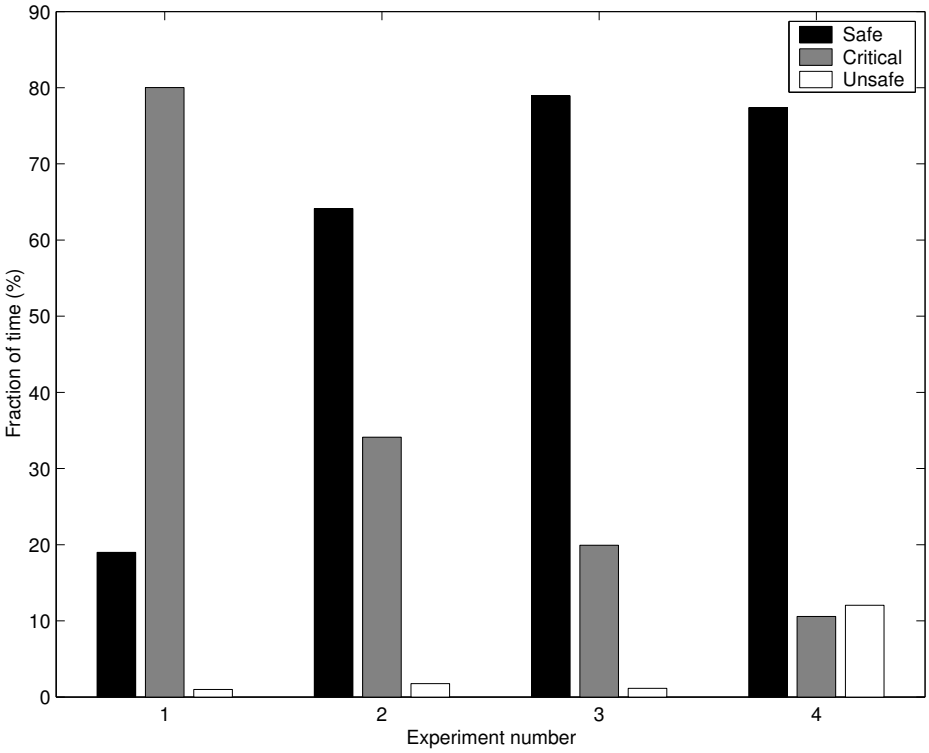

Figure 5.9: Effect of δ in the mean time the robots spend in each region of their configuration spaces. The value of δ increases from experiment 1 to 4.

the navigation function. This can be observed in Table 5.2 where the mean completion time for each set of experiments is shown. Observe, by comparing experiment sets 1, 2 and 3 that smaller is the time the robots stay in their MAINTAIN mode, smaller is the time of completion. By Table 5.2 one can also notice that time of completion is very compromised when the robots enter in their unsafe configuration spaces, as is the case in the forth set of experiments. It happens because, in the unsafe space, the robots completely forget the navigation function they are following in order to rapidly satisfy the constraints. By acting like this, the robots may even move in a direction contrary to the gradient of the navigation function.

Results in Figure 5.10 are equivalent to the ones in Figure 5.9, but now we

Table 5.2: Time of completion for each set of experiments.

	Set 1	Set 2	Set 3	Set 4
Mean time (s)	38.67	34.90	34.60	41.64
Standard deviation (s)	2.46	1.55	1.74	5.74

have independently plotted the percentage of time each of the two constraints was active. Then, in Figure 5.10, the term "close" is relative to the avoidance constraint and the term "far" is relative to the maximum distance constraint. "Unsafe-close", for example, is relative to collisions among the robots. One can observe in Figure 5.10 that the avoidance constraint is more likely to be violated than the maximum distance one. This is because environment and goal positions favor situations where the robots cluster together rather than situations where they are far from each other. Apart from that, it turns out to be very difficult to compare the efficiency of the controller relative to the nature of the constraints. An important result would be to compare the effect of the convexity of the constraint. In this case de avoidance constraint is concave and the maximum distance constraint is convex. With the present results it seems that the controller presents almost the same efficiency for both constraints. The study of situations where the constraints are equally activated is left as future work.

Finally, by Figures 5.9 and 5.10, it can be observed that, among the δs tested, set 3 present the best results, since it yields the smallest percentage of time in the unsafe space and the largest amount of time in the safe space. Consequently, this situation is the one that presents the smallest time of completion (Table 5.2). Notice that safe regions that are smaller or bigger than the one in set 3 (Figure 5.8) present poorest performance. This is probably an indication that convex optimization can be used in real time to set the best value for δ. This is left as future work.

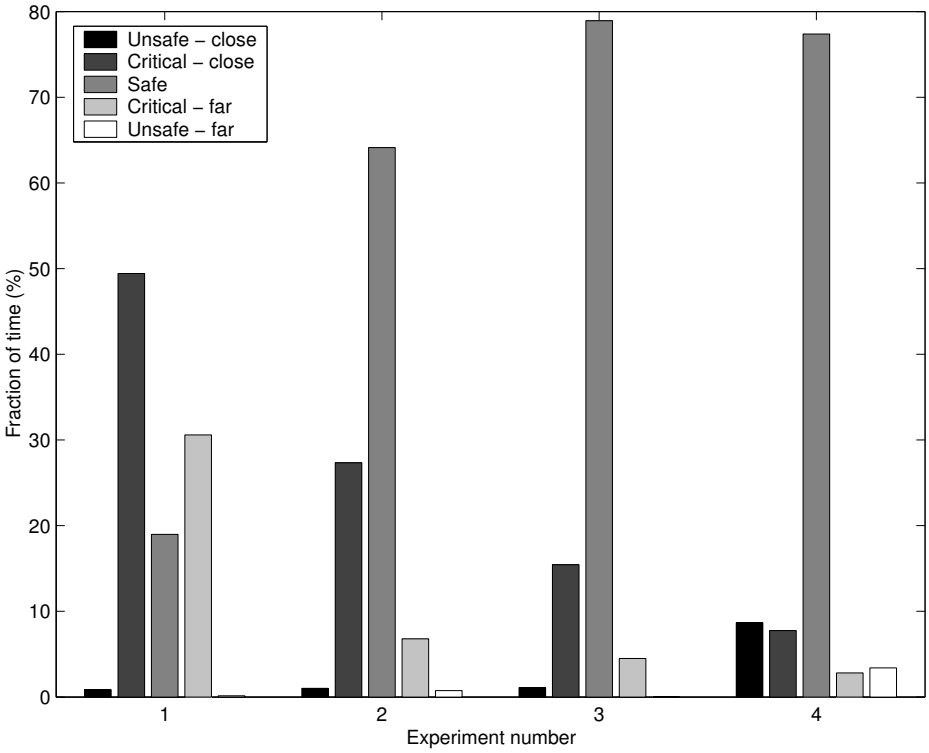

Figure 5.10: This figure shows the same results in Figure 5.9 but now critical and unsafe spaces were divided in two groups: (i) close – relative to the concave, avoidance constraint, and (ii) far – relative to the convex, maximum distance constraint.

5.5 Concluding Remarks

This chapter presented a very direct example of our general framework for motion planning and control of teams of cooperating robots. We have theoretically shown that the approach guarantees sensing and communication constraints. Experimentally we showed that the control laws are efficient to recover possible constraint violations. Our control algorithms are totally decentralized and do not require communication among the robots. Therefore they are robust to communication failures. Other advantages include simplicity and low computational requirements.

We can also point some limitations of the approach. Our communication

(sensing) model is binary, meaning that within a radius, the communication is perfect and null otherwise. This model is very convenient and adequate for the approach, but more sophisticated models could be used in order to define optimal trajectories for the robots. Also, we assume that the constraints preserve their shapes in the presence of obstacles. It is not always true since some obstacles may block communication. For relative large and modelled obstacles this is not a problem, but in some situations (when the obstacle appears in between two robots, for example) the communication among the robots may be broken, even if the constraints are satisfied.

An important direction of future work is related to neighborhood assignment. In order to extend the methodology for large groups of robots, decentralized algorithms for planning the graphs G_{cn} and G_{sn} should be studied. Research in ad-hoc networking has pointed solutions to this problem but, in most of them, there is no specification for fixed neighbors as we require in our approach. Instead, a minimum number of robots is specified [Xue and Kumar, 2003]. In order to adopt this methodology we should be able to include time varying graphs. However, this option should be better studied since some of our current proofs would fail if such kind of graph were considered.

Chapter 6

Application to Manipulation

This chapter presents a second application to the methodology presented in Chapter 4. In this case more complex constraints are imposed by a novel manipulation technique that has been called *object closure*. Section 6.1 compares the object closure methodology with similar manipulation techniques encountered in the literature. Object closure is detailed in Section 6.2 and a discussion on how to use the control laws of Chapter 4 for manipulation is addressed in Section 6.3. Experiments are presented in Section 6.4 and concluding remarks are made in Section 6.5.

6.1 Introduction

The problem of controlling multiple mobile robots in order to transport objects by caging was introduced in [Sudsang and Ponce, 1998], where a centralized algorithm was developed in order to coordinate a group of identical circular holonomic robots. In synthesis, the caging problem consists in constructing and guaranteeing a compact and closed set of configurations for the object being manipulated. Therefore, two basic solutions to the problem can be found in the literature. In the first and more direct solution the set of configurations is constrained to be in the set $SO(2) \times [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ where $x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max} \in \mathbb{R}^2$ are limits to the object translation.

Practically, this approach is implemented by surrounding the object by a group of agents and guaranteeing that the distance between two consecutive robots is smaller than the smallest length of the object. In this situation the object is able to rotate freely among the agents but cannot be removed. The computations needed can be decentralized since each robot only needs to control its distance to its nearest neighbors. An example of such a strategy, obtained by using potential field controllers and holonomic robots is shown in [Song and Kumar, 2002] and [Chaimowicz et al., 2001a].

In the second solution to the caging problem the set of object configurations are constrained to $[\theta_{\min}, \theta_{\max}] \times [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. In such an approach, besides translation, the agents also constrain the rotation of the object. Although this solution be inherently more difficult to implement than the previous one, its advantages include smaller number of agents needed and precision in the manipulation. The big drawbacks are the higher computational cost and the need for centralized algorithms, since the relative positioning of all agents in the group (not only the nearest neighbors) is necessary. Furthermore, the robots movements are highly dependable on the object actual configuration, implying in a low degree of robustness. This strategy was successful applied in [Sudsang and Ponce, 2000, Wang and Kumar, 2002b].

Object closure inherits the simplicity, distributively and robustness of the first approach, but also includes some advantages of the second one, as the reduced number of agents necessary. We assume dynamical restrictions for the object in other to perform an object closure test. Hence, considering a maximum object's angular velocity, $\omega_{\max}$, we determine a set of possible orientations for the object and test for closure at this orientations. Thus, we guarantee object closure in the set of configurations $[\theta_{\mathrm{o}} - \Delta\theta_{\max}, \theta_{\mathrm{o}} + \Delta\theta_{\max}] \times [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ where θ_o is the actual object orientation

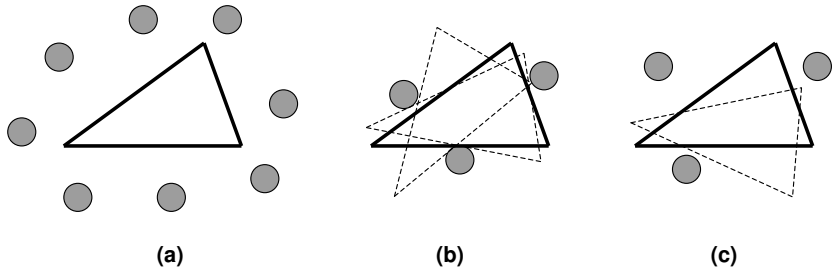

Figure 6.1: Three approaches for caging. (a) – the object is able to rotate and, independently of its orientation, it cannot be removed from the robot formation; (b)– the object's rotation is restricted. Caging is achieved for all possible orientations. The dashed objects represent the maximum and minimal orientations; (c) – *Object closure*, the object is able to rotate. Closure is only guaranteed for a small set of orientations. The dashed object represents an example where the object can scape with a series of rotations and translations. Our approach is based on the fact that the object cannot execute movements like this one in small periods of time.

and $\Delta\theta_{\max} = \Delta T\ \omega_{\max}$ is the maximum angular displacement of the object until the next time step. Observe that the agents do not restrict the object rotation but only assume bounds to this movement. Figure 6.1 shows the three caging approaches for a triangular object.

Using the methodology presented in Chapter 4, our basic goal is to develop decentralized control policies for a group of robots to achieve a condition of object closure, and then, move toward a goal position while maintaining this condition of object closure. Unlike previous work [Sudsang and Ponce, 2000, Wang and Kumar, 2002a], we do not require the robots to be circular. However, we do introduce a number of simplifying assumptions to enable real-time implementation. Further, our interest is in transporting the object from an initial position toward a goal position in $\mathbb{R}^2$. We do not address the problem of precisely positioning and orienting the object in the plane.

6.2 Object Closure

6.2.1 Definition

Consider a planar world, $\mathcal{W} = \mathbb{R}^2$, occupied by a convex, polygonal object $\mathcal{O}$, and a group of n convex, polygonal robots. The i^{th} robot R_i is described by the convex set $\mathcal{A}_i(q_i) \in \mathcal{W}$, where $q_i = (x_i, y_i, \theta_i)$ denotes the configuration of R_i. The configuration of the object is described by the coordinates $q = (x, y, \theta)$. We will use $\mathcal{C}_{\mathcal{R}_i}$ to denote the configuration space of the i^{th} robot, while $\mathcal{C}$ will denote the configuration space for the object $\mathcal{O}$.

If robot positions and orientations are held fixed, the region in the configuration space that corresponds to an interpenetration between the object $\mathcal{O}$ and the robot i is:

$$\mathcal{C}_{obj\_i} = \{q \in \mathcal{C} \mid \text{interior}(\mathcal{A}_i(q_i) \cap \mathcal{O}(q) \neq \emptyset)\}, \tag{6.1}$$

where $\mathcal{O}(q)$ is the representation of $\mathcal{O}$ in the configuration q.

In order to make it easier to explain the basic ideas of object closure, we will reconsider the three first assumptions of Chapter 4 and add one more. These assumptions will be relaxed in the next sections.

Assumption 6.1 The manipulated object cannot rotate — the coordinates of the object are given by $q = (x, y)$ and $\mathcal{C} \subset \mathbb{R}^2$.

Figure 6.2 shows the boundary of $\mathcal{C}_{obj\_i}$ for a five-sided polygonal object and the point robot, R_i.

The union of $\mathcal{C}_{obj\_i}$ for $1 \leq i \leq n$ determines the region in $\mathcal{C}$ which cannot

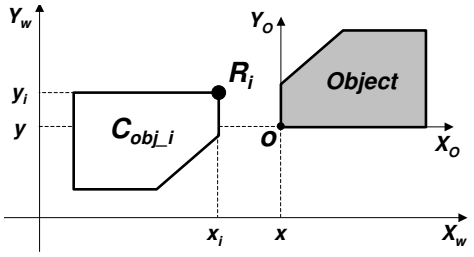

Figure 6.2: $\mathcal{C}_{obj\_i}$ for a point robot considering only object translations. By sliding the object around the robot, the origin, o, of the object-fixed reference frame traces out the boundaries of $\mathcal{C}_{obj\_i}$.

be occupied by the object. Then,

$$\mathcal{C}_{obj} = \bigcup_{i=1}^{n} \mathcal{C}_{obj\_i} \,. \tag{6.2}$$

Let the complement of $\mathcal{C}_{obj}$ in $\mathcal{C}$ be $\bar{\mathcal{C}}_{obj}$. When $\bar{\mathcal{C}}_{obj}$ consists of two (or more) disjoint sets, we use the term *object closure* to refer to the condition when one of these sets is compact and contains the object configuration, q. This is shown for four robots in Figure 6.3, where the compact set, which we refer to as the *closure configuration space* and denote by $\mathcal{C}_{cls}$, is shown shaded. Observe that the object is trapped or caged (in the terminology of [Rimon and Blake, 1996]) when its origin is in $\mathcal{C}_{cls}$.

We can easily relax Assumption 6.1 to accommodate the more general case with translations and rotations. In this case, Equation (6.2) remains the same, but $\mathcal{C}_{obj\_i}$ in Equation (6.1) is a three-dimensional solid whose cross-section (slice), for a given angular orientation, is similar to the picture in Figure 6.2, and the compact subset $\mathcal{C}_{cls}$ consists of one or more three dimensional solids whose cross-section is similar to the one shown in Figure 6.3 [Wang and Kumar, 2002b].

We now define a *non-essential robot* with the help of Figure 6.4. In

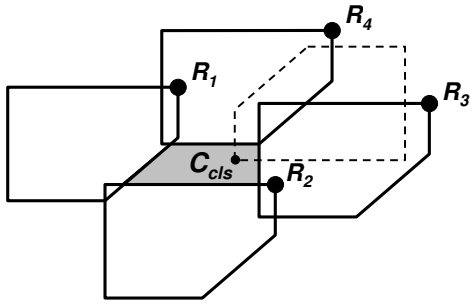

Figure 6.3: Object closure: the interior (shaded gray) represents the *closure configuration space*, $\mathcal{C}_{cls}$, for a team of 4 robots. The dashed polygon represents the object. Notice that the origin of the object's reference frame is inside $\mathcal{C}_{cls}$, a compact set, indicating a condition of object closure.

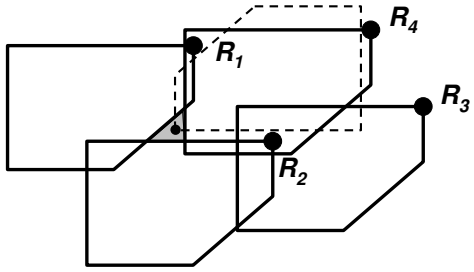

Figure 6.4: Essential Robots: even with the removal of R_3 the closure properties of the group are preserved and so, R_3 is a non-essential robot.

contrast to Figure 6.3 in which all four $\mathcal{C}_{obj\_i}$ (and therefore all four robots) are essential to construct the boundary for the closure configuration space, R_3 is not essential for object closure in Figure 6.4. In a group of robots maintaining object closure, a non-essential robot, R_x, is a robot whose removal (and consequently the absence of the constraint due to $\mathcal{C}_{obj\_x}$) does not violate the state of object closure.

We now introduce a fifth assumption:

Assumption 6.2 There are initially no non-essential robots in the group.

6.2.2 A Test for Object Closure

We are interested in decentralized control policies and in strategies that do not depend on communication links for sharing information. Secondly, we are interested in making the manipulation task robust to errors position and orientation estimation of the object. In what follows, we develop a decentralized test for object closure that does not require precise estimates of the position and orientation of the object. Checking the object closure condition involves two steps: (i) establishing the existence of $\mathcal{C}_{cls}$; and (ii) verifying $q \in \mathcal{C}_{cls}$. Step (i) requires obtaining state information from all robots and Step (ii) requires obtaining position (pose, in the more general case) of the object.

The key idea comes from Figure 6.3 where robots are numbered R_1 through R_n in a counterclockwise fashion. A necessary condition for object closure with no non-essential robots is that the i^{th} robot's position satisfies: $\mathcal{C}_{obj\_i-1} \cap \mathcal{C}_{obj\_i} \neq \emptyset$; and $\mathcal{C}_{obj\_i} \cap \mathcal{C}_{obj\_i+1} \neq \emptyset$. This condition is not sufficient. The sufficient condition involves verifying $\mathcal{C}_{cls} \neq 0$ and $q \in \mathcal{C}_{cls}$. However, this condition is necessary and sufficient for maintaining object closure once a condition of object closure is achieved. Hence we can state the following:

Proposition 6.1 ([Pereira et al., 2003c]) *If an object is in a state of object closure with a group of robots with no non-essential robots, a sufficient condition for maintaining object closure is* $\mathcal{C}_{obj\_i-1} \cap \mathcal{C}_{obj\_i} \neq \emptyset$*; and* $\mathcal{C}_{obj\_i} \cap \mathcal{C}_{obj\_i+1} \neq \emptyset$.

We now explain how to derive the algebraic equations for object closure. Define $\mathcal{I}_i$ to be a set of R_k's configuration space that represents the intersection between $\mathcal{C}_{obj\_i}$ and $\mathcal{C}_{obj\_k}$:

$$\mathcal{I}_i = \{q_k \in \mathcal{C}_{R_k} \mid \mathcal{C}_{obj\_i}(q_i) \cap \mathcal{C}_{obj\_k}(q_k) \neq \emptyset\}\,.$$

Note that $\mathcal{C}_{obj\_i}(q_i)$ and $\mathcal{C}_{obj\_k}(q_k)$ are identical polygons, which introduces a symmetry in the form of $\mathcal{I}_i$.

