MODELING AND CONTROL OF THE COORDINATED MOTION OF A GROUP OF AUTONOMOUS MOBILE ROBOTS

by

NUSRETTIN GULEC

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabanci University
Spring 2005

MODELING AND CONTROL OF THE COORDINATED MOTION
OF A GROUP OF AUTONOMOUS MOBILE ROBOTS

Nusrettin GULEC

APPROVED BY

Assoc. Prof. Dr. Mustafa UNEL .....
(Thesis Advisor)

Prof. Dr. Asif SABANOVIC .....
(Thesis Co-Advisor)

Assist. Prof. Dr. Kemalettin ERBATUR .....

Assoc. Prof. Dr. Mahmut F. AKSIT .....

Assist. Prof. Dr. Husnu YENIGUN .....

DATE OF APPROVAL: .....

to my beloved sister

&

my father

&

my mother

Biricik Ablama

&

Babama

&

Anneme

Autobiography

Nusrettin Gulec was born in Izmir, Turkey in 1981. He received his B.S. degree in Microelectronics Engineering from Sabanci University, Istanbul, Turkey in 2003. His research interests include coordination of autonomous mobile robots, control of nonholonomic mobile robots, sensor and data fusion, machine vision, visual servoing, robotic applications with PLC-SCADA systems.

The following were published out of this thesis:

- N. Gulec, M. Unel, A Novel Coordination Scheme Applied to Nonholonomic Mobile Robots, *accepted for publication in the Proceedings of the Joint* 44^{th} *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05), Seville, Spain, December 12-15, 2005.*

- N. Gulec, M. Unel, A Novel Algorithm for the Coordination of Multiple Mobile Robots, *to appear in LNCS, Springer-Verlag, 2005.*

- N. Gulec, M. Unel, Coordinated Motion of Autonomous Mobile Robots Using Nonholonomic Reference Trajectories, *accepted for publication in the Proceedings of the* 31^{st} *Annual Conference of the IEEE Industrial Electronics Society (IECON 2005), Raleigh, North Carolina, November 6-10, 2005.*

- N. Gulec, M. Unel, Sanal Referans Yorungeler Kullanilarak Bir Grup Mobil Robotun Koordinasyonu, *TOK'05 Otomatik Kontrol Ulusal Toplantisi, 2-3 Haziran 2005.*

Acknowledgments

I would like to express my deepest gratitude to Assoc. Prof. Dr. Mustafa Unel, who literally helped me find my way when I was completely lost - with that admirable research enthusiasm that has always enlightened me, specifically those eleven hours in front of the monitor that thought me lots, that invaluable insight saving huge time for my research - and on top of all, who had always been frank with me, which is the best to receive.

I would also like to acknowledge Prof. Dr. Asif Sabanovic, for that trust he had in me two years ago that made my way through today. Without him, neither would this thesis be completed, nor my graduate study could get started.

Among all members of the Faculty of Engineering and Natural Sciences, I would gratefully acknowledge Assist. Prof. Dr. Kemalettin Erbatur, Assoc. Prof. Dr. Mahmut F. Aksit and Assist. Prof. Dr. Husnu Yenigun for spending their valuable time to serve as my jurors.

I would also be glad to acknowledge Prof. Dr. Tosun Terzioglu, Prof. Dr. Alev Topuzoglu, Zerrin Koyunsagan and Gulcin Atarer for their never-ending trust and support against any difficulty I had throughout my life in Sabanci University.

Among my friends, who were always next to me whenever I needed, I would happily single out the following names; Burak Yilmaz, who is essentially the most caring person I know, Sakir Kabadayi, who has been the 'Big Brother' in my worst times, Izzet Cokal, whose presence around was a great relief, Ozer Ulucay, who is the purest person I ever met, Firuze Ilkoz, who has always supported me without question, Eray Korkmaz, whose friendship was stronger than anything, Onur Ozcan, who has been nothing but a sincere friend for more than three years now, Esranur

Sahinoglu, without whom I could never work for the last three months, Arda of Café Dorm for all supplies he provided, Khalid Abidi, who was ready to discuss anything whenever I needed, Dogucan Bayraktar and Celal Ozturk, in the absence of whom I could never conduct the experiments, Didem Yamak, the motivation of whom was the best to receive for two years, Can Sumer, with whom I shared those late-night talks and discussions, Borislav Hristov Petrinin, whose friendship and support is one of the best I have ever seen or had, Cagdas Onal, who has always surprised me with that amazing friendship, Mustafa Fazil Serincan, whose friendship always made me smile, and all others I wish I had the space to acknowledge in person: Kazim, Ertugrul, Ilker, Shahzad, Selim, Nevzat, ...

Very special thanks go to Didem Yamak and Onur Bolukbas, for utilizing each and every moment I looked for some tranquility during this thesis, especially Didem for that confidential support she provided, beyond logic, for my academic career.

Finally, I would like to thank my family for all that patience and support they provided through each and every step of my life.