It can be observed that:

$$\mathcal{C}_{obj\_i} \cap \mathcal{C}_{obj\_k} \neq \emptyset \Leftrightarrow (q_k \in \mathcal{I}_i \wedge q_i \in \mathcal{I}_k)\,.$$

Thus, the object closure conditions for each robot, which can be rewritten as $q_i \in \mathcal{I}_{i-1}$ and $q_i \in \mathcal{I}_{i+1}$ (see Figure 6.5(a)) are represented as a set of inequality constraints of the form $g_j(q_{i-1}, q_i) \leq 0$ or $g_j(q_i, q_{i+1}) \leq 0$, where g_j are the functions that delimit $\mathcal{I}_{i-1}$ or $\mathcal{I}_{i+1}$ respectively. $\mathcal{I}_{i-1}$ ($\mathcal{I}_{i+1}$) is a $2\,m$-sided polygon defined by $2\,m$ algebraic constraints, each linear in q_{i-1} and q_i (q_i and q_{i+1}). Since each polygon has up to $2\,m$ sides, the number of constraints for each robot is $4\,m$. For the situation we are considering, where the robots are points and the object cannot rotate, the boundary of $\mathcal{C}_{obj\_i}$ is formed by the same edges of $\mathcal{O}$ but ordered in a different way (see Figure 6.2). Then, each $\mathcal{I}_i$, which depends on $\mathcal{C}_{obj\_i-1}$ and $\mathcal{C}_{obj\_i+1}$, is bounded by two sets of the object's edges (refer to the algorithm presented in [Latombe, 1991] for proofs). Consequently, $\mathcal{I}_{i-1}$ is given by functions $g_j(q_{i-1}, q_i)$, while $\mathcal{I}_{i+1}$ is given by another set of functions $g_j(q_i, q_{i+1})$. Each function is directed derived from the functions $f_i(x, y)$ used to describe the object.

We can now rewrite Proposition 6.1 as follows:

Proposition 6.2 *If an object is in a state of object closure with a group of robots with no non-essential robots, a sufficient condition for maintaining object closure is $q_i \in \Gamma_i$, where $\Gamma_i = \mathcal{I}_{i-1} \cap \mathcal{I}_{i+1}$.*

Notice that Γ_i is bounded by a subset of the constraints $g_j(q_i) \leq 0$, $1 \leq j \leq 4m$. An example of Γ_i can be seen in Figure 6.5(b).

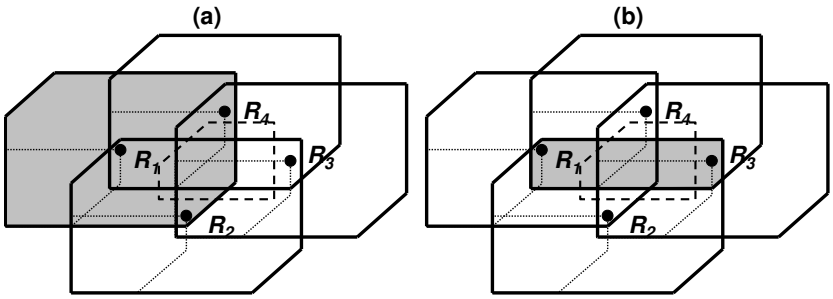

Figure 6.5: Object closure is achieved if each robot i is inside Γ_i. The shaded areas represent (a) $\mathcal{I}_1$ and (b) Γ_1.

6.2.3 Introducing Rotations

Thus far, we have ignored rotations. In reality, since the robots will collide with and bump against the object, the object can rotate. Even if object closure is guaranteed for a given object orientation, a small rotation followed by a translation may cause the object to escape from the robot formation.

Our approach to incorporate rotations is to establish guarantees for object closure under the worst case rotation. Because the object has no actuators, its maximum velocity is limited by the maximum velocity of the robots. Thus, if the object orientation at any instant is estimated to be θ_o, the orientation in the ensuing interval ΔT must be in the interval, $[\theta_{min}, \theta_{max}]$, where $\theta_{min} = \theta_o - \Delta T\, \omega_{max}$, $\theta_{max} = \theta_o + \Delta T\, \omega_{max}$, and ω_{max} is the (estimated) maximum object's angular velocity. Let $\mathcal{J}_i$ be defined as:

$$\mathcal{J}_i = \bigcap_{\theta=\theta_{min}}^{\theta_{max}} \mathcal{I}_i(\theta)\,,$$

where $\mathcal{I}_i(\theta)$ is $\mathcal{I}_i$ computed for an object orientation θ. Following the previous methodology, the conditions that guarantee object closure for all $\theta \in [\theta_{min}, \theta_{max}]$ are: $q_i \in \mathcal{J}_{i-1}$ and $q_i \in \mathcal{J}_{i+1}$.

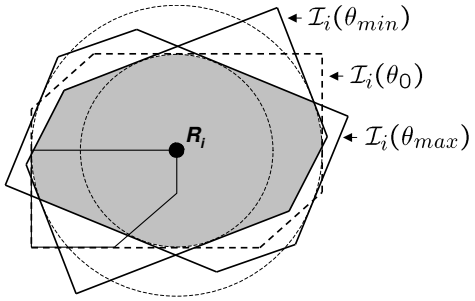

Figure 6.6: Closure region for a maximum rotation of 20°.

Since $\mathcal{C}_{obj\_i}$ is represented by the same polygon for every robot, the shape of $\mathcal{I}_i(\theta)$ is independent of the object orientation. As θ changes, $\mathcal{I}_i(\theta)$ is obtained by simply rotating $\mathcal{I}_i(\theta_o)$ around R_i. The intersection set $\mathcal{J}_i$, can be constructed as shown in Figure 6.6. The shaded area represents the configuration space where q_i must be in order to guarantee object closure for object orientations between θ_{min} and θ_{max}. It is bounded by circular arcs and the sides of $\mathcal{I}_i(\theta_{min})$ and $\mathcal{I}_i(\theta_{max})$. Notice that we continue with a set of inequalities constraints but now two of them are nonlinear. The set $\mathcal{J}_i$ is still convex. From a practical standpoint, this set-valued approach for modeling the uncertainty in orientation allows us to be robust to errors in pose estimation.

6.2.4 Polygonal Robots

The main difficulty of working with polygonal robots is the computation of $\mathcal{C}_{obj\_i}$ in real time. Although an efficient algorithm exists, it must run every time step since changes in robot orientation alters not only the orientation but also the shape of $\mathcal{C}_{obj\_i}$. Furthermore, differences on the shapes of $\mathcal{C}_{obj\_i}$ and its neighbors make the form of closure constraints very complicated. However, since $\mathcal{C}_{obj\_i}$ can be constructed by the union of the $\mathcal{C}_{obj\_i}$ of the points in the convex polygon that form the robot, we can easily establish a

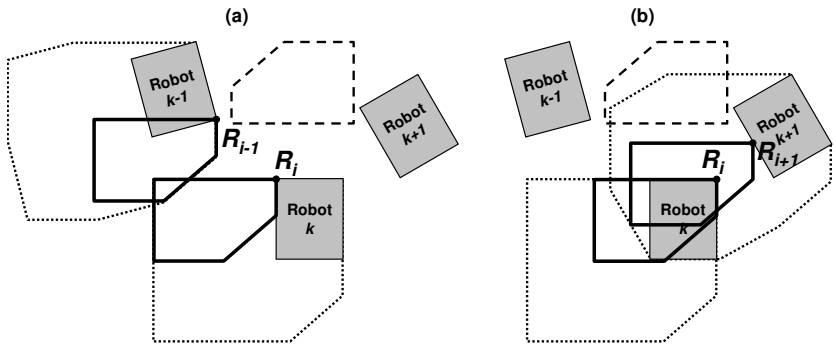

Figure 6.7: Robot k checks closure (a) using the imaginary point robots, R_i and R_{i-1} (left) (b) using a different set of point robots, R_i and R_{i+1} (right). The dotted polygons are the actual object configuration space.

sufficient condition that guarantees closure. If the intersection between $\mathcal{C}_{obj\_i}$ of two virtual point robots located at the closest pair of points between robot k and robot $k+1$ is non zero, then $\mathcal{C}_{obj\_k} \cap \mathcal{C}_{obj\_k+1} \neq \emptyset$. Since $\mathcal{C}_{obj\_i}$ of a point robot can be computed off-line, the online computation is limited to the translation of this set to the location of the virtual point robots. This computation is illustrated in Figure 6.7.

We now use the concept of Minkowski sum presented in Section 3.4. For each slice of the configuration space (a specific orientation), the object configuration space for the polygonal robot k can be written as:

$$\mathcal{C}_{OBJ\_k} = \mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0) \ ,$$

where $\mathcal{C}_{obj\_i}(0)$ is the the object configuration space for the point robot located in the origin of the world reference frame. We can observe also that:

$$\mathcal{C}_{obj\_i}(q) = \{q\} \oplus \mathcal{C}_{obj\_i}(0) \ .$$

Therefore, we state the following proposition:

Proposition 6.3 *If* $\mathcal{C}_{obj\_i}(q_a) \cap \mathcal{C}_{obj\_i}(q_b) \neq \emptyset$, *and* $q_a \in \mathcal{A}_k(q_k)$ *and* $q_b \in \mathcal{A}_l(q_l)$ *are the closest pair of points between the two convex robots, then:*

$$(\mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0)) \cap (\mathcal{A}_l(q_l) \oplus \mathcal{C}_{obj\_i}(0)) \neq \emptyset\,.$$

Proof: Observe that $\{q_a\} \oplus \mathcal{C}_{obj\_i}(0) \subset \mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0)$ and $\{q_b\} \oplus \mathcal{C}_{obj\_i}(0) \subset \mathcal{A}_l(q_l) \oplus \mathcal{C}_{obj\_i}(0)$, and remember that if $x \subset X$ and $y \subset Y$, x, X, y and Y generic sets, and $x \cap y \neq \emptyset$ then $X \cap Y \neq \emptyset$. ∎

Thus, using the closest pair of points as reference for our computations leads us to a conservative but simple test for object closure for polygonal robots.

6.2.5 Circular Robots and Objects

The theory described so far consider point or polygonal robots and polygonal objects. This section discusses the implications of working with circular objects and robots.

Circular Objects

When circular objects are considered we can easily improve the efficiency of the previous methodology. In the case of point robots, for example, because C_{obj_i} is a cylinder in the configuration space (constant for all orientations), the test for object closure is reduced to comparison between the diameter of the object with the distance between the robots.

When we are working with polygonal robots we can change Proposition 6.3 and have an exact solution for testing for object closure (in contrast to a conservative one in the case of polygonal object).

Proposition 6.4 *For a circular object, if $q_a \in \mathcal{A}_k(q_k)$ and $q_b \in \mathcal{A}_l(q_l)$ are the closest pair of points between the two convex robots then:*

$$\mathcal{C}_{obj\_i}(q_a) \cap \mathcal{C}_{obj\_i}(q_b) \neq \emptyset \Leftrightarrow (\mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0)) \cap (\mathcal{A}_l(q_l) \oplus \mathcal{C}_{obj\_i}(0)) \neq \emptyset\,.$$

Proof:

($\Rightarrow$) See proof of Proposition 6.3.

($\Leftarrow$) Suppose $(\mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0)) \cap (\mathcal{A}_l(q_l) \oplus \mathcal{C}_{obj\_i}(0)) \neq \emptyset$. In this case, exist at least one pair of points, $q_1 \in \mathcal{A}_k(q_k)$ and $q_2 \in \mathcal{A}_l(q_l)$, such that $(\{q_1\} \oplus \mathcal{C}_{obj\_i}) \cap (\{q_2\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset$. Because, $\mathcal{C}_{obj\_i}$ is circular $I(q_1) = \{x | (\{q_1\} \oplus \mathcal{C}_{obj\_i}) \cap (\{x\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset\}$ is a circle. Since $(\{q_2\} \oplus \mathcal{C}_{obj\_i})$ intersects $(\{q_1\} \oplus \mathcal{C}_{obj\_i})$, q_1 must be in this circle. Then, $\|q_1 - q_2\| \leq 2r$, where r is the radius of $\mathcal{C}_{obj\_i}$. Suppose now that $q_b \notin I(q_a)$. Thus, $\|q_a - q_b\| > 2r$ and consequently $\|q_1 - q_2\| < \|q_a - q_b\|$. It is a contradiction since q_a and q_b are the closest pair of points between $\mathcal{A}_k(q_k)$ and $\mathcal{A}_l(q_l)$. Hence, if exist q_1 and q_2 such that $(\{q_1\} \oplus \mathcal{C}_{obj\_i}) \cap (\{q_2\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset$ then $(\{q_a\} \oplus \mathcal{C}_{obj\_i}) \cap (\{q_b\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset$. ■

Circular Robots

When all robots in the group are circular they can always be considered as point robots if the object is increased by their sizes. This was the approach used by the previous authors that considered manipulation by caging [Sudsang and Ponce, 2000, Wang and Kumar, 2002b].

If only some of the robots are circular, we can again take advantage of this fact in order to perform an exact test between a polygonal and a circular robot. In this case, we perform two kinds of test: (i) – a test between two circular robots by considering them as points; (ii) – a test between two

polygonal robots using Proposition 6.3; and (iii) – a test between a circular and a polygonal robots using Proposition 6.5.

Proposition 6.5 *If $\mathcal{A}_k$ describes a circular robot and $\mathcal{A}_l$ describes a convex polygonal robot, and $q_a \in \mathcal{A}_k(q_k)$ and $q_b \in \mathcal{A}_l(q_l)$ are the closest pair of points between the two robots, then:*

$$\mathcal{C}_{obj\_i}(q_a) \cap \mathcal{C}_{obj\_i}(q_b) \neq \emptyset \Leftrightarrow (\mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0)) \cap (\mathcal{A}_l(q_l) \oplus \mathcal{C}_{obj\_i}(0)) \neq \emptyset\,.$$

Proof:

($\Rightarrow$) See proof of Proposition 6.3.

($\Leftarrow$) Suppose $(\mathcal{A}_k(q_k) \oplus \mathcal{C}_{obj\_i}(0)) \cap (\mathcal{A}_l(q_l) \oplus \mathcal{C}_{obj\_i}(0)) \neq \emptyset$. In this case, there exists at least one pair of points, $q_1 \in \mathcal{A}_k(q_k)$ and $q_2 \in \mathcal{A}_l(q_l)$, such that $(\{q_1\} \oplus \mathcal{C}_{obj\_i}) \cap (\{q_2\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset$. Because, $\mathcal{A}_k$ is circular $I(q_k) = \{x \in \mathcal{A}_k(q_k) | \{x\} \oplus \mathcal{A}_k)\}$ is a circle. Since $(\{q_1\} \oplus \mathcal{C}_{obj\_i})$ intersects $(\{q_2\} \oplus \mathcal{C}_{obj\_i})$, q_2 must be in this circle. Then, $\|q_1 - q_2\| \leq (r + d)$, where r is the radius of $\mathcal{A}_k$ and d is some constant related to the object size. Suppose now that $q_b \notin I(q_k)$. Thus, $\|q_a - q_b\| > (r + d)$ and consequently $\|q_1 - q_2\| < \|q_a - q_b\|$. It is a contradiction since q_a and q_b are the closest pair of points between $\mathcal{A}_k(q_k)$ and $\mathcal{A}_l(q_l)$. Hence, if exist q_1 and q_2 such that $(\{q_1\} \oplus \mathcal{C}_{obj\_i}) \cap (\{q_2\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset$ then $(\{q_a\} \oplus \mathcal{C}_{obj\_i}) \cap (\{q_b\} \oplus \mathcal{C}_{obj\_i}) \neq \emptyset$. ■

6.3 Planning and Control

So far, we have presented ways to describe a set of manipulation constraints in terms of the object's geometry and orientation. As in Chapter 4, this set of constraints is represented by the boundaries of the configuration

space Γ_i. Notice however that, in most of times, a direct computation of Γ_i is not necessary. Simple tests of intersection between polygons can be used to verify a situation of object closure and identify the active constraints, if there is any. Notice that in this chapter each robot needs to maintain constraints with only two robots, which are its essential neighbors. At first, the robots do not have desirable neighbors, although these could be introduced in order to avoid collisions and jamming due to friction.

In order to control the robots we can basically use the same set of controllers presented in Chapter 4. One important difference, however, is that here we can plan a unique navigation function for the group. The robots would follow the gradient of this navigation function, computed in function of the object position. Therefore, we enforce the proof of Propositions 4.1 in Chapter 4 because we are guaranteing that for two neighboring robots R_i and R_j, $\nabla\Phi_i(q_i) = \nabla\Phi_j(q_j)$ as was assumed in Assumption 4.4. In this case $\nabla\Phi_i(q_i) = \nabla\Phi_j(q_j) = \nabla\Phi(q)$, where $\Phi(q)$ is a navigation function for the ensemble constituted by the robots and the object. Assuming that object's and robot's geometry are known, the navigation function can be constructed by considering the group as a single compact entity that is basically the union of the basic entities' geometry. Approaches for creating this unique entity in order to allow a centralized planning are discussed in [Sudsang and Ponce, 2000]. In the experiments presented in next section, however, we assume an obstacle free environment and rely on the robots local sensors in order to move the group to a specified goal.

It is also important to notice that for manipulation, the values of k_1 and k_2 in Equation (4.7) can be freely chosen (*i.e.* $k_1 < 3k_2$ is unnecessary) and the normalization of the gradient of the navigation function is not required. Then, we will show that even when the robots are in the MAINTAIN mode

trying to preserve the constraints, the whole team (including the object) moves toward the goal. In order to show this, we define the group position, $\bar{q}$, and the group velocity, $\dot{\bar{q}}$, respectively as follows:

$$\bar{q} = \frac{1}{n}\sum_{i=1}^{n} q_i \ , \qquad \dot{\bar{q}} = \frac{1}{n}\sum_{i=1}^{n} \dot{q}_i \ .$$

We will now show that when the robots are either in the MAINTAIN mode or the GOTOGOAL mode, the group velocity is always parallel to team input $u_T = \nabla\Phi(q)$.

Proposition 6.6 ([Pereira et al., 2003c]) *If all robots are in a state of object closure, the controllers in Equations (4.7) and (4.6) guarantee that the group velocity is in the direction of $u_T = \nabla\Phi(q)$ independently of the values of the positive constants k_1 and k_2 and the normalization of such a gradient.*

Proof: Define v_T to be a unit vector perpendicular to u_T. We need to prove that (i) $u_T \cdot \dot{\bar{q}} > 0$ and (ii) $v_T \cdot \dot{\bar{q}} = 0$. Given the control law in Equation (4.7) (recall that there are no desirable neighbors for this task) we can write:

$$u_T \cdot \dot{q}_i = -k_1\, u_T \cdot \left(\alpha_i \nabla g^a + \beta_i \nabla g^b\right) + k_2 \|u_T\|^2 \ ,$$

and therefore:

$$\begin{aligned} \sum_{i=1}^{n} u_T \cdot \dot{q}_i &= -\sum_{i=1}^{n} k_1\, u_T \cdot \left(\alpha_i \nabla g^a + \beta_i \nabla g^b\right) + \sum_{i=1}^{n} k_2 \|u_T\|^2 \\ u_T \cdot \sum_{i=1}^{n} \dot{q}_i &= -k_1\, u_T \cdot \sum_{i=1}^{n} \left(\alpha_i \nabla g^a + \beta_i \nabla g^b\right) + n\, k_2 \|u_T\|^2 \ . \end{aligned} \tag{6.3}$$

For each active constraint with gradient ∇g^k, there is another identical constraint with gradient $-\nabla g^k$ as discussed before. Thus, the summation in the

right hand side of (6.3) is zero and we can rewrite this equation as:

$$u_T \cdot \sum_{i=1}^{n} \dot{q}_i = n\, k_2 \|u_T\|^2 \ .$$

Since k_2 is a positive constant,

$$u_T \cdot \dot{\bar{q}} = k_2 \|u_T\|^2 > 0 \ .$$

In the same way we can write:

$$v_T \cdot \dot{q}_i = -k_1\, v_T \cdot \left(\alpha_i \nabla g^a + \beta_i \nabla g^b\right) + k_2\, v_T \cdot u_T \ ,$$

and, since v_T is perpendicular to u_T,

$$v_T \cdot \sum_{i=1}^{n} \dot{q}_i = -k_1\, v_T \cdot \sum_{i=1}^{n} \left(\alpha_i \nabla g^a + \beta_i \nabla g^b\right) \ .$$

Because the summation in the right hand side is zero,

$$v_T \cdot \dot{\bar{q}} = 0 \ .$$

Therefore, since the group velocity is in the direction of $u_T = \nabla\Phi(q)$, the ensemble follows the reference input toward the goal. ■

6.4 Experimental Results

In this chapter we show experimental results with our team of polygonal car-like robots. Videos of the experiments can be found in [Pereira, 2003]. In the first set of experiments presented, we have used a triangular cardboard box as the object to be manipulated. The box corners were marked with color

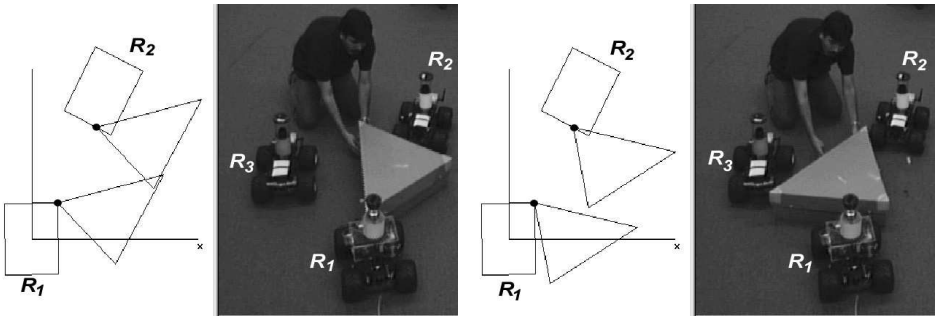

Figure 6.8: Three robots caging a triangular object. R_1's computation of $\mathcal{C}_{obj\_1}$ and $\mathcal{C}_{obj\_2}$ for the imaginary point robots located at the closest pair of points is shown. The overlap (left) indicates the object is constrained for this specific orientation, and the lack of overlap (right) shows that object closure is not maintained for this slice of the configuration space.

paper in order to facilitate the robots vision processing necessary to track the box. Figure 6.8 illustrates the test for object closure performed by robot R_1. R_1 estimates the position of its neighbor R_2, and the orientation of the object as well. Object tracking and localization are discussed in Apendix B. In what follows R_1 computes $\mathcal{C}_{obj\_1}$ and $\mathcal{C}_{obj\_2}$ based on its estimate of the pair of closest points, one on R_1 and one on R_2. As the figure shows, the snapshot on the left shows overlap and therefore a positive test for object closure. The snapshot on the right shows that the object can actually escape. A similar test (not shown in the figure) needs to be performed with robot R_3.

Data collected from an overhead camera for typical experimental runs are shown in Figures 6.9 and 6.10. Robots R_1, R_2, and R_3 transport the triangular box toward a goal position. Figure 6.9 shows a situation where robots R_2 and R_3 change their control behaviors in order to perform the task. In Figure 6.10, the actual $\mathcal{C}_{OBJ\_i}$ for the rectangular robot geometry is overlaid on the experimental data. Note, however, that the robots do not use the $\mathcal{C}_{OBJ\_i}$ for maintaining object closure, but instead they work with the virtual point robots as explained in Section 6.2.4. The object is caged in the

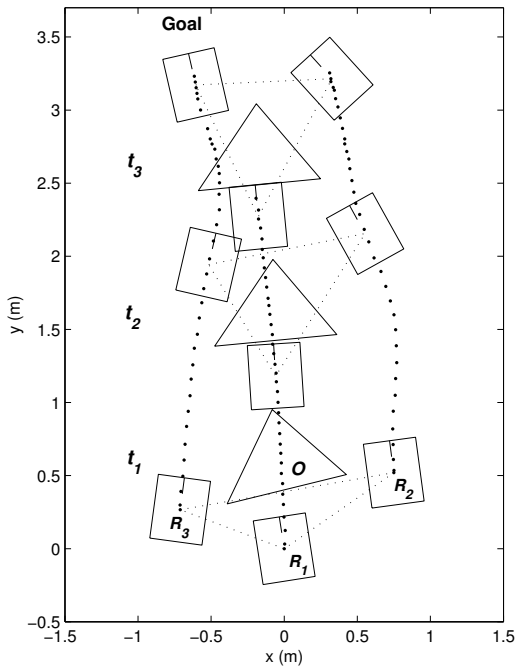

Figure 6.9: Object transportation: t_1 – R_2 and R_3 are in the ACHIEVE mode (see Figure 4.2, page 50) trying to achieve object closure; t_2 – Object closure constraints are satisfied, R_2 and R_3 are in the MAINTAIN mode; t_3 – The robots are in the GOTOGOAL mode. R_1 is in the GOTOGOAL mode in all three snapshots.

three snap-shots shown.

Figures 6.11 and 6.12 show experiments with a circular, active object. A holonomic robotic platform (Nomad XR4000) was used as an object. Figure 6.11 shows robot R_2 performing a test for object closure. The circumferences in this figure are only illustrative because, as mentioned in Section 6.2.5, the circular shape of the robot (object) reduces the caging test to a simple comparison of the Nomad diameter with the distance between the robots' closest pair of points. Figure 6.12 shows an experimental trial where four robots are caging the XR4000 and moving it to a new position. Four snapshots of the experiment are shown. In this experiment, the Nomad is running a simple infrared based obstacle avoidance that considers the other robots as

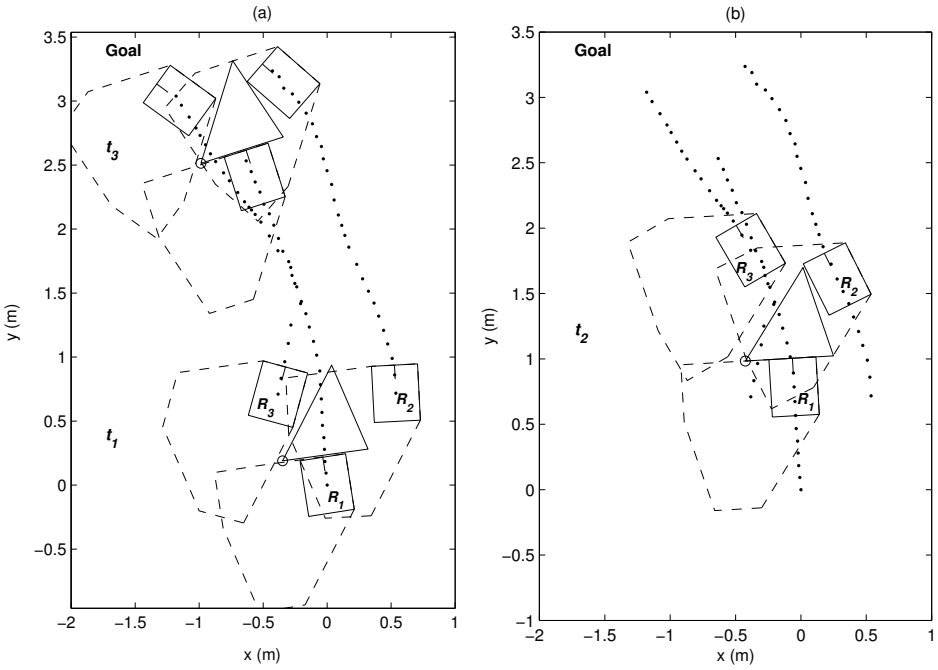

Figure 6.10: The actual $\mathcal{C}_{OBJ\_i}$ (dashed polygons) for each robot. The origin of the object ($\circ$) is always inside $\mathcal{C}_{cls}$ (the compact set delimited by the three $\mathcal{C}_{obj\_i}$) indicating an object closure condition. (a) – initial and final configurations; (b) – an intermediate configuration.

obstacles.

6.5 Concluding Remarks

This chapter presented a methodology for manipulating objects with multiple mobile robots combining the paradigms of pushing and caging. We defined the concept of object closure, a condition that ensures the objects are caged during manipulation.

There are two main advantages of our approach. The decentralized algorithms only rely on the robots' ability to estimate the positions of their neighbors. Because robots are easily instrumented (in our case, this is done by tagging them with colored collars), this is relatively easy even in an un-

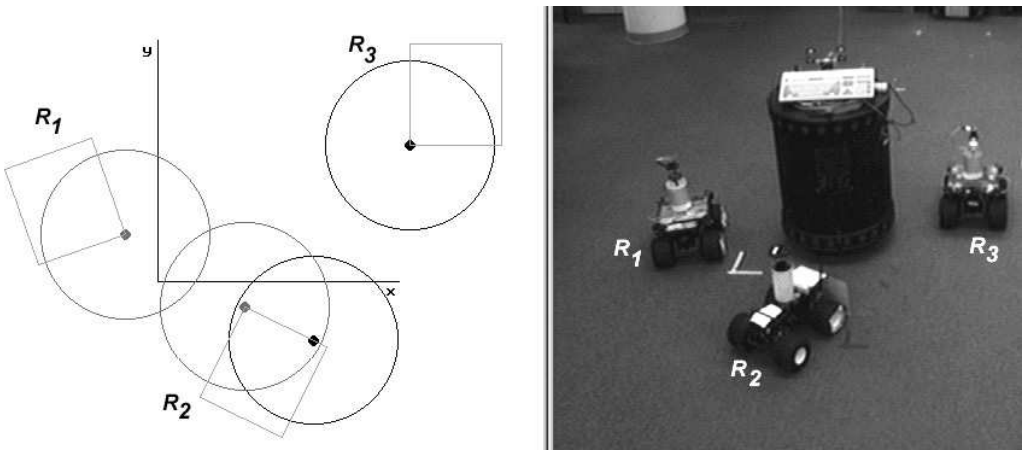

Figure 6.11: Test for object closure when the object (in this case another robot) has a circular shape. An overlap between C_{obj_1} and C_{obj_2} of the imaginary point robots at R_1 and R_2 indicates that the circular robot cannot scape using this space. The same test for R_2 and R_3 indicates that the robot can scape through this space.

structured environment. Therefore, our methodology is potentially scalable for larger groups of robots operating in unstructured environments. Second, our algorithms do not rely on exact estimates of the position and orientation of the manipulated object. Therefore they are robust to measurement errors in this variables.

The main limitations of the algorithms used here include (i) the assumption of convex shapes; (ii) the over approximation that is involved in verifying object closure when rotations are present; and (iii) the use of the virtual point robots which result in sufficient conditions for maintaining object closure. All these yield in conservative results with associated degradation in performance. For example, ensuring object closure with concave objects is often simpler than is the case for convex objects. However, these assumptions and over approximations enable real-time performance and decentralized decision making with *guarantees*, and are important from a practical standpoint.

There are important directions of future work. Firstly, we do not specifi-

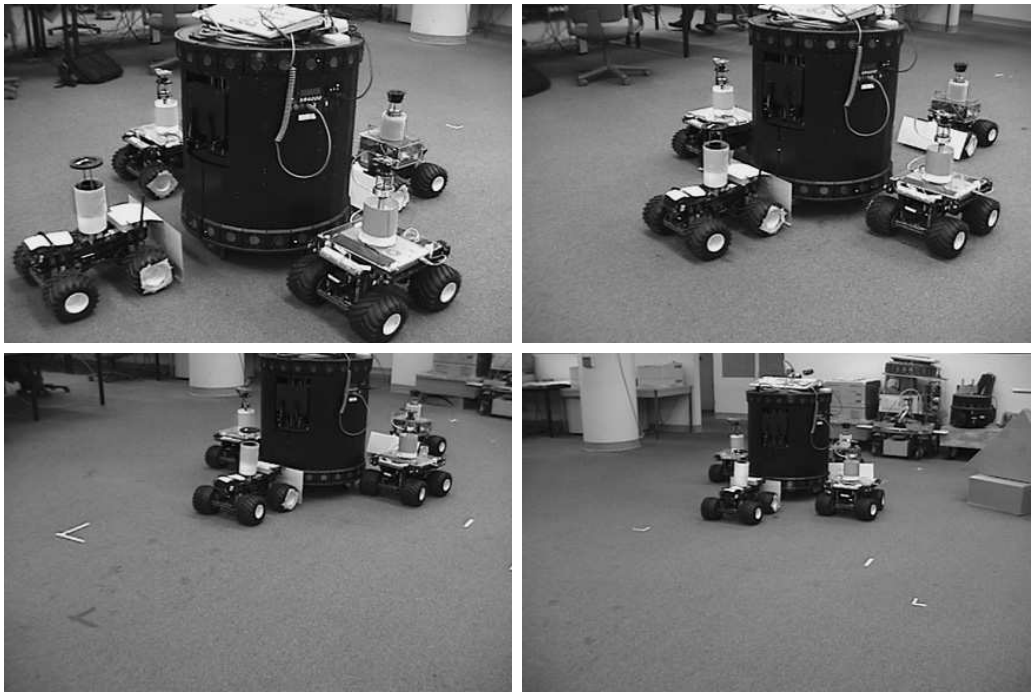

Figure 6.12: Four robots caging a circular holonomic robot. Four snapshots of the experiment are shown from left to right, and top to bottom.

cally consider algorithms for acquiring the object, establishing object closure (the ENCLOSE mode) here. The paper [Song and Kumar, 2002] provides some approaches to this, with guarantees for small (3-4) teams of robots. There are challenges in designing decentralized policies that scale up to large numbers of robots. One of the key steps here is to remove the assumption related to non-essential robots. Also, we do not address the precise positioning and orienting of the object. In this case it is also essential to plan trajectories for the individual robots, instead of simply prescribing a common feedback control signal. The work in [Sudsang and Ponce, 2000] provides a starting point in this direction.

Chapter 7

Application to Formation Control

In this chapter we present a variant of the methodology in Chapter 4. We slightly change the methodology in order to allow rigid formations. We also introduce the term *cooperative leader-following*. This formation control approach is introduced in Section 7.1 and is mathematically defined in Section 7.2. New potential functions for leader-following are in Section 7.3 and the use of control laws of Chapter 4 for formation control is discussed in Section 7.4. Experiments are presented in Section 7.5 and concluding remarks are made in Section 7.6.

7.1 Introduction

Many approaches for motion coordination of large scale multi-robot systems use the leader-following framework [Desai et al., 1998, Balch and Arkin, 1998, Lawton et al., 2000]. In this framework, each robot has a designated leader. Thus, each robot is a follower that tries to maintain a specified relative configuration to its leader(s), which can be other robots in the group or a virtual robots representing pre-computed trajectories supplied by a higher level planner.

One disadvantage with this framework is that there is an explicit dependence of the motion of followers on their leaders, but the leaders' motion is independent of their followers. If, for example, a robot fails or slows down, its followers' motion will be directly affected by this behavior, while its leaders will continue their task without modifying their plans. In situations where it is important to maintain a sensing or communication network, a single failure could result in the failure of the task.

Most of the formation control approaches in the literature do not explicitly address communication and sensing constraints. As in Chapter 5, these constraints are generally inequalities in the configuration space. When it is possible to arrive at a rigid formation that satisfies all the constraints, it is meaningful to follow the formation. However, such configuration constraints change dynamically and it may not be possible to plan and execute changes in formation shape every time a configuration constraint changes.

In this chapter we modify the notion of leader-following and present a framework where robots change their motion plans in real time in order to satisfy constraints related to other robots. These constraints may have to do with a task of maintaining a pre-specified formation but, alternatively, robots may have constraints because of limited ranges or fields of view of sensors, or of communication radios and antenna. Thus we introduce *cooperative leader-following*, a modification of the standard leader-following approach, where the motion of the robots may be dependent not only on their leaders but also on other robots including their followers.

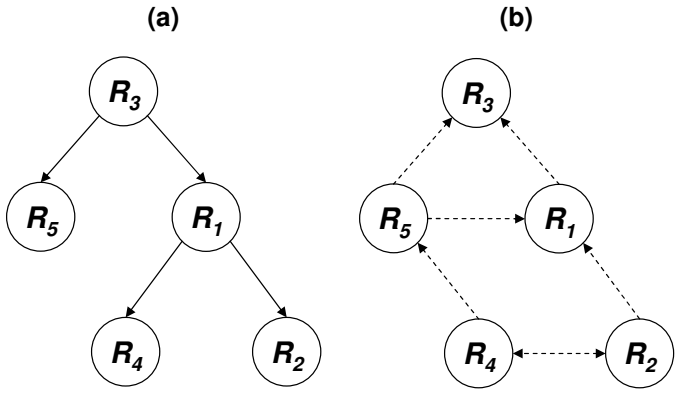

Figure 7.1: Graph modeling for a group of 5 robots: (a) – formation control graph; (b) – constraint graph.

7.2 Problem definition

In this chapter, a formation of n robots will be represented by a directed graph called *formation control graph*[1], $G_f = (\mathcal{R}, \mathcal{E}_f)$, and a second directed graph called *constraint graph*, $G_c = (\mathcal{R}, \mathcal{E}_c, \mathcal{G})$, where $\mathcal{R}$ is the set of nodes, $\mathcal{E}_f$ and $\mathcal{E}_c$ are edge sets and $\mathcal{G}$ is the set of formation constraints as in Chapter 4.

For the formation control graph, G_f, each edge $e_{ij} = (R_i, R_j) \in \mathcal{E}_f$ is associated with a specification for R_j following R_i. For each edge, R_i is the *leader* and R_j is the *follower*. The robot that does not have any leaders and is responsible for guiding the others through the environment is called the *lead robot* [Desai et al., 1998]. Only one lead robot is allowed in our approach. Also, the robots that do not have any followers are called *terminal followers*. Figure 7.1(a) shows an example of a formation control graph where R_3 is the lead robot and R_2, R_4 and R_5 are terminal followers. Robot R_1 follows R_3 and is followed by R_2 and R_4.

The edges $e_{ij} = (R_i, R_j) \in \mathcal{E}_c$ of G_c are associated with constraints on

[1]The term *control graph* is used in [Desai et al., 1998, Das et al., 2002c] to describe what we are calling a formation control graph.

relative position and orientation. While $\mathcal{E}_f$ describes leader-following relationships and set-points for the shape of the formation, $\mathcal{E}_c$ describes inequalities that reflect constraints such as communication and sensing constraints. Figure 7.1(b) shows an example of a constraint graph. In this figure R_3, for example, needs to maintain constraints with respect to R_1 and R_5. The bidirectional edge between R_2 and R_4 indicates that these robots need to maintain constraints with each other.

With this model, as in previous chapters, the control problem can be divided in two parts namely *graph assignment* and *controller design*. The first problem involves designing G_f and G_c and is not our main focus. Measures of performance that depend on G_f are discussed in [Tanner et al., 2002] and heuristics for selecting edges are described in [Das et al., 2002b]. We are concerned with the problem of maintaining the formation described by G_f and the constraints described by G_c. We assume that graphs themselves are preassigned and focus our attention on controlling the robots to satisfy the edge specifications. For G_f, the specification for each edge is a configuration for robot R_j with respect to its leader R_i. On the other hand, the specification for each edge in G_c is a convex function $g(q_i, q_j)$ that represents the allowable configuration space for R_j parameterized by the configuration of R_i. While G_f specifies, for each robot (except the lead), a unique point in configuration space, G_c specifies the allowable subset of configuration space.

Although G_f and G_c are apparently independent, in order to allow robot R_i to reach its set-point $q_i^d(q_j)$ specified by G_f and still satisfy the constraints specified by G_c, we need to guarantee that, except for the lead robot, R_L, the configuration q_i^d is inside the allowable configuration space $\mathcal{C}_i^d(q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n)$ defined by all constraints in G_c. Thus, the edge

definition for the two graphs must satisfy the following condition:

$$(q_1^d, q_2^d, \ldots, q_{L-1}^d, q_{L+1}^d, \ldots, q_n^d) \in \mathcal{C}_1^d \times \mathcal{C}_2^d \cdots \times \mathcal{C}_{L-1}^d \times \mathcal{C}_{L+1}^d \cdots \times \mathcal{C}_n^d \,. \quad (7.1)$$

Moreover, since each $\mathcal{C}_i^d$ is an intersection of convex sets (and hence is convex), the right hand side of Equation (7.1) is also a convex set. Therefore, if the robots' configurations are initially inside this set, the robots can always reach their goals without violating their motion constraints.

Our goal is to design control laws that take in account the formation set-points and the allowable configuration spaces. Before continuing any further we will make an assumption:

Assumption 7.1 G_f is acyclic and the in-valency at each node is 1. In other words, every follower has only one leader[2].

7.3 Potential Functions

The methodology presented in Chapter 4 assumes a potential function, in form of a navigation function for each robot. In the leader-following problem, we chose a navigation function (Section 3.5) as a potential function for the lead robot. For the robots that have a leader robot, the potential function is constructed as a function of the leader's position. In the case where R_i follows R_j, we can describe the follower's relative configuration in local coordinates as $\bar{q} = (q_j - q_i)$. We consider a quadratic Lyapunov function candidate of the form:

$$\Phi_i(\bar{q}) = \frac{1}{2}\|\bar{q}^d - \bar{q}\|^2 \,.$$

[2]This is somewhat restrictive since the in-degree for systems with two inputs can be up to two [Desai et al., 1998, Das et al., 2002a].

If Φ_i is a Lyapunov function we can use it as a leader-following potential function. The input for the follower robot R_i is then given by the negated gradient of $\Phi_i(q_i, q_j)$ as:

$$u_i = -k\nabla\Phi_i = -k(\bar{q}^d - \bar{q})\,,$$

where k is a positive constant and $\nabla\Phi_i = \partial\Phi_i/\partial q_i$. The derivative of the potential function for this input is given by:

$$\begin{aligned}\dot{\Phi}_i &= -\nabla\Phi_i \cdot \dot{\bar{q}} = -\nabla\Phi_i \cdot (\dot{q}_j - \dot{q}_i) = -\nabla\Phi_i \cdot (\dot{q}_j - u_i) \\ &= -\nabla\Phi_i \cdot (\dot{q}_j + k\nabla\Phi_i) = -k\|\nabla\Phi_i\|^2 - \nabla\Phi_i \cdot \dot{q}_j\,.\end{aligned}$$

Observe that $\dot{\Phi}_i$ decreases along the system trajectory if:

$$k\|\nabla\Phi_i\|^2 > -\nabla\Phi_i \cdot \dot{q}_j\,.$$

In the worst case, $\dot{q}_j$ and $-\nabla\Phi_i$ are parallel and the previous condition can be written as the following sufficient condition:

$$k\|\nabla\Phi_i\| > \|\dot{q}_j\|\,. \tag{7.2}$$

Because a real robot is subject to dynamics, there is a practical limit on its velocities. We assume each robot (*i.e.*, all leaders) have a maximum velocity of $\dot{q}_{max}$. From Equation (7.2), it is clear that if we exclude the region given by the ball:

$$\|\bar{q}^d - \bar{q}\| < \gamma = \frac{\dot{q}_{max}}{k}\,,$$

$\dot{\Phi}_i$ decreases along the trajectories of the system. Thus we can show that trajectories that start outside the ball (*i.e.*, when $\|\bar{q}^d - \bar{q}\| > \gamma$), will converge

to the ball. In other words, $\dot{\Phi}_i < 0$ for $\|\bar{q}^d - \bar{q}\| \geq \gamma$. The constant γ is the maximum allowable steady state error in $\bar{q}$.

We note that it is possible to make γ arbitrarily small by allowing for feedforward control. If the follower input is given by:

$$u_i = -k\nabla\Phi_i + \dot{q}_j = -k(\bar{q}^d - \bar{q}) + \dot{q}_j \,,$$

where $\dot{q}_j$ is feedforward information, the controller exponentially converges to $\bar{q} = 0$. The feedforward velocity requires estimation of the leader's velocity by the follower robot and is discussed in [Das et al., 2002a].

7.4 Controllers

Based on the two graphs, G_f and G_c, described in Section 7.2 we define a third graph that will govern the switching between the three controllers presented in Chapter 4. We call this time dependent graph that changes with the state of the robots, the *formation graph* $H = (\mathcal{R}, \mathcal{E}_h)$, where $\mathcal{E}_h$ is defined as the union of two subsets of $\mathcal{E}_f$ and $\mathcal{E}_c$:

$$\mathcal{E}_h = \bar{\mathcal{E}}_f \cup \bar{\mathcal{E}}_c \,,$$

where:

$$\bar{\mathcal{E}}_f = \{e_{ij} | \, [e_{ij} \in \mathcal{E}_f] \wedge [g(k,j) < 0 \, \forall e_{kj} \in \mathcal{E}_c, R_k \in \mathcal{R}]\} \,,$$

$$\bar{\mathcal{E}}_c = \{e_{ij} | \, [e_{ij} \in \mathcal{E}_c] \wedge [g(i,j) \geq \delta]\} \,.$$

Observe that the edges of the formation control graph G_f appears in H if all constraints for the follower robot are satisfied. On the other hand, edges of the constraint graph G_c are included in H if the constraint relative to this

edge is active.

Thus based on H, each robot R_i can be in one of the three basic behaviors or modes presented in Chapter 4, depending on the number and type of incoming edges at R_i. The control law for these modes are also similar to the ones presented in Chapter 4. The only difference here is that, except for the lead robot, the navigation functions is replaced by the potential functions of Section 7.3. Thus, if there is only one incoming control edge ($e_{ji} \in \mathcal{E}_f$), the robot is in the GoToGoal mode and is simply following the negated gradient of its potential function (navigation function for the lead robot). When all incoming edges are constraint edges ($e_{ji} \in \mathcal{E}_c$), R_i is in the Achive mode where it is trying to satisfy the constraints. A robot switches to the Maintain mode if it has one incoming control edge ($e_{ji} \in \mathcal{E}_f$) and at least one constraint edge ($e_{ji} \in \mathcal{E}_c$).

As an example of how the switching among the control modes is governed, consider a possible H generated by the combination of the two graphs of Figure 7.1. In Figure 7.2, the dotted arrows show active constraints, while solid arrows denote equality specifications. R_5 is in the Unsafe mode, R_1 is in the Safe mode, and R_2, R_3, and R_4 are in the Critical mode of the controller.

One can notice that, in opposition to previous chapters, for leader following we do not require the constraint graph to have a Hamiltonian path. Hence, we can observe that for some graphs the proof of completeness (Proposition 4.2) may fail, even if the robots are instantaneously following the same potential (navigation) function's gradient. This is because we cannot guarantee that the robots will never enter in their unsafe spaces. In general, however, each leader has to maintain constraints with its follower. Therefore, for each edge in the formation graph, G_f, there is another one in the

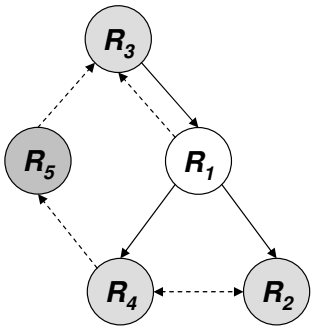

Figure 7.2: A formation graph based on the combination of the graphs of Figure 7.1. Based on incoming edges, R_5 has configuration space constraints on its position relative to R_4, R_1 follows a potential function to acquire a position relative to R_3, while R_2, R_3, and R_4 must execute a combination of two reactive behaviors.

constraint graph, G_f, with opposite direction. This two edges will work together to maintain the robots inside their configuration spaces.

The above analysis lends itself to stronger results for specific graphs. Consider, for example, a group of robots in a linear formation where each robot has to satisfy a constraint with its immediate follower. The system would never enter in the unsafe mode if for a generic active constraint, $g(q_i, q_j)$, the control input ensure $\dot{g}(q_i, q_j) \leq 0$, as discussed in Chapter 4. Remember that the time derivative of $g(q_i, q_j)$ is given by:

$$\dot{g}(q_i, q_j) = \frac{\partial g}{\partial q_i}\dot{q}_i + \frac{\partial g}{\partial q_j}\dot{q}_j \,. \tag{7.3}$$

Denote $\partial g/\partial q_i$ by ∇g^i, and $\partial g/\partial q_j$ by ∇g^j. The key observation is $\partial g/\partial q_j = -\partial g/\partial q_i$, or $\nabla g^i = -\nabla g^j$.

In the worst case, both the leader R_i and the follower R_j are in their critical spaces. Then, assume that there is a constraint $g(q_i, q_j)$ with gradient ∇g^j active for the leader R_i and a constraint $g(q_j, q_k)$ with gradient ∇g^k active for the follower R_j. Substituting $\dot{q}_i$ and $\dot{q}_j$ in (7.3) by the control

inputs in Equation (4.7) we rewrite the time derivative of $g(q_i, q_j)$ as:

$$\dot{g}(q_i, q_j) = -\nabla g^j \cdot \left(k_1 \nabla g^j + k_2 \nabla \phi_i\right) + \nabla g^j \cdot \left(k_1 \nabla g^k + k_2 \nabla \phi_j\right) . \qquad (7.4)$$

For the specific case of circular constraints ($g(q_i, q_j) = (x_i - x_j)^2 + (y_i - y_j)^2 - r^2$), observe that ∇g^j is anti-parallel to $\nabla \phi_j$. Also, notice that, for a linear formation in steady-state, $\nabla \phi_i$ and ∇g^k are anti-parallel as well. Grouping those vectors together we rewrite the previous equation as:

$$\dot{g}(q_i, q_j) = -\nabla g^j \cdot (k_1 \nabla g^j - k_2 \nabla \phi_j) + \nabla g^j \cdot (k_1 \nabla g^k - k_2 \nabla \phi_i) , \qquad (7.5)$$

where the first term is contributing to satisfy the constraints and the second has the opposite effect.

By Equation (7.5), observe that if $\nabla g^k = 0$ the constraint is satisfied. This is because the sum of the parallel vectors $k_1 \nabla g^j$ and $-k_2 \nabla \phi_j$ will be always bigger than the vector $-k_2 \nabla \phi_i$. Thus, starting with the terminal follower, for which ∇g^k does not exist, and finishing in the lead robot, it is easy to see that all constraints can be satisfied. Again considering the chain of robots, notice that small velocities of the lead robot (small $\nabla \phi_i$) contribute to satisfy the constraints. Also, observe that if robot R_j fails, causing $\nabla \phi_j = \nabla g^k = 0$, the constraint will eventually be violated and R_i will enter its unsafe configuration space. In practice, this situation causes all interconnected robots to continuously switch between the ACHIEVE and MAINTAIN modes forcing the group to stop. It is in some way a desirable condition since the main idea of the methodology is forcing the robots to wait for their teammates. Other similar observations can be made depending on the values of the other terms in Equation (7.5).

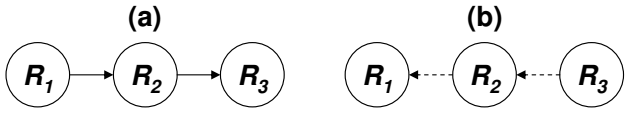

Figure 7.3: (a) – The control graph and (b) – the constraint graph for the first experiment.

7.5 Experimental Results

In this chapter we have also used the two teams of mobile robots shown in Apendix A. Videos of the experiments can be found in [Pereira, 2003]. With the car-like robots, because visibility of other robots is important for orientation estimation, as in Chapter 5, three robots must maintain sensing constraints with their neighbors. Thus, in our first experiment, the three robots are commanded to maintain a line formation as shown by G_f and G_c in Figure 7.3. The function $g(q_i, q_j)$ was set to be a circle of radius 1.6 m. Observe that this radius is much smaller than the distance where the robots are actually blind (2 m) in order to guarantee that the task is completed even if the robots enter in their unsafe configuration spaces. The threshold δ for the critical region was chosen to make it a circle of 1.3 m radius.

Figure 7.4 shows four snapshots of our experiment. In (c) the terminal follower (R_3) was manually stopped. In what follows all the robots switch to their ACHIEVE modes. When R_3 starts moving again the robots switch back to their MAINTAIN and then GOTOGOAL modes and complete their tasks. Figure 7.5 shows the y coordinates of the robots for the same experiment.

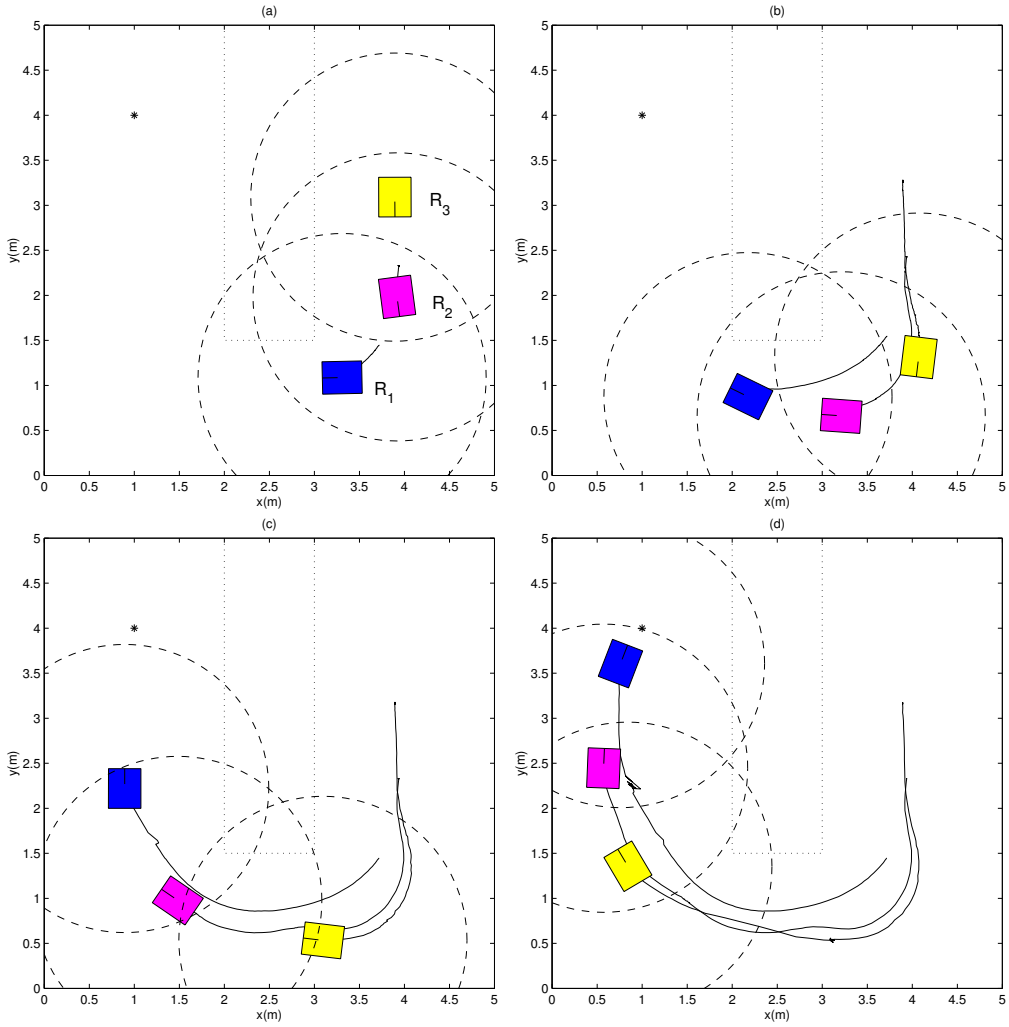

Figure 7.4: Four snapshots of an experiment where three robots are in line formation and keeping visibility constraints with their followers. The goal configuration for the lead robot is marked with a (*). The dashed circumferences represent the sensors' field of view. In (c) robot R_3 was manually stopped for 7 seconds. The robots stop following their potential functions and wait for R_3 so that the constraints are preserved.

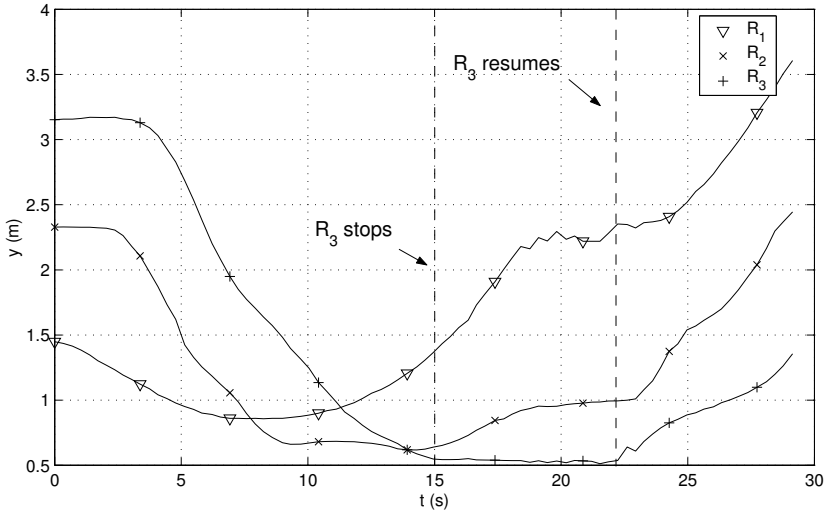

Figure 7.5: The y coordinate for the experiment in Figure 7.4. The terminal follower, R_3, was stopped for approximately 7 s at the time 15 s. It causes the other robots to switch to their UNSAFE modes and stop, as expected.

We have also implemented the approach in our team of holonomic robots. Three robots were programmed to keep control and constraint graphs shown in Figure 7.6. Figure 7.7 shows four snapshots of our experiment. Observe that the robots keep a triangular formation in all snapshots shown. In Figure 7.7(c), R_3 is turned off. The formation is preserved because the lead robot R_1 enter the UNSAFE mode and stops. This also causes R_2 to stop. The robots proceed moving as soon as R_3 backs on. This temporal behavior can be seen in Figure 7.8, which shows the y coordinates of the robots.

7.6 Concluding Remarks

This chapter presented another example of our general framework for motion planning and control of teams of cooperating robots. Observe that we slightly modified the approach by considering time- and leader-dependent potential functions instead of statical navigation functions for each robot. We

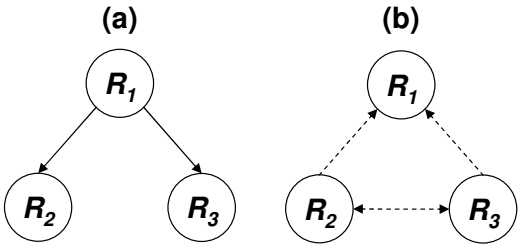

Figure 7.6: (a) – The control graph and (b) – the constraint graph for the second experiment.

also did not impose any constraint on the neighborhood relationship of the robots, allowing any kind of graph. With this generalization, the strong theoretical results of Chapter 4 no longer apply for any formation.

Using our approach we have modified the traditional leader-following methodologies in order to transform it in a tightly coupled cooperative task. Because the motion of the leader does not depends on the followers' motion, the methodologies encountered in the literature so far are loosely coupled. We have shown that with this framework, it is possible to set the graphs in a way a leader waits for its followers. Thus, they are in one sense, more robust to failures.

In this chapter we have imposed sensing or communication constraints to the robots in the constraint graph. Future work includes introducing manipulation constraints, as shown in Chapter 6. In this way we can improve the manipulation task, because, by setting a basic rigid formation for the robots, collisions with the object being manipulated could be avoided. Therefore, we can relax the assumption related to rigid objects and work with deformable and fragile ones.

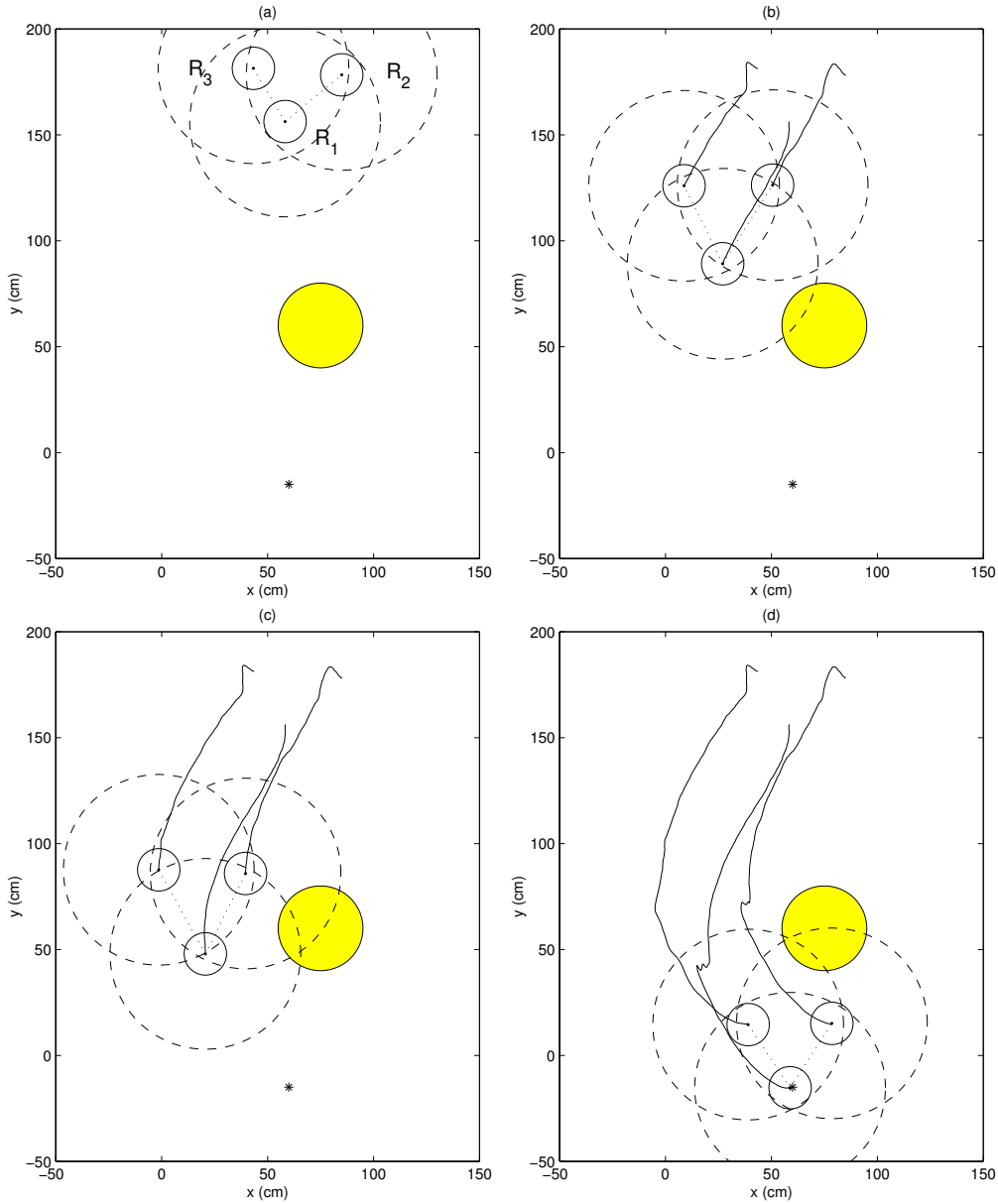

Figure 7.7: Four snapshots of an experiment where three holonomic robots are in a triangular formation. The goal configuration for the lead robot is marked with a (*). The dashed circumferences represent the boundaries of the valid configuration space. In (c) robot R_3 was stopped. The lead robot (R_1) stop following its navigation function, waiting for R_3 causing its other follower, R_2, to stop as well.

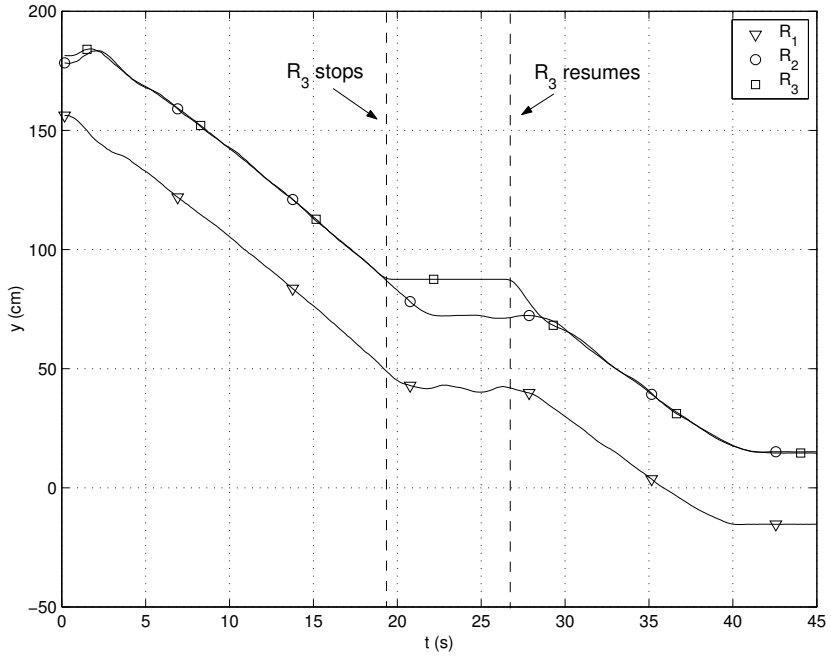

Figure 7.8: The y coordinate for the experiment in Figure 7.7. The terminal follower, R_3, was stopped for approximately 8 s at the time 19 s. It causes the lead robot to switch to its ACHIEVE mode and stop, as expected.

Chapter 8

Conclusions and Future Work

Search others for their virtues, thyself for thy vices.

Benjamin Franklin (1706–1790)

8.1 Summary

We have proposed a general framework of motion planning and control for coordinating the execution of tasks by groups of cooperating mobile robots. We have shown that this framework is suitable for solving several types of cooperative tasks with different groups of robots. The idea of a framework such as this one would be to reduce several different cooperative tasks to a problem of motion planning and control and consequently use the same algorithm for all of them. Problem reduction is a well known mathematical methodology. One such example is in the computability theory of NP-complete problems that states: "There is always a polynomial-time algorithm for transforming an instance of any NP-complete problem into an instance of any other NP-complete problem. So if you could solve one you could solve any other by transforming it to the solved one." [Cormen et al., 1990]. In robotics, a large number of cooperative problems are not of the NP-complete class and some may even present several solutions. In order to allow a single team of robots

to perform several different tasks, we have shown that it is feasible to provide each robot with a single suite of algorithms parameterized only by the specific characteristics of the task. Therefore, new, and probably unknown, tasks were successfully executed by the team of robots once this particular task was transformed to one of the tasks the robots were able to execute. This is actually the main advantage of our framework over previous works in the literature. While in previous methodologies each task is treated individually, in our framework a new task is seen as an instance of an existent one for which solutions and algorithms are already known.

In this work we propose reducing cooperative problems to individual constrained motion planning problems. We believe that this is the most adequate, and perhaps the only robotic class of problems that can map (or be mapped to) all types of cooperative tasks. The objective in the constrained motion planning problem, for which several solutions can be found in the literature, is to drive an individual robot to an independent goal position, specified by the task, while satisfying temporal constraints. So far, most works have considered temporal constraints related to dynamical obstacles, people and other robots in the environment. We have extended this concept to cooperative environments and have considered several kinds of interaction among the robots during the task execution as dynamical constraints. This enables each robot to work as if it were alone, since it knows the dynamical constraints it is subject to. Therefore the cooperative problem can be inherently distributively solved. Hence, our approach leads to general and scalable solutions to cooperative problems.

We also proposed a simple solution for the constrained motion planning problem that involves on-line modification of pre-computed navigation functions for each robot. Independent navigation functions, which are proven

to lead individual robots to their respective goals, are combined with control laws that guarantees constraints satisfaction. Our methodology was also experimentally validated, presenting encouraging results.

Since our approach in mobile robotics is to guarantee successful task completion at the same time within safety constraints, performance and precision are not our main objectives. Therefore, for each problem our solution is not guaranteed to be optimal with respect to time, traveling distance, power consumption or other requirement. However, these apparent drawbacks did not present a hindrance to develop simple, reliable, and robust algorithms that can be easily reproduced and implemented on a large variety of groups of robots executing a broad spectrum of tasks.

From a theoretical standpoint, our work presents relevant contributions. To our knowledge, reducing one robotic problem to another one has not been addressed in the literature. This idea opens new research possibilities and future robotic efforts could concentrate in developing algorithms and tools to transform new and therefore yet unsolved problems to well known problems that have well known solutions. We believe, however, that the formalization of this theory and the characterization of the problems are not easy tasks, and much investigation is still necessary.

Among our main contributions is the manipulation methodology presented in Chapter 6. By relaxing some of the assumptions related to force contacts between the robots and the object being manipulated, we were able to develop robust and scalable algorithms for multi-robot manipulation. Moreover, differently from sensing and communication problems, we have been able to relax most of the assumptions and still come up with an exact solution for the manipulation problem for teams of either point or circular robots.

We also presented relevant results in the formation control example in Chapter 7, which is a novel approach. At this point we were not able to present tight guarantees for all possible formations. However, experimental trials have shown that, even without theoretical guarantees, the methodology can be practically applied to the leader-following problem and used to control a rigid formation.

Portions of this work have been published in international peer reviewed conference proceedings, workshop books, and journals:

1. G. A. S. Pereira, V. Kumar, J. Spletzer, C. J. Taylor, and M. F. M. Campos, "Cooperative transport of planar objects by multiple mobile robots using object closure," in *Experimental Robotics VIII*, (B. Siciliano and P. Dario , eds.), pp. 275–284, Springer, 2002.

2. G. A. S. Pereira, V. Kumar, M. F. M. Campos, "Decentralized Algorithms for Multirobot Manipulation via Caging", *Algorithmic Foundations of Robotics V*, (K. Goldberg, S. Hutchinson, J. Burdick, and J-D. Boissonnat , eds.), pp. 242–258, Springer, 2003.

3. G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos, "Decentralized Motion Planning for Multiple Robots subject to sensing and communication constraints," in *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, (A. Schultz, L. E. Parker, and F. Schneider, eds.), pp. 267–278, Kluwer Academic Press, 2003.

4. G. A. S. Pereira, V. Kumar, M. F. M. Campos, "Localization and Tracking in Robot Networks" *International Conference on Advanced Robotics (ICAR'03)*, (Coimbra, Portugal), pp. 465–470, 2003.

5. G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos, "For-

mation control with configuration space constraints," in *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems (IROS'03)*, (Las Vegas, USA), pp. 2755–2760, 2003.

6. G. A. S. Pereira, V. Kumar, and M. F. M. Campos, "A framework for motion planning of cooperative mobile robots," in *Anais do VI Simpósio Brasileiro de Automação Inteligente (SBAI'03)*, (Bauru, SP), pp. 846–851, 2003.

7. G. A. S. Pereira, V. Kumar, and M. F. M. Campos, "Decentralized algorithms for multi-robot manipulation via caging," *International Journal of Robotics Research (IJRR)*, 2003 (to appear).

8.2 Future Work

Important and necessary theoretical investigations need to be considered for future work. The methodology proposed in Chapter 4 guarantees that each robot will satisfy the constraints when it has at most two neighboring robots. The strong results of this methodology guarantees robot formations that can be represented by Hamiltonian graphs. However, our methodology cannot guarantee other simple formations such as those represented by simple trees (other than a Hamiltonian path). In order to introduce this aspect to our methodology, we are currently looking at recent solutions for leader-following, where each robot may have up to two leaders and infinite followers. Our first effort in this direction was shown in Chapter 7.

Another important direction for future work is neighborhood assignment. A close form solution for this problem would be very important, not only for the methodology presented in this work, but also for other methodologies

such as those based on formation control using leader-following, where leader assignment is required.

Also, it is necessary to explicitly model the robot's non-holonomic behavior. We believe that elegant and generic solutions are difficult to be obtained if potential function based controllers are used. However, those solutions may be satisfactorily achieved if online trajectory generators are applied. An example is presented in [van Nieuwstadt and Murray, 1998].

Finally, it would be very important to develop a theory that classifies all possible cooperative tasks and indicates which class is more suitable to be reduced to a motion planning problem. A coarse approximation to this classification may already exist if tasks are divided in "loosely"and "tightly"coupled tasks. If this partitioning of the task space is valid, then the problem remains, which is to prove this partioning.

From the experimental point of view, future work includes implementing the motion planning methodology in all-terrain mobile robots working in an outdoor environments, guided by GPS (Global Positioning Systems) and cameras for inter-robot localization. We also intend to experiment the methodology with subaquatic or aerial robots, such as the Aurora autonomous blimp [Elfes et al., 1998]. Cooperation between aerial and ground robots is under way. We believe that in such situations, the simplicity and robustness of our method will present a fundamental differential when compared to other solutions that use more complex sensoring and computational resources.

Bibliography

[Ando et al., 1995] Ando, H., Suzuki, I., and Yamashita, M. (1995). Formation and agreement problems for synchronous mobile robots with limited visibility. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 453–460.

[Arai et al., 1989] Arai, T., Ogata, H., and Suzuki, T. (1989). Collision avoidance among multiple robots using virtual impedance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 479–485.

[Arkin, 1998] Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge, MA.

[Aronov et al., 1998] Aronov, B., de Berg, M., van der Stappen, A. F., Svestka, P., and Vleugels, J. (1998). Motion planning for multiple robots. In *Proceedings of the Internaltional Symposium on Computational Geometry*, pages 374–382.

[Arulampalam et al., 2002] Arulampalam, S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

[Asama et al., 1989] Asama, H., Matsumoto, A., and Ishida, Y. (1989). Design of an autonomous and distributed robot system: ACTRESS. In *Proceedings of IEEE/RJS International Workshop on Intelligent Robots and Systems*, pages 283–290.

[Balakrishnan, 1987] Balakrishnan, A. V. (1987). *Kalman Filtering Theory*. Optimization Software.

[Balch and Arkin, 1994] Balch, T. and Arkin, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):1–25.

[Balch and Arkin, 1998] Balch, T. and Arkin, R. C. (1998). Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):1–15.

[Beni, 1988] Beni, G. (1988). The concept of cellular robot. In *In Proceedings of IEEE Symposium on Intelligent Control*, pages 57–61.

[Brown and Jennings, 1995] Brown, R. G. and Jennings, J. S. (1995). A pusher/steerer model for strongly cooperative mobile robot manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 562–568.

[Burgard et al., 2000] Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 476–481.

[Chaimowicz et al., 2001a] Chaimowicz, L., Kumar, V., and Campos, M. F. M. (2001a). A framework for coordinating multiple robots in cooperative manipulation tasks. In *Proceedings of SPIE 4571 - Sensor Fusion and Decentralized Control IV*, pages 331–336.

[Chaimowicz et al., 2001b] Chaimowicz, L., Sugar, T., Kumar, V., and Campos, M. F. M. (2001b). An architecture for tightly-coupled multi-robot cooperation. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2292–2297.

[Cormen et al., 1990] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to Algorithms*. MIT Press.

[Das et al., 2002a] Das, A., Spletzer, J., Kumar, V., and Taylor, C. (2002a). Ad hoc networks for localization and control. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2978–2983.

[Das et al., 2002b] Das, A. K., Fierro, R., and Kumar, V. (2002b). Control graphs for robot networks. In Murphey, R. and Pardalos, P., editors, *Cooperative Control and Optimization*, Applied Optimization, chapter 4. Kluwer Academic Press.

[Das et al., 2002c] Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., and Taylor, C. J. (2002c). A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825.

[Davidson and Blake, 1998] Davidson, C. and Blake, A. (1998). Caging planar objects with a three-finger one-parameter gripper. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2722–2727.

[Desai et al., 1998] Desai, J. P., Ostrowski, J., and Kumar, V. (1998). Controlling formations of multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2864–2869.

[Durrant-Whyte and Manyika, 1993] Durrant-Whyte, H. F. and Manyika, J. (1993). *Data fusion and sensor management: a decentralized information-theoretic approach*. Prentice Hall, London, UK.

[Elfes et al., 1998] Elfes, A., Bueno, S. S., Bergerman, M., and Ramos, J. J. G. (1998). A semi-autonomous robotic airship for environmental monitoring missions. In *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, pages 3449–3455.

[Eren et al., 2002] Eren, T., Belhumeur, P. N., , and Morse, A. S. (2002). Closing ranks in vehicle formations based rigidity. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2959–2964.

[Esposito and Kumar, 2002] Esposito, J. M. and Kumar, V. (2002). A method for modifying closed-loop motion plans to satisfy unpredictable dynamic constraints at runtime. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 1691–1696.

[Fox et al., 2000] Fox, D., Burgard, W., Kruppa, H., and Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344.

[Fukuda and Nakagawa, 1987] Fukuda, T. and Nakagawa, S. (1987). A dynamically reconfigurable robotic system (concept of a system and optimal configurations). In *Proceedings of the Conference of the IEEE Industrial Electronics Society*, pages 588–595.

[Gentili and Martinelli, 2000] Gentili, F. and Martinelli, F. (2000). Robot group formations: a dynamic programming approach for a shortest path computation. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 3152–3157.

[Geyer and Daniilidis, 2002] Geyer, C. and Daniilidis, K. (2002). Para-catadioptric calibration. *IEEE Transactions on Pattern Analyzis and Machine Intelligence*, 24(5):687–695.

[Guo and Parker, 2002] Guo, Y. and Parker, L. E. (2002). A distributed and optimal motion planning approach for multiple mobile robots. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2612–2619.

[Hopcroft et al., 1984] Hopcroft, J. E., Schwartz, J. T., and Sharir, M. (1984). On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the Warehouseman's Problem. *International Journal of Robotics Research*, 3(4):76–88.

[Howard et al., 2003] Howard, A., Matarić, M., and Sukhatme, G. (2003). Putting the 'I' in team: an ego-centric approach to cooperative localization. In *Proceedings of the IEEE International Conference on Robotics Automation*, (to appear).

[Howard et al., 2002] Howard, A., Matarić, M. J., and Sukhame, G. S. (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, pages 299–308.

[Jadbabaie et al., 2002] Jadbabaie, A., Lin, J., and Morse, A. S. (2002). Coordination of groups of mobile autonomous agents using nearest neighbor rules. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2953–2958.

[Jennings et al., 1997] Jennings, J. S., Whelan, G., and Evans, W. F. (1997). Cooperative search and rescue with a team of mobile robots. In *Proceedings of the International Conference on Advanced Robotics*, pages 193–200.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98.

[Khatib et al., 1995] Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., Casal, A., and Baader, A. (1995). Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Proceedings of the International Symposium on Robotics Research*, pages 333–342.

[Koditschek and Rimon, 1990] Koditschek, D. E. and Rimon, E. (1990). Robot navigation function in manifolds with boundary. *Advances in Applied Mathmatetics*, 11:412–442.

[Kosuge and Oosumi, 1996] Kosuge, K. and Oosumi, T. (1996). Decentralized control of multiple robots handling an object. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 318–323.

[Kumar et al., 2000] Kumar, A., Sood, R., Saran, H., and Shorey, R. (2000). Performance of multiple TCP connections over different routing protocols in mobile ad-hoc networks. In *IEEE International Conference on Personal Wireless Communications*, pages 500–503.

[Kurazume and Hirose, 1998] Kurazume, R. and Hirose, S. (1998). Study on cooperative positioning system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2896–2903.

[Latombe, 1991] Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA.

[LaValle and Hutchinson, 1998] LaValle, S. M. and Hutchinson, S. A. (1998). Optimal motion planning for multiple robots having independent goals. *IEEE Transactions on Robotics and Automation*, 14(6):912–925.

[Lawton et al., 2000] Lawton, J., Young, B., and Beard, R. (2000). A decentralized approach to elementary formation maneuvers. In *Proc. IEEE Int'l. Conf. Robot. Automat.*, pages 2728–2733.

[Leonard and Fiorelli, 2001] Leonard, N. E. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2968 –2973.

[Lian and Murray, 2003] Lian, F.-L. and Murray, R. (2003). Cooperative task planning of multi-robot systems with temporal constraints. In *Proceedings of the IEEE International Conference on Robotics Automation*.

[Lowekamp et al., 2000] Lowekamp, B. B., O'Hallaron, D., and Gross, T. (2000). Direct queries for discovering network resource properties in a distributed environment. *Cluster Computing*, 3(4):281–291.

[Lynch, 1996] Lynch, K. M. (1996). Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research*, 15(6):533–556.

[MacKenzie and Stout, 1993] MacKenzie, P. D. and Stout, Q. F. (1993). Optimal parallel construction of hamiltonian cycles and spanning trees in random graphs. In *Proc. 5th ACM Symp. on Parallel Algorithms and Architectures*, pages 224–229.

[Majumder et al., 2001] Majumder, S., Scheding, S., and Durrant-Whyte, H. (2001). Multi-sensor data fusion for underwater navigation. *Robotics and Autonomous Systems*, 35(1):97–108.

[Manet, 2003] Manet (2003). Mobile ad hoc networking (manet). http://protean.itd.nrl.navy.mil/manet/manet_home.html.

[Matarić et al., 1995] Matarić, M., Nilsson, M., and Simsarian, K. (1995). Cooperative multi-robot box-pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–561.

[McInnes, 1995] McInnes, C. R. (1995). Autonomous ring formation for a planar constellation of satellites. *AIAA Journal of Guidance Control and Dynamics*, 18(5):1215–1217.

[Mesbahi and Hadaegh, 2001] Mesbahi, M. and Hadaegh, F. (2001). Formation flying of multiple spacecraft via graphs, matrix inequalities, and switching. *AIAA Journal of Guidance, Control and Dynamics*, 24(2):369–377.

[Murray et al., 1994] Murray, R. M., Li, Z., and Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

[Olfati-Saber and Murray, 2002a] Olfati-Saber, R. and Murray, R. M. (2002a). Distributed cooperative control of multiple vehicle formations using structural potential functions. In *The 15th IFAC World Congress*.

[Olfati-Saber and Murray, 2002b] Olfati-Saber, R. and Murray, R. M. (2002b). Graph rigidity and distributed formation stabilization of multi-vehicle systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2965–2971.

[Ota et al., 1995] Ota, J., Miyata, N., and Arai, T. (1995). Transferring and regrasping a large object by cooperation of multiple mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 543–548.

[Parker, 2000] Parker, L. E. (2000). Current state of the art in distributed autonomous mobile robotics. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, pages 3–12, Knoxville, TN.

[Pereira, 2003] Pereira, G. A. S. (2003). Thesis home page. http://www.verlab.dcc.ufmg.br/projetos/gpereira/thesis/.

[Pereira et al., 2000] Pereira, G. A. S., Campos, M. F. M., and Aguirre, L. A. (2000). Data based dinamical model of vision observed small robots. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 3312–3317.

[Pereira et al., 2003a] Pereira, G. A. S., Das, A., Kumar, V., and Campos, M. F. M. (2003a). Decentralized motion planning for multiple robots subject to sensing and communication constraints. In Schultz, A., Parker, L. E., and Schneider, F., editors, *Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II*, pages 267–278. Kluwer Academic Press.

[Pereira et al., 2003b] Pereira, G. A. S., Das, A. K., Kumar, V., and Campos, M. F. M. (2003b). Leader-following with configuration space constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2755–2760.

[Pereira et al., 2003c] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2003c). Decentralized algorithms for multi-robot manipulation via caging. *International Journal of Robotics Research*, (to appear).

[Pereira et al., 2003d] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2003d). Decentralized algorithms for multirobot manipulation via caging. In Goldberg, K., Hutchinson, S., Burdick, J., and Boissonnat, J.-D., editors, *Algorithmic Foundations of Robotics V*, pages 242–258. Springer.

[Pereira et al., 2003e] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2003e). A framework for motion planning of cooperative mobile robots. In *Anais do VI Simpósio Brasileiro de Automação Inteligente*, pages 846–851.

[Pereira et al., 2003f] Pereira, G. A. S., Kumar, V., and Campos, M. F. M. (2003f). Localization and tracking in robot networks. In *Proceedings of the International Conference on Advanced Robotics*, pages 465–470.

[Pereira et al., 2002a] Pereira, G. A. S., Kumar, V., Spletzer, J., Taylor, C. J., and Campos, M. F. M. (2002a). Cooperative transport of planar objects by multiple mobile robots using object closure. In Siciliano, B. and Dario, P., editors, *Experimental Robotics VIII*, pages 275–284. Springer.

[Pereira et al., 2002b] Pereira, G. A. S., Pimentel, B. S., Chaimowicz, L., and Campos, M. F. M. (2002b). Coordination of multiple mobile robots in an

object carrying task using implicit communication. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 281–286.

[Pimentel, 2002] Pimentel, B. S. (2002). Cooperation for communication: exploring ad hoc connectivity in a team of cooperating mobile robots. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG.

[Reif and Wang, 1995] Reif, J. and Wang, H. (1995). Social potential fields: A distributed behavioral control for autonomous robots. In *Proceedings of the International Workshop on Algorithmic Foundations of Robotics*, pages 431–459. A. K. Peters, Wellesley, MA.

[Reshko et al., 2002] Reshko, G., Mason, M., and Nourbakhsh, I. (2002). Rapid prototyping of small robots. Technical Report CMU-RI-TR-02-11, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[Reynolds, 1987] Reynolds, C. (1987). Flocks, birds, and schools: a distributed behavioral model. *Computer Graphics*, 21:25–34.

[Rimon and Blake, 1996] Rimon, E. and Blake, A. (1996). Caging 2D bodies by one-parameter, two-fingered gripping systems. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 1458–1464.

[Rimon and Koditschek, 1992] Rimon, E. and Koditschek, D. E. (1992). Exact robot navigation using artificial potencial functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–517.

[Roumeliotis and Bekey, 2000] Roumeliotis, S. I. and Bekey, G. A. (2000). Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2958–2965.

[Roumeliotis and Bekey, 2002] Roumeliotis, S. I. and Bekey, G. A. (2002). Distributed multirobot localization. *IEEE Transactions on Robotics and Automation*, 18(5):781–795.

[Rus, 1997] Rus, D. (1997). Coordinated manipulation of objects in a plane. *Algorithmica*, 19(1/2):129–147.

[Rus et al., 1995] Rus, D., Donald, B., and Jennings, J. (1995). Moving furniture with teams of autonomous robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 235–242.

[Rybski et al., 2000] Rybski, P. E., Hougen, D. F., Benjaafar, S., Bonney, J., Budenske, J. R., Dvorak, M., Gini, M., French, H., Krantz, D. G., Li, P. Y., Malver, F., Nelson, B., Papanikolopoulos, N., Stoeter, S. A., Voyles, R., and Yesin, K. B. (2000). A robotic reconnaissance and surveillance team. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 501–507.

[Sastry, 1999] Sastry, S. (1999). *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag.

[Simeon et al., 2002] Simeon, T., Leroy, S., and Laumond, J.-P. (2002). Path coordination for multiple mobile robots: a resolution complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49.

[Smith and Cheeseman, 1986] Smith, R. C. and Cheeseman, P. (1986). On the representation and estimation of spacial uncertainty. *International Journal of Robotics Research*, 5(4):56–68.

[Smith et al., 2001] Smith, T. R., Hanssmann, H., and Leonard, N. E. (2001). Orientation control of multiple underwater vehicles. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4598–4603.

[Song and Kumar, 2002] Song, P. and Kumar, V. (2002). A potential field based approach to multi-robot manipulation. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 1217–1222.

[Soreson, 1966] Soreson, H. W. (1966). *Advances in Control Systems*. Academic Press.

[Spletzer et al., 2001] Spletzer, J., Das, A., Fierro, R., Taylor, C., Kumar, V., and Ostrowski, J. (2001). Cooperative localization and control for multi-robot manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 631–636.

[Stiwell and Bishop, 2001] Stiwell, D. J. and Bishop, B. E. (2001). A framework for decentralized control of autonomous vehicles. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2358–2363.

[Stroupe et al., 2001] Stroupe, A. W., Martin, M. C., and Balch, T. (2001). Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 1092–1098.

[Sudsang and Ponce, 1998] Sudsang, A. and Ponce, J. (1998). On grasping and manipulating polygonal objects with disc-shaped robots in the plane. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2740–2746.

[Sudsang and Ponce, 2000] Sudsang, A. and Ponce, J. (2000). A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 1068–1075.

[Sudsang et al., 1999] Sudsang, A., Ponce, J., Hyman, M., and Kriegman, D. J. (1999). On manipulating polygonal objects three 2-DOF robots in the plane. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2227–2234.

[Sugar and Kumar, 1998] Sugar, T. and Kumar, V. (1998). Decentralized control of cooperating mobile manipulators. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 2916–2921.

[Tan and Lewis, 1997] Tan, K. and Lewis, M. A. (1997). Virtual structures for high-precision cooperative mobile robot control. *Autonomous robots*, 4(4):387–403.

[Tanner et al., 2002] Tanner, H. G., Kumar, V., and Pappas, G. J. (2002). The effect of feedback and feedforward on formation ISS. In *Proc.IEEE Int'l Conf. on Robot. and Automat.*, pages 3448–3453.

[Thrun, 2001] Thrun, S. (2001). An online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363.

[Thrun et al., 2000] Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2000). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141.

[Toh et al., 2002] Toh, C.-K., Delwar, M., and Allen, D. (2002). Evaluating the communication performance of an ad hoc wireless network. *IEEE Transactions on Wireless Communications*, 1(3):402–414.

[Udapa, 1977] Udapa, S. (1977). Colision detection and avoidance in computer controlled manipulators. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[van Nieuwstadt and Murray, 1998] van Nieuwstadt, M. J. and Murray, R. M. (1998). Real time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, 8(11):995–1020.

[Varaiya, 1993] Varaiya, P. (1993). Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207.

[Vicsek et al., 1995] Vicsek, T., Czirok, A., Jacob, E. B., Cohen, I., and Schochet, O. (1995). Novel type of phase transitions in a system of self-driven particles. *Physical Review Letters*, 75:1226–1229.

[Wang, 1989] Wang, P. K. C. (1989). Navigation strategies for multiple autonomous mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 486–493.

[Wang and Kumar, 2002a] Wang, Z. and Kumar, V. (2002a). A decentralized test algorithm for object closure by multiple cooperating mobile robots. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*.

[Wang and Kumar, 2002b] Wang, Z. and Kumar, V. (2002b). Object closure and manipulation by multiple cooperative mobile robots. In *Proceedings of the IEEE International Conference on Robotics Automation*, pages 394–399.

[Xue and Kumar, 2003] Xue, F. and Kumar, P. R. (2003). The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, (to appear).

Appendix A

Multi-Robot Platforms

This appendix describes the multi-robot platforms used to validate our methodology. The appendix is divided in two main sections, one for each multi-robot team used in the thesis.

A.1 GRASP Lab. Platform

Most of our experiments were developed at General Robotics, Automation, Sensing and Perception (GRASP) Laboratory of the University of Pennsylvania. Each GRASP robot is a car-like robot equipped with omnidirectional cameras as their primary sensors (see Figure A.1). Also, a calibrated overhead camera is used to localize the robots in the environment when this is necessary. The communication among the robots relies on IEEE 802.11b wireless networking.

Each mobile robot is constructed using commercial radio controlled trucks. Each truck has a servo controller for steering and a proportional speed controller for forward/backward motion. Since the robots do not have any kind of proprioceptive sensor such as odometry, the control loop is closed by vision or infrared sensors (available only in the newest version of the robot). Three versions of the system are available so far:

Version 1 – In the first version all the computation is done in a remote

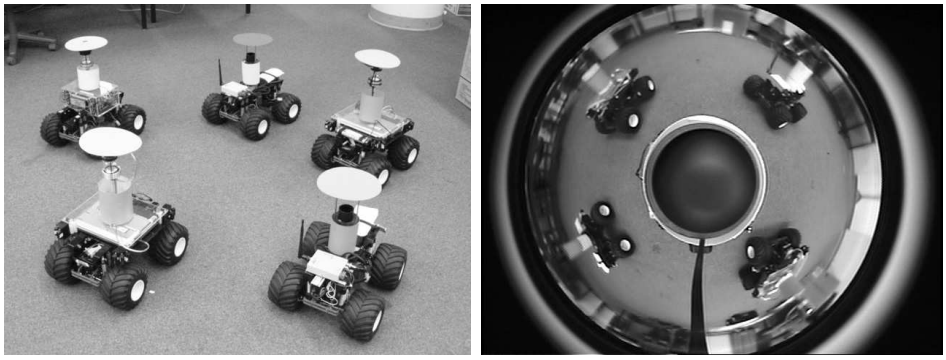

Figure A.1: The GRASP Lab. robots (left) and a sample image from an omnidirectional camera (right).

computer. Each robot is equipped with a wireless video transmitter and a receiver for actuation. Images from the omnidirectional system are transmitted to remote computers equipped with frame-grabbers. These computers calculate the control signals, and transmit the commands back to the robots.

Version 2 – The second version of the robots carries a customized onboard computer with IEEE 802.11b and frame-grabber cards. Image processing and control are executed on the robot.

Version 3 – The newest robots are based on laptop computers with IEEE 1394 (fire-wire) ports and IEEE 802.11b wireless networking. They are equipped with IEEE 1394 cameras and 12 infrared sensors.

The three versions of the robots are able to work together in indoor environments. However, processing and communication issues make the first version difficult to work outdoors. In our experiments we have worked with a team of five robots, where 2 robots are of the first version, 1 robot is of the second version, and the other 2 are of version 3.

The robots' control system is implemented using Microsoft Visual C++ under Microsoft Windows NT/2K/XP. Except for a few differences in the sensor's and actuator's drivers, all versions of the robots use the same soft-

ware. Just as an example of the robots capabilities, the slowest robot computer (Pentium III, 850 MHz, NT) is able to operate at a rate of 15 frames per second when it is running a typical localization and control algorithm as the ones presented in the other chapters of this text. The fastest computer (Pentium IV, 2 GHz, 2K) works at 30 frames per second running the same algorithm.

A.1.1 Vision system

This section describes both the onboard and external vision systems used by the robots as their only sensors.

Omnidirectional vision system

Each robot vision system is constituted by a parabolic mirror and a color CCD camera forming a omnidirectional camera. These cameras are very attractive sensors for robotics because they provide 360^o field of view. One of the primary advantages of such catadioptric systems is that they provide a single effective point of projection. This means that every point in the omnidirectional image can be associated with a unique ray through the focal point of the camera. Figure A.2 shows a cartoon with a longitudinal cut of the camera, which explains how a point in the world is transformed in a pixel in the image. By this figure it is also easy to derive an equation that relates a point in the image with a point in the space. Before we proceed with this derivation we must make some assumptions:

1. The radius of the mirror and the projection of its center in the image are available. This information can be obtained by calibration [Geyer and Daniilidis, 2002].

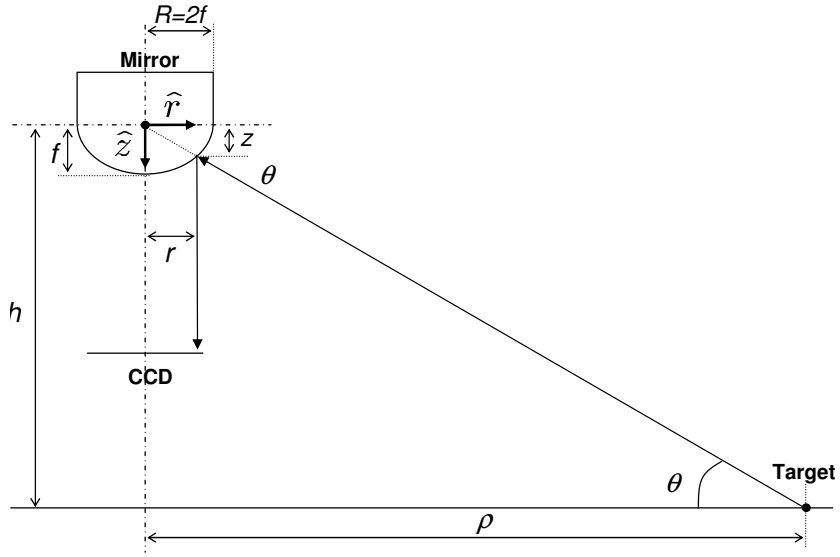

Figure A.2: Transformation from the image plane to ground plane using an omnidirectional camera.

2. The equation for the curvature of the mirror is known. This information is in general given by the manufacturer of the mirror. In our case this function is given by:

$$z = \frac{R^2 - r^2}{2R}\,, \tag{A.1}$$

where R is the radius of the mirror and r is the distance from the center of the mirror to the projection of the target in the image plane.

3. The hight, h, between the focus of the mirror and the target is specified.

Given these assumptions we can write (refer to Figure A.2):

$$\tan\theta = \frac{z}{r} = \frac{h}{\rho}\,,$$

that yields:

$$\rho = \frac{h\,r}{z}\,. \tag{A.2}$$

Substituting (A.1) in (A.2) we have:

$$\rho = \frac{2\,R\,h\,r}{R^2 - r^2}\,, \tag{A.3}$$

which gives a range measurement from the center of the camera to every point in the world that is below the focal point of the mirror.

Another information of interest is the bearing of a target. Since the center of the mirror in the image, (u_0, v_0), is known, bearing information can be directed computed as:

$$\phi = \arctan\frac{v - v_0}{u - u_0}\,, \tag{A.4}$$

where (u, v) are the coordinates of the target projection in the image plane.

Notice that, in real time, the only information necessary to compute range and bearing of targets are their projections in the image plane. In our case, in order to facilitate the vision processing we instrument the targets with color markers. Thus, in order to be easily identified by the other robots, each robot was fitted with a colored collar that yields a 360^o target.

Observe that the camera is symmetric to rotations and, at first, all targets situated at any distance from the center of camera could be detected. However, a big limitation of the omnidirectional cameras is that their resolution decrease very fast with the distance of the targets. At 2 m, for instance, the projection of an observed robot in the image plane is only one pixel in size, what suggest that for distances grater than that, the robots are completely blind. Since this is a characteristic of the hardware, we have introduced this limitation as a constraint for our motion control algorithms as shown in Chapter 5.

The quality of range measurements is also very compromised with the distance from the center of the camera. Observe in Figure A.2 that the

range equation can be written as:

$$\rho = \frac{h}{\tan\theta}\,. \tag{A.5}$$

If we differentiate Equation (A.5) in θ we have:

$$\frac{\partial\rho}{\partial\theta} = \frac{-h}{\sin^2\theta}\,. \tag{A.6}$$

By squaring both sides of Equation (A.6) and taking the expectation value of the result we obtain an approximation of range covariance, σ_ρ^2, in function of the covariance of θ, σ_θ^2 as:

$$\sigma_\rho^2 \approx \frac{h^2}{\sin^4\theta}\,\sigma_\theta^2\,. \tag{A.7}$$

This indicates that σ_ρ^2 is proportional to σ_θ^2, but this linear relationship is dictated by θ, which changes with the distance. Notice that, even if we have a small and constant variance for θ, what can be done by proper vision algorithms and structured targets, the variance σ_ρ^2 will increase with the the power of four of θ (assuming $\theta \approx \sin\theta$ for small θ). Thus, the farther is the target and consequently, the smaller is θ, the greater is σ_ρ^2. In our experiments we have assumed that ρ is proportional to θ and assumed that σ_ρ^2 increases with power of four of ρ. Experimentally we have found that the smallest variance is around $0.5\,\mathrm{cm}^2$, which increases to $16\,\mathrm{cm}^2$ at $\rho = 2\,\mathrm{m}$. Regarding to bearing measurement, ϕ, it can be seen by Equation (A.4) that its quality, measured by the covariance σ_ϕ^2, only depends on the vision processing. Then, we have assumed that σ_ϕ^2 is constant and equals to $0.1\,\mathrm{deg}^2$ when the target is inside the camera field of view. Actually, as verified experimentally, the covariance of ϕ decreases with ρ, because the smaller is the target projection,

the more accurate is the vision processing. Since this is a small variation, this observation was not considered in our experiments.

Overhead video system

In order to globally localize the robots in the environment an overhead camera is used. In our particular case the camera is located in one of the corners of the laboratory with a tilt angle of approximately 45^o, chosen in order to maximize its field of view. The transformation between pixel coordinates, (u, v), in the image plane and actual positions, (x, y), in the environment, is carried out using a homographic transformation. Thus,

$$\begin{bmatrix} s\,x \\ s\,y \\ s \end{bmatrix} = \mathbf{H} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{A.8}$$

where the homography matrix $\mathbf{H}$, is obtained via calibration. This matrix includes the intrinsic camera parameters (focus length, lens distortion, etc.) the extrinsic ones (position and orientation), and other constants such the robots height. $\mathbf{H}$ is constructed by comparing at least four known points in the environment with their projections on the image. The only assumption is that all points in the environment are in the same plane, forcing the transformation to be 2D to 2D. In order to comply with this requirement, a flat floor in the laboratory was considered and the color markers in the robots were all positioned at the same hight from the ground.

The quality of the pose estimation computed using the homography matrix can be estimated by the covariance matrix of this transformation. The analytical computation of this matrix would involve the computation of the Jacobian of the intrinsic and extrinsic camera parameters matrix. Still, we

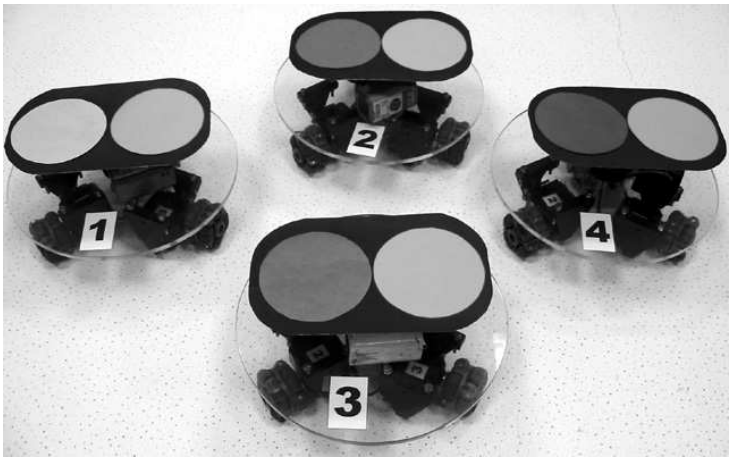

Figure A.3: The VERLab holonomic robots.

should be able to estimate the covariance of the vision process responsible for detecting the targets, which is not a simple operation. Thus, alternatively we have performed experimental tests and determined that the covariances for x and y are not greater than $2\,\mathrm{cm}^2$ in our 4 by 5 m workspace.

A.2 VERLab Platform

Some of the experiments of this thesis were executed in the Vision and Robotics Laboratory (VERLab) of the Federal University of Minas Gerais. The VERLab platform consists of a team of holonomic or omnidirectional robots. These remote controlled robots were inspired by the Palm Pilot's controlled ones presented in [Reshko et al., 2002]. A picture of the robots can be seen in Figure A.3

Each holonomic robot has three omnidirectional wheels mounted in a circular, 10 cm radius, plexiglass platform as shown in Figure A.4. Omnidirectional wheels have rollers that allow them to freely roll sideways but control the motion in the direction the wheel is pointing. Because of this, the holonomic robots can move in any direction. One standard servo, modified for

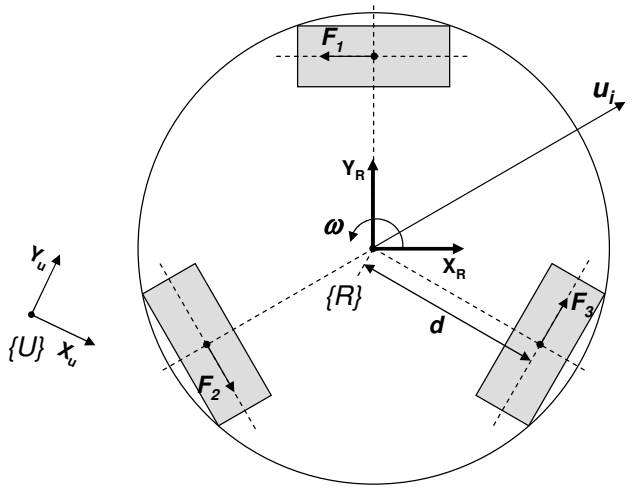

Figure A.4: Holonomic robot schematic. The shadowed rectangles represent the omnidirectional wheels.

rotation, drives each omnidirectional wheel. The three servos receive control signals from a remote computer through a pair transmiter/receiver. Digital signals from the computer are converted in the pulse-width-modulation (PWM) signals required by the transmitter using a microcontroller interface.

The robots have no local sensors and are localized in the environment using an overhead camera system similar to the one presented in Section A.1.1. The difference here is that each robot has two color markers in order to allow the estimation of its orientation. Regardless the omnidirectional nature of the robots, orientation is important in order to compute the control signals for the servos.

In order to understand how the control signals are computed, refer to Figure A.4. At first we are interested in making the robot follow the vector u_i computed with respect to a fixed reference frame $\{U\}$. The first step is then to transform u_i to the robot's reference frame $\{R\}$ through a simple rotation and translation. The rotation matrix is determined by the orientation of the robot, estimated by the overhead vision system. Once u_i is transformed to

the robot's frame, the linear velocity of each wheel is computed by projecting u_i in the unit vector perpendicular to each wheel axel (F_1, F_2 and F_3 in Figure A.4). Thus, the angular velocity of each wheel is computed as:

$$\omega_j = (u_i \cdot F_j)/r\,, \qquad 1 \leq j \leq 3\,, \tag{A.9}$$

where $F_1 = [-1, 0]^T$, $F_2 = [1/2, -\sqrt{3}/2]^T$, $F_3 = [1/2, \sqrt{3}/2]^T$, and r is the wheel radius. We can also impose an angular velocity ω to the robot by rewriting Equation (A.9) as:

$$\omega_j = (u_i \cdot F_j + d\,\omega)/r\,, \qquad 1 \leq j \leq 3\,, \tag{A.10}$$

where d is the wheel baseline as shown in Figure A.4. For the sake of simplicity, in our experiments we assume that the control signals are proportional to the wheel's angular velocity. In our previous work [Pereira et al., 2000] we had shown that this is not always the case because besides non-linearities, dynamics and delays are present. However, since the holonomic robots have small velocities when compared to the sensor's sampling rate, the closed loop is able to correct the small errors due to approximations. A deeper study about this issue is left as a future work.

As for the GRASP robots, the holonomic robots' control system is implemented using Microsoft Visual C++ under Microsoft Windows. The difference here is that up to four robots are controlled using the same machine. In a Pentium II, 300 MHz running Windows XP, the system runs at 10 frames per second when it is controlling four robots (extracting five colors). This is apparently sufficient given the maximum robot velocity of 7 cm/s. With this velocity and a typical homography matrix, the velocity of the robots in the camera plane is not greater than 70 pixels/s.

Appendix B

Collaborative Localization and Tracking

This appendix presents a localization and object tracking approach based on statistical operators and simple graph searching algorithms. The approach was implemented in the team of five car-like robots described in Appendix A and used in the experiments presented in previous chapters of this proposal. We start in Section B.1 with a brief introduction and some related work followed by the mathematical and graph modelling used in this appendix in Section B.2. Sections B.3 and B.4 present basic operators used in the graph algorithm described in Section B.5. Experiments other than the ones presented in the rest of the proposal are presented in Section B.6. Some concluding remarks are in Section B.7

B.1 Introduction

Robotic systems are in general equipped with several sensors. In this appendix we consider situations where sensors are placed on networked, co-operative mobile robots. Thus, our objective is to estimate in real time the position and orientation of a group of mobile robots using only information from their vision system. Additionally, we are also interested in simultaneously tracking a rigid unknown object. Our approach is based on the fact

that the combination of multiple simultaneous observations of the same object can provide information that is more complete, more accurate, and more robust when compared to a single observation.

The approach proposed here is closely related to those presented in [Roumeliotis and Bekey, 2002] and [Stroupe et al., 2001] in the sense that the robots have access to their teammates sensor data (or some related information) and combine such information with the one coming from its own sensors. In those papers, the robots use distributed sensing to improve self localization [Roumeliotis and Bekey, 2002] or target localization [Stroupe et al., 2001]. Both papers rely their methodologies on Kalman Filters. Papers [Spletzer et al., 2001] and [Das et al., 2002a] present solutions for the relative multi-robot localization problem by combining information exchanged by the robots using least square optimization. In this work we present a different approach for localization and object tracking based on statistical operators and simple graph searching algorithms. Furthermore, differently from the previous approaches, we formulate the problem in such a way localization and object tracking can be solved by the same algorithm. We also show how the advantages related to optimality of previous works can be easily incorporated in our methodology. As an example, we show how to incorporate an Extended Kalman Filter (EKF) [Balakrishnan, 1987] in order to improve object tracking.

B.2 Mathematical Modeling

Consider a planar world, $\mathcal{W} = \mathbb{R}^2$, occupied by a rigid polygonal object with m edges and a group $\mathcal{R} = \{R_1, R_2, \ldots R_n\}$ of n robots. The i^{th} robot R_i is represented by the configuration $q_i = (x_i, y_i, \theta_i)$. The object is described by its corner set $\mathcal{O} = \{O_1, O_2, \ldots O_m\}$ where each O_j is represented by the

configuration $o_j = (x_j, y_j)$.

The physical locations of the robots coupled with the characteristics of the hardware and the requirements of the sensing and control algorithms dictate the sensing network for the group of robots. This network can be represented by a graph. Thus, let $G = (\mathcal{V}, \mathcal{E}, \mathcal{Z}, \mathcal{P})$ be a sensing graph where $\mathcal{V} = \mathcal{R} \cup \mathcal{O}$ is the set of vertices, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges that represent the presence of measurements between two vertices, $\mathcal{Z}$ is the set of measurements, and $\mathcal{P}$ is a set of variances that represent the quality of those measurements. Observe that our graph is a directed graph. Then, if a vertex v_j has an incoming edge from a vertex v_i, it means v_i has sensing information about v_j. This edge will be represented by $e_{ij} = (v_i, v_j)$. Clearly in this case v_i has to be a robot ($v_i \in \mathcal{R}$) while v_j can be either a robot or an object corner ($v_j \in \mathcal{R} \cup \mathcal{O}$). Each element e_{ij} is associated with an element $z_{ij} \in \mathcal{Z}$. An element z_{ij} is a tuple (ρ_{ij}, ϕ_{ij}), composed by the range and bearing measurements of the j^{th} vertex in relation to the i^{th} vertex in i's reference frame. Each element z_{ij} is associated with one element $p_{ij} = (\sigma^2_\rho, \sigma^2_\phi) \in \mathcal{P}$, where σ^2_ρ and σ^2_ϕ are the variances of ρ_{ij} and ϕ_{ij} respectively. The representation of measurements by range and bearing came naturally due to the omnidirectional vision systems used by our robots as sensors. In these systems the coordinates can be directly measured and their quality directly estimated (see Apendix A). Observe that we do not assume any kind of proprioceptive information such as robot's velocity and acceleration. Figure B.1 shows an example of our graph modelling.

B.3 Measurements Transformation

In the previous section we have assumed that the variables measured by each robot are target's range and bearing. This representation is very

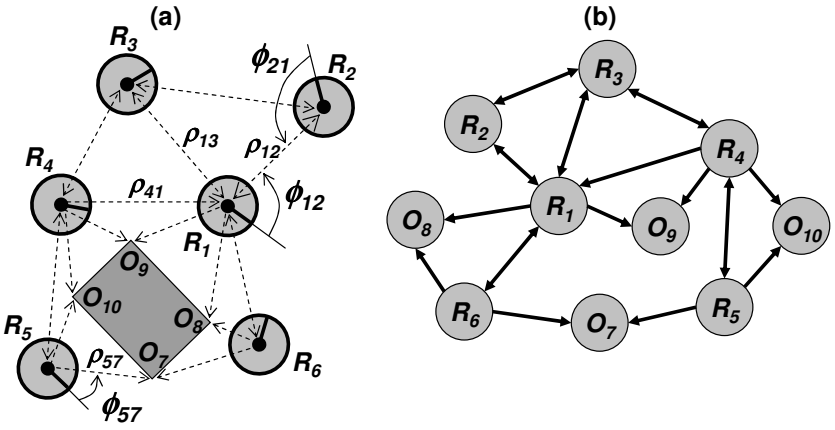

Figure B.1: (a) – A group of robots localizing and tracking a rectangular object; and (b) – a sensing graph for this snapshot.

convenient for estimation of robots' orientations but needs to be converted in order to estimate other variables such as robots' positions. Observe in Figure B.2(a) that ρ_{ij} and ϕ_{ij} can be converted to x_{ij} and y_{ij}, which are R_j's position in R_i's reference frame as:

$$\begin{aligned} x_{ij} &= f(\rho_{ij}, \phi_{ij}) = \rho_{ij} \cos(\phi_{ij}) \\ y_{ij} &= g(\rho_{ij}, \phi_{ij}) = \rho_{ij} \sin(\phi_{ij}) \ . \end{aligned} \tag{B.1}$$

In order to transform the associate covariance matrix of ρ_{ij} and ϕ_{ij} and obtain a covariance matrix for x_{ij} and y_{ij}, we will use the Jacobian of the transformation as proposed in [Smith and Cheeseman, 1986]. This is an approximation that works very well when the variables can be represented by unbiased normal distributions with small standard deviations. We also assume that ρ_{ij} and ϕ_{ij} are independent (although it is not required for the methodology) and consequently we can write the covariance matrix of the

robots measurements as:

$$\mathbf{P}_{\rho\phi} = \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} . \tag{B.2}$$

The Jacobian of the transformation, $\mathbf{J}$, relates the deviation of original and transformed variables as:

$$\begin{aligned} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} &= \mathbf{J} \begin{bmatrix} \Delta\rho \\ \Delta\phi \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial \rho} & \frac{\partial f}{\partial \phi} \\ \frac{\partial g}{\partial \rho} & \frac{\partial g}{\partial \phi} \end{bmatrix} \begin{bmatrix} \Delta\rho \\ \Delta\phi \end{bmatrix} \\ &= \begin{bmatrix} \cos(\phi) & -\rho\sin(\phi) \\ \sin(\phi) & \rho\cos(\phi) \end{bmatrix} \begin{bmatrix} \Delta\rho \\ \Delta\phi \end{bmatrix} . \end{aligned} \tag{B.3}$$

If we multiply both sides of Equation (B.3) by the respective transposes and take the expectation matrix of the result we have the transformed covariance matrix as:

$$\mathbf{P}_{xy} = \mathbf{J}\,\mathbf{P}_{\rho\phi}\,\mathbf{J}^T , \tag{B.4}$$

which is a matrix of the form:

$$\mathbf{P}_{xy} = \begin{bmatrix} \sigma_x^2 & \alpha\,\sigma_x\sigma_y \\ \alpha\,\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} , \tag{B.5}$$

where σ_x^2 and σ_y^2 are covariances along x and y respectively and α is their correlation coefficient.

Together with x_{ij} and y_{ij}, another variable of interest is the robots' relative orientation, θ_{ij}. Although it can not be estimated by a single robot measurement, if two robots exchange their bearing measurements, the rela-

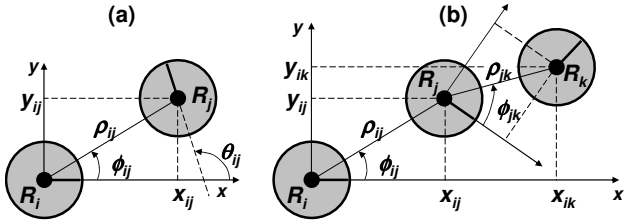

Figure B.2: (a) – Local transformation of variables; (b) – Sequential transformation.

tive orientation can be estimated as:

$$\theta_{ij} = \phi_{ij} - \phi_{ji} + \pi \,, \tag{B.6}$$

and its variance as:

$$\sigma^2_{\theta_{ij}} = \sigma^2_{\phi_{ij}} + \sigma^2_{\phi_{ji}} \,. \tag{B.7}$$

We are also interested in transforming variables measured by one robot to another robot's reference frame. In Figure B.2(b), for example, it could be interesting for R_i to have the relative position and orientation of R_k even when they cannot sense each other. In this specific situation R_i can sense R_j which can sense R_k. Therefore, we want to transform R_j's measurements to R_i's reference frame.

Assuming positions $(x_{jk},\, y_{jk})$ and $(x_{ij},\, y_{ij})$, and orientations θ_{jk} and θ_{ij} respectively of R_k in relation to R_j and R_j in relation to R_i are available, the expected values of x_{ik}, y_{ik} and θ_{ik} are given by a well known frame transformation:

$$\begin{aligned} x_{ik} &= x_{ij} + x_{jk}\cos(\theta_{ij}) - y_{jk}\sin(\theta_{ij}) \\ y_{ik} &= y_{ij} + x_{jk}\sin(\theta_{ij}) + y_{jk}\cos(\theta_{ij}) \\ \theta_{ik} &= \theta_{ij} + \theta_{jk} \,. \end{aligned} \tag{B.8}$$

Repeating the procedure explained before we obtain the Jacobian matrix of the transformation as:

$$\mathbf{J}=\begin{bmatrix} 1 & 0 & -x_{jk}\sin(\theta_{ij}) - y_{jk}\cos(\theta_{ij}) & \cos(\theta_{ij}) & -\sin(\theta_{ij}) & 0 \\ 0 & 1 & x_{jk}\cos(\theta_{ij}) - y_{jk}\sin(\theta_{ij}) & \sin(\theta_{ij}) & \cos(\theta_{ij}) & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \tag{B.9}$$

and the covariance matrix of the measurements of R_j in R_i's reference frame as:

$$\mathbf{P}_{ik} = \mathbf{J} \begin{bmatrix} \mathbf{P}_{ij} & \alpha\mathbf{P}_{ij}\mathbf{P}_{jk} \\ \alpha\mathbf{P}_{ij}\mathbf{P}_{jk} & \mathbf{P}_{jk} \end{bmatrix} \mathbf{J}^T , \tag{B.10}$$

where α is a matrix that correlates $\mathbf{P}_{ij}$ and $\mathbf{P}_{jk}$.

B.4 Measurements Combination

In the previous section we have shown how to transform the robots' measurements and how to estimate their covariance matrices. In this section we will see how to combine such information in order to have pose estimation of both robots and targets.

As proposed in [Smith and Cheeseman, 1986], by using the Kalman filter equations for static-state estimation, if two estimates $\hat{q}_1 = [x_1\ y_1\ \theta_1]^T$ and $\hat{q}_2 = [x_2\ y_2\ \theta_2]^T$ of the same variable q, are expressed in the same reference frame, and have covariance matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, then a better estimate $\hat{q}$ of q can be obtained as:

$$\begin{aligned} \mathbf{K} &= \mathbf{P}_1 (\mathbf{P}_1 + \mathbf{P}_2)^{-1} \\ \hat{q} &= \hat{q}_1 + \mathbf{K}(\hat{q}_2 - \hat{q}_1) \\ \mathbf{P} &= \mathbf{P}_1 - \mathbf{K}\mathbf{P}_1 , \end{aligned} \tag{B.11}$$

where $\mathbf{K}$ is the Kalman gain and $\mathbf{P}$ is the resulting covariance matrix. It is easy to verify that in the case of independent one dimension measurements these equations reduce to:

$$\begin{aligned} \hat{\beta} &= \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\,\hat{\beta}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\,\hat{\beta}_2 \\ \sigma^2 &= \frac{\sigma_2^2\,\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\,, \end{aligned} \tag{B.12}$$

where $\hat{\beta}_1$ and $\hat{\beta}_2$ are the estimates and σ_1^2 and σ_2^2 are their auto-covariances. These one dimensional equations are very useful and can be used, for example, in order to merge two range values measured by two neighboring robots before their transformation in x and y using Equation (B.1).

The previous equations are very direct and can combine any number of estimates if they can be paired. It's well know, however, that the Kalman filter is an iterative way to solve a weighted least squares problem since its equations are based on the minimization of the sum of the squares of the errors (innovations). In this way, if more than two measurements are available, any general weighted least squares method (including iterative and more efficient ones) can be used in order to produce the same results obtained when several applications of Equation (B.11) are made. Thus, if all measurements are grouped into a linear system as:

$$\mathbf{A}\;\hat{q} = \mathbf{q}\,, \tag{B.13}$$

a closed formula for the weighted least squares method is given by:

$$\hat{q} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{q}\,, \tag{B.14}$$

where $\mathbf{q} = [\hat{q}_1\ \hat{q}_2\ \ldots\ \hat{q}_k]^T$, $\hat{q}_i \in \mathbb{R}^3$, is the vector of measurements to be

combined and $\mathbf{W} = \text{diag}([\mathbf{P}_1^{-1}\ \mathbf{P}_2^{-1}\ \ldots \mathbf{P}_k^{-1}])$, $\mathbf{P}_i \in \mathbb{R}^3$ is the weight matrix. The covariance matrix of the result is given by:

$$\mathbf{P} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\ . \tag{B.15}$$

As an example, consider the combination of three one dimensional measurements. The linear system can be posed as:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \hat{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{bmatrix}\ , \tag{B.16}$$

and the weighted least square problem can be solved as:

$$\hat{\beta} = \left[\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_2^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_3^2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & 0 \\ 0 & \frac{1}{\sigma_2^2} & 0 \\ 0 & 0 & \frac{1}{\sigma_3^2} \end{bmatrix} \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_3 \end{bmatrix},$$

which results in:

$$\hat{\beta} = \frac{\sigma_2^2\sigma_3^2\hat{\beta}_1 + \sigma_1^2\sigma_3^2\hat{\beta}_2 + \sigma_1^2\sigma_2^2\hat{\beta}_3}{\sigma_1^2\sigma_2^2 + \sigma_2^2\sigma_3^2 + \sigma_1^2\sigma_3^2}\,. \tag{B.17}$$

One can easily verify that Equation (B.17) can also be obtained by combining $\hat{\beta}_1$ and $\hat{\beta}_2$ using Equation (B.12) and combining the result with $\hat{\beta}_3$ using the same equation. Thus, observe that for linear systems, the recursive application of Equation (B.11) and the least squares methodologies are equivalent and provide the same results. However, Equation (B.14) cannot be directly applied for non-linear systems because, without initial values for the variables, it is not possible to compute a complete linear approximation of the system using the Jacobian. In these cases a recursive combination

using Equation (B.11) on the locally linearized systems is necessary.

In the discussion above we are assuming that data to be combined are always measurements of the same variable estimated in the same reference frame. When measurements are made in different frames, the transformation steps discussed in the previous section must be used. However, when linear combinations of the variables of interest are available they can be used directly in the least-squares method provided that **A** in Equation (B.13) is adequately designed (it also can be used in the Kalman filter with a few modifications). As an example, consider we are trying to compute better estimates of θ, α and β and the available measurements are $\hat{\theta}$, $\hat{\beta}$, $\widehat{(\alpha-\beta)}$, $\widehat{(\theta-\beta)}$ and $\widehat{(\alpha-\theta)}$. The corresponding linear system is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \hat{\theta} \\ \hat{\beta} \\ \widehat{(\alpha-\beta)} \\ \widehat{(\theta-\beta)} \\ \widehat{(\alpha-\theta)} \end{bmatrix} \tag{B.18}$$

that can be solved using Equation (B.14).

B.4.1 Dynamical Systems

So far we have considered the estimation of variables that could be correlated by their covariance matrices but at first are not correlated in time. Thus, using the previous equations in order to combine the robots' measurements implies in estimating, in an optimal way, the pose of robots and object in a single snapshot or instant of time. In order to consider the system dynamics and also the physical correlation between the variables, the dynamic version of the Kalman filter can be used. In this case, other sensor infor-

mation such as robots' velocity and acceleration can be used to improve the robots localization. We will use the non-linear version of the Kalman Filter in order to improve object tracking. Since this is a very well known tool, it is not discussed here in details. The theory behind the filter along with its equations can be found in [Soreson, 1966].

B.5 Localization Approach

Our localization approach assumes that each robot has a unique identification (ID) both for communication and sensing. At first, we also assume object corners have distinct sensing IDs, but this assumption can be relaxed if simple heuristics based on pattern classification, such as the one presented in [Pereira et al., 2002a], is used to solve the problem of associating robots measurements. Our approach is centralized in the sense that each robot collects sensing information from other robots and combines this information using its own computational resources. Thus, given the previous background, localization in a network of robots can be addressed by combining a series of operations that involves transformations and combinations.

We have defined four basic operations in this appendix: (i) a transformation from the robots measurements $z_{ij} = (\rho_{ij}, \phi_{ij})$ to the target coordinate $q_{ij} = (x_{ij}, y_{ij})$ that we represent here by a superscript t (Equation (B.1)); (ii) a combination/transformation from the robots measurements z_{ij} and z_{ji} to robot pose $q_{ij} = (x_{ij}, y_{ij}, \theta_{ij})$, that will be denoted by "$\circ$"(Equation (B.6) for estimating θ, (B.12) for combining two ρs, and (B.1) for coordinate transformation); (iii) a transformation from the measurements of one robot to another reference frame, that will be denoted by "$\vee$"(Equation (B.8)); and (iv) a combination of two measurements in the same reference frame that will be denoted by "$\wedge$"(Equation (B.11)).

Assuming that each robot has its own sensing data and the information collected from the other robots organized in a graph similar to the one presented in Figure B.1, the localization can be performed by using the previous operators in a graph searching algorithm similar to Breadth First Search (BFS). The BFS algorithm visits, only once, all nodes of a tree by visiting all the nodes at the same depth before going deeper. Here, since we are not considering trees, the nodes can be visited more than once. Thus, if there is more than one path between the root of the graph and a specific node, all these paths will be used. Thus, the first time a vertex is visited its position is estimated. From then on, each time a node is reached its previously estimated pose is combined with the pose recently estimated using this new path.

Because we allow a node to be visited more than once, theoretically, the algorithm could enter in loops. Loops cause situations of interdependence where, for example, the pose of v_i can be computed using information from v_j and the pose of v_j can be computed using information from v_i. In order to avoid this problem, the original graph is transformed into a directed graph where loops are removed. The new graph is similar to a tree (however, it is not a tree) with the root being the robot chosen as the origin. Loops are avoided by removing edges between robots with the same depth. Because objects do not have measurements they never create loops and their edges are never deleted. The same occurs with unidirectional edges between two robots. A bidirectional edge between two robots of different depths may also create loops. However, situations like this are prevented because the algorithm treats the graph as a tree and never moves towards the root. Exceptions are made with unidirectional edges that are always followed independently of the direction it points to.

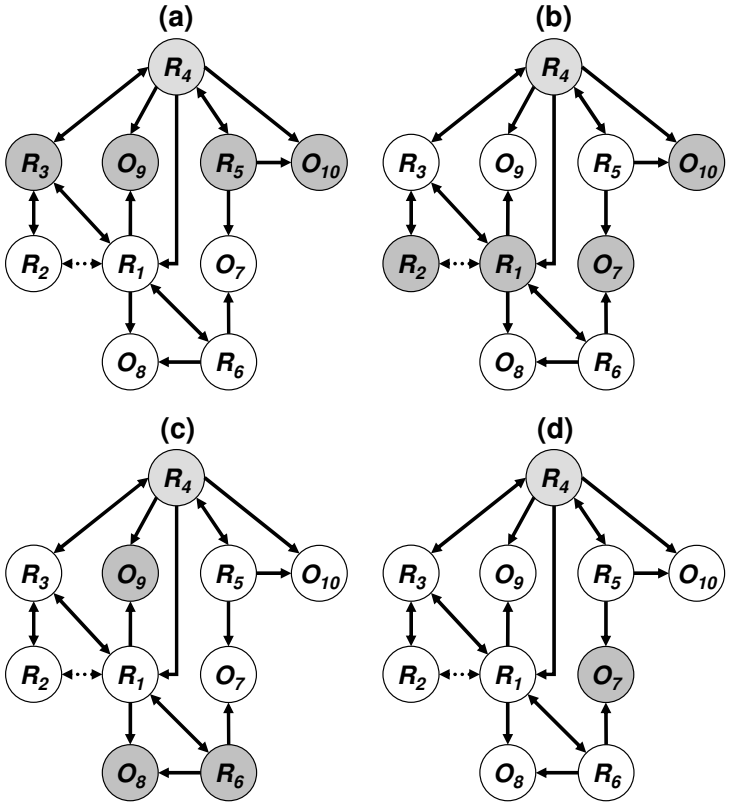

Figure B.3: Four steps of the localization algorithm. All robots and objects are localized in relation to R_4. Sensor information between R_1 and R_2 (dotted edge) is eliminated in order to avoid a loop. In step (a) R_3, R_5, O_9 and O_{10} are localized. In step (b) R_1, R_2 and O_7 are localized and O_{10} position is updated with information from R_5. The algorithm proceeds until all information is used or a determined graph depth is achieved.

Figure B.3 shows an example of the derived graph and four steps of the algorithm for the situation presented in Figure B.1 when R_4 is the origin. The vertices are shaded gray when their poses are estimated or upgraded. Notice how the edges between R_1 and R_2 (dotted line) were eliminated in order to avoid a loop. Observe also that edge $e_{41} = (R_4, R_1)$ was used only in the second level of the graph since the absence of e_{14} prevents the computation of θ_{41}. After this variable is estimated by another path, e_{41} is used in order to improve x_{41} and y_{41} estimates.

If we use the previous algorithm to localize O_9, for example, the sensor combination is expressed using the previously defined operators as:

$$\hat{q}_{49} = z^t_{49} \wedge \left\{ \left([(z_{43} \circ z_{34}) \vee (z_{31} \circ z_{13})] \wedge z^t_{41} \right) \vee z^t_{19} \right\} . \tag{B.19}$$

This is actually the combination of R_4 and R_1 measurements, but because R_4 cannot localize R_1 directly, a path passing by R_3 needed to be used.

One of the issues with the previous algorithm is that some of the edges (such as e_{12} and e_{21} in the previous example) are not used in the robots' and object's pose estimation. In order to avoid wasting useful information one could divide the algorithm into two parts assuming the robot's orientations and positions can be computed separately. This idea was previously used in [Das et al., 2002a]. The two parts are: (i) – estimation of the robot's orientations using the same algorithm; and (ii) – estimating the robots and targets positions using a weighted least squares method that assumes that the positions of two robots (R_i and R_j) in a third robot (R_k) coordinate frame are linearly related by:

$$x_{ki} - x_{kj} = \rho_{ij} \cos(\theta_{ki} + \phi_{ij}) \tag{B.20}$$

$$y_{ki} - y_{kj} = \rho_{ij} \sin(\theta_{ki} + \phi_{ij}) . \tag{B.21}$$

where θ_{ki} and ϕ_{ij} are assumed to be known. The necessity of knowing θ_{ki} explains the division into two parts. Besides computing the orientations, the first part of the algorithm is responsible for computing the number of robots connected to the network and consequently defining the variables to be computed. Some edges are still wasted in this part of the algorithm, what is reasonable under the observation that bearing measurements tend to be much better than the range ones.

Another improvement can be achieved by using a dynamic Kalman Filter in order to estimate the object's position. Because the object is rigid, if a model that relates its corners is used, tracking can be performed even when some corners are not seen by the robots. A simple discrete model where the dimensions of the object are not necessarily known is:

$$\begin{aligned} x_{j+1}(k+1) &= x_j(k) + vx_j\,T + d_j(k)\cos(\theta_j(k)) \\ y_{j+1}(k+1) &= y_j(k) + vy_j\,T + d_j(k)\sin(\theta_j(k)) \\ d_j(k+1) &= d_j(k) \\ \theta_j(k+1) &= \theta_j(k) + \omega(k) \\ vx_j(k+1) &= vx_j(k) \\ vy_j(k+1) &= vy_j(k) \\ \omega(k+1) &= \omega(k)\,, \end{aligned} \tag{B.22}$$

where vx_j and vy_j are the velocity components of O_j, d_j and θ_j are the size and orientation of the edge between O_j and O_{j+1}, T is the sampling time, and ω is the object angular velocity. This model considers that the object's velocity components are constant. Since this is not always true, during filter design, low values must be assigned to the variables that represent the confidence level of the three last lines of model. Because we are using a nonlinear model, an Extend Kalman Filter (EKF) is necessary. The measurements used in this filter will be x and y of each object corner relative to the origin robot. Thus, once the robots's pose are estimated, the transformation in Equations (B.8) and (B.10) are used in order to compute the vector of measurements and its covariance matrix. Note that the Kalman Filter introduces another step in the algorithm. Thus, the vertices of the graph relative to object corners can be removed in the localization step since sensor combination is now performed by the filter.

B.5.1 Centralized × Distributed

While the previous centralized approach works very well for a relative small group of robots, network issues such as traffic, delays and the lack of computational power in one single robot can pose significant problems when we are considering groups with tens or hundreds of robots. In these cases the use of distributed algorithms become mandatory. Then, we want a way to use the same algorithm and reduce both, the computation and bandwidth needed by decentralizing part of the processing.

We start noticing that, in general, mobile robots only need local information in order to perform a task. Thus, if each robot collects information from its immediate neighbors and combine these data locally, it has the information it needs most of the time. In the case where a robot can listen to every robot in the group (i.e., all robots are within its communication range) it could receive all the data but, for example, localize only the robots located within a certain distance (measured by the depth in the graph) from it. In the same communication configuration, the robot could also be more selective and choose to localize only the robots that can see a given object or location.

Another way to decentralize processing, which may be useful when ad-hoc networks are used and the robots cannot talk directly to each other, is localize only the neighboring robots (robots in the communication range). In this situation the robots do not need to work as routers for communication messages and much bandwidth is preserved. If global information is necessary, a robot may ask one or more of its neighbors for the localization information (not the raw data) they have and compute the other robots' (or a specific robot) position by using transformations in Equations (B.8) and (B.10). When real-time is not a constraint, this procedure can be used recursively until full

information for the whole team is available.

B.5.2 Global Localization

Thus far, only relative localization is considered once each robot computes the position of the others in its own reference frame. When measurements relative to a fixed frame in the world are available for the root robot, simple transformations such as those shown in Equations (B.8) and (B.10) are used in order to transform the relative estimates into world coordinates. This information can be also used in the least-squares methods with minor modifications in the matrices. Notice that if complete measurements about the root robot are available (x, y and θ), this information is sufficient to localize all connected robots in the world coordinates. When partial information is available (x and y for example) global information for at least two robots is necessary.

B.6 Experimental Results

This section presents experimental results with our team of five mobile robots. The only sensor the robots carry is an omnidirectional camera having as field of view a circle of approximately 1.5 m of radius as shown in Apendix A. Color markers are used in order to facilitate the identification of robots and object's corners. Videos of the experiments can be found in [Pereira, 2003].

Figure B.4 shows a snapshot of an experiment in which one robot (R_1) moves towards a target that is localized by another robot (R_2). In order to do so, this robot needs to localize the target through information broadcasted by the other robots. Ground-truth data, obtained using a calibrated overhead

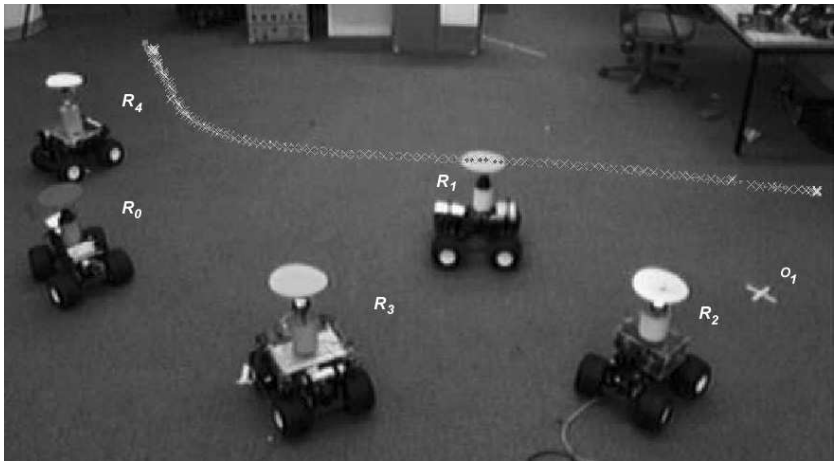

Figure B.4: R_1 moves towards O_1 based on information collected by R_2 and shared through the network.

camera (see Apendix A), for the same experiment is shown in Figure B.5. The dashed circles, which represent the cameras' field of view, show that R_1 can basically see one or two robots simultaneously. In this figure we also show the estimation of R_1's motion in R_0's reference frame (which coincides with the global reference frame) and R_1's actual trajectory as seen by the overhead camera.

Figure B.6 shows real data from an experiment where two perfectly localized robots are cooperating by tracking a square box using an EKF. In the beginning (Figure B.6), when some corners are far from the robots, the covariance of some of the corners are very large and the box format is wrong. Once the box is spined the ellipses of covariance reduced and the box format is improved (Figure B.6(b)).

Figure B.7 shows two snapshots of three robots tracking a triangular box using an EKF. R_0's reference frame is shown. Robots R_1 and R_2 are able to see the box corners while R_0 is only used in order to localize the other two. Even though R_0 cannot see the box corners, it is able to track the box

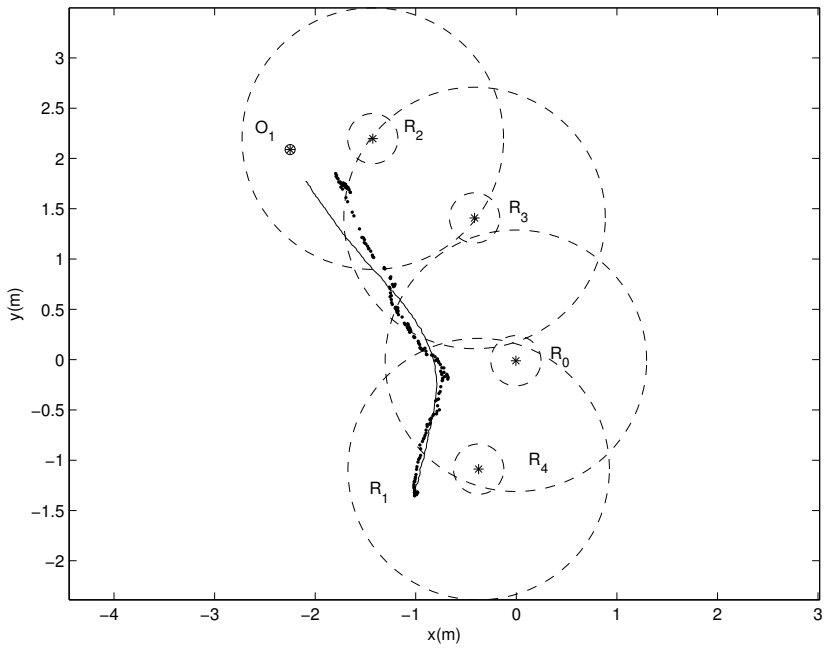

Figure B.5: Ground truth for the experiment shown in Figure B.4. The dots represent R_1's position estimated by R_0 and the continuous line the actual trajectory. The inner dashed circles represent the robot size and the outward ones represent the cameras field of view.

using information from its teammates. The snapshot of the right shows that when one robot goes blind the corners covariance increases but the box is still tracked.

Our last result shows how the robots are able to be globally localized if data from an external calibrated camera is used. Figure B.8 shows two images from the external camera used in order to localize the robots in the environment. Localization results in these two configurations plus an intermediary configuration (Configuration 2) in which R_4 can be seen by the camera, are shown in Table B.1. Observe that due to the information from other robots, robots that are not seen by the external camera can still be localized.

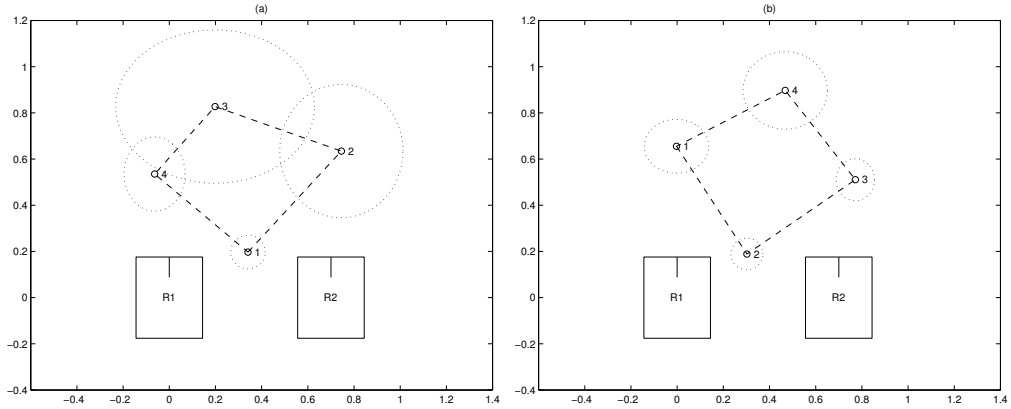

Figure B.6: (a) Initial and (b) final instants of a box tracking experiment. The dotted ellipses are the 3σ region of confidence.

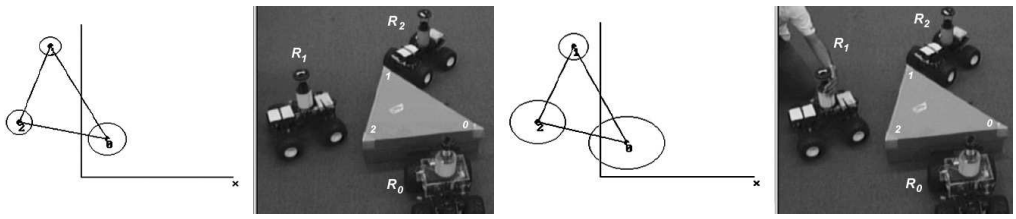

Figure B.7: Two snapshots of the experiment where three robots are tracking a triangular box. The ellipses are the 3σ region of confidence.

B.7 Concluding Remarks

We have presented a simple and efficient algorithm for localization and tracking in networks of robots based on basic statistical operators and graph algorithms. The algorithm uses well known statistical concepts in order to combine multi-robot information. A clear drawback of the algorithm is that it does not use information about the robots dynamics in order to improve on the robots localization. Actually, dynamics could be easily incorporated in the computation with a few modifications. The main reason why it was not done in this work is the lack of proprioceptive sensors, such as encoders, gyroscopes or accelerometers, which are important in order to integrate the dynamical models of the robots. If those sensors were available it would

Table B.1: Localization Results in three different configurations.

	Ground-Truth			Configuration 1			Configuration 2			Configuration 3		
R_i	x (m)	y (m)	θ ($^\circ$)	x (m)	y (m)	θ ($^\circ$)	x (m)	y (m)	θ ($^\circ$)	x (m)	y (m)	θ ($^\circ$)
0	0.26	0.32	68.0	0.25	0.32	70.8	0.25	0.32	71.5	0.25	0.31	71.1
1	0.99	2.04	-41.0	0.95	2.01	-44.8	1.07	1.87	-46.1	1.03	1.88	-46.6
2	1.97	1.29	125.0	1.87	1.31	122.2	1.88	1.31	122.11	1.88	1.31	122.2
3	0.17	1.48	40.0	0.12	1.48	38.9	0.12	1.48	40.7	0.12	1.48	40.7
4	1.64	0.20	-125.0	1.64	0.27	-135.7	1.63	0.24	-128.1	1.56	0.42	-94.5

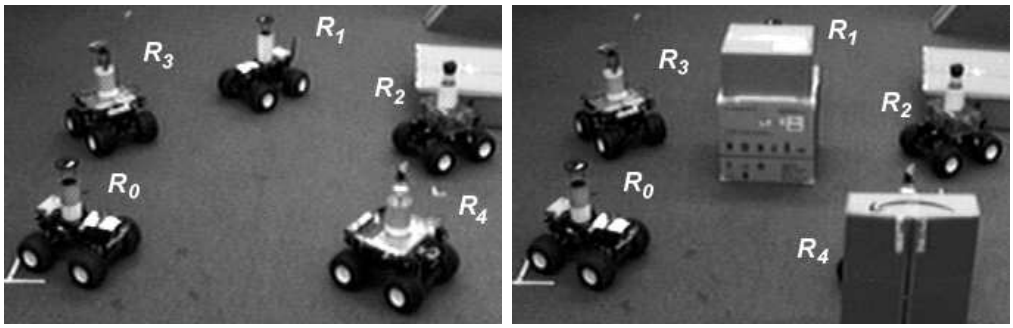

Figure B.8: Configurations 1 (left) and 3 (right) which localization results are shown in Table B.1.

make more sense to use a dynamical Kalman filter together with the current algorithm in order to improve the robots estimates.