MODELING AND CONTROL OF THE COORDINATED MOTION OF A GROUP OF AUTONOMOUS MOBILE ROBOTS

Nusrettin GULEC

Abstract

The coordinated motion of a group of autonomous mobile robots for the achievement of a coordinated task has received significant research interest in the last decade. Avoiding the collisions of the robots with the obstacles and other members of the group is one of the main problems in the area as previous studies have revealed. Substantial amount of research effort has been concentrated on defining virtual forces that will yield reference trajectories for a group of autonomous mobile robots engaged in coordinated behavior. If the mobile robots are nonholonomic, this approach fails to guarantee coordinated motion since the nonholonomic constraint blocks sideway motions. Two novel approaches to the problem of modeling coordinated motion of a group of autonomous nonholonomic mobile robots inclusive of a new collision avoidance scheme are developed in this thesis. In the first approach, a novel coordination method for a group of autonomous nonholonomic mobile robots is developed by the introduction of a virtual reference system, which in turn implies online collision-free trajectories and consists of virtual mass-spring-damper units. In the latter, online generation of reference trajectories for the robots is enabled in terms of their linear and angular velocities. Moreover, a novel collision avoidance algorithm, that updates the velocities of the robots when a collision is predicted, is developed in both of the proposed models. Along with the presentation of several coordinated task examples, the proposed models are verified via simulations. Experiments were conducted to verify the performance of the collision avoidance algorithm.

BIR GRUP OTONOM MOBIL ROBOTUN
KOORDINELI HAREKETININ MODELLENMESI VE KONTROLU

Nusrettin GULEC

Ozet

Bir grup otonom mobil robotun, verilen bir gorevi basarmak icin koordineli hareketi son on yilda onemli bir arastirma konusu olmustur. Robotlarin engellerle ve grubun diger elemanlariyla carpismalarinin engellenmesi onceki calismalarin da gosterdigi gibi, bu alandaki en temel problemlerden biridir. Onemli miktarda arastirma cabasi koordineli davranis icindeki bir grup otonom mobil robot icin referans yorungeler ortaya koyacak sanal kuvvetler tanimlama yonunde yogunlasmistir. Eger mobil robotlar holonom degillerse, holonom olmama kisitlamasi yanal yondeki hareketi engelleyecegi icin, bu yaklasim koordineli hareketi kesin olarak saglamayabilir. Bu tez calismasinda bir grup otonom holonom olmayan mobil robotun koordineli hareketini modelleme ve kontrol etme problemine, yeni bir carpisma engelleme algoritmasi da iceren, iki yeni yaklasim gelistirilmistir. Birinci yaklasimda, cevrimici carpismasiz yorungeler ortaya koyacak sanal kutle-yay-amortisor birimlerinden olusan bir sanal referans model kullanilarak, otonom holonom olmayan mobil robotlar icin yeni bir koordinasyon metodu gelistirilmistir. Ikinci yaklasimda ise, robotlar icin cevrimici referans yorungeler dogrusal ve acisal hizlari cinsinden olusturulmustur. Ayrica, onerilen iki modelde de bir carpisma ongoruldugu zaman robotlarin hizlarini guncelleyen yeni bir carpisma engelleme algoritmasi gelistirilmistir. Bazi koordineli gorev orneklerinin sunulmasiyla birlikte, onerilen modeller benzetimlerle dogrulanmistir. Carpisma engelleme algoritmasinin performansinin dogrulanmasi icin deneyler yapilmistir.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Science today is essentially about establishing models that mimic the behavior of real-life systems to be able to predict the outcome of certain events encountered in nature. Models for technical issues like electrical, mechanic, pneumatic and hydraulic systems as well as social issues like economic growth of countries and population growth of communities have been well-established and developed. However, subjects related to intelligent behavior observed in nature such as coordinated motion and coordinated task handling of social groupings along with the autonomous behavior of individual agents in those groups are still in the phase of research. Many studies have been directed towards understanding and modeling the way of biological systems, particularly humans and animals performing certain tasks together. A variety of scientific disciplines - such as artificial intelligence, mechatronics, robotics, computer science and telecommunications - deal with these problems from different aspects. For example, artificial intelligence researchers work on establishing a framework for the algorithms to be followed by each autonomous individual in the group to achieve coordinated motion of the entire group, while researchers in the area of telecommunications are interested in developing methods for efficient transfer of necessary data between the autonomous elements of the group.

The research effort towards modeling the coordinated behavior of natural groupings has triggered the studies on several other areas such as decentralized systems, distributed sensing, data fusion and mobile robot vision.

The following sections outline the basic concepts regarding the coordinated motion of a group of autonomous mobile robots. The last section of the chapter is devoted to the formulation of the problem that will be attacked in this thesis.

1.1 Coordinated Motion and Coordinated Task Manipulation

Modeling groups of autonomous mobile robots engaged in coordinated behavior has been of increasing interest in the last years [1] - [19], [23] - [27], [49]. The applications of such a research field include tasks such as exploration, surveillance, search and rescue, mapping of unknown or partially known environments, distributed manipulation and transportation of large objects, reconnaissance, remote sensing, hazard identification and hazard removal [2], [6]. In particular, robotic soccer has been an important application area and eventually became a diverse and specific problem towards which many studies have been carried out [20] - [22].

The term *coordinated motion* generally denotes the motion of systems, which consist of more than one robot where the motion of each is dependent on the motion of the others in the group, mostly to accomplish a coordinated task. *Coordinated task manipulation* by a group of mobile robots, on the other hand, is defined as the accomplishment of a specified task together in certain formations. The necessary formation may vary based on the specifications of the coordinated task [10]. A rectangular formation could be better to carry a heavy rectangular object whereas circular formations might be better for capturing and enclosing the invader to provide security in surveillance areas [12], [13].

Robotics has made great steps forward, triggering the development of individual autonomous mobile robots, while multi-robot systems research lags behind. The reason for this lagging lies in the fact that coordinated motion of a group of autonomous mobile robots is a very complicated problem. At the highest level, the overall group motion might be dealt with by viewing such a collection as an ensemble. On the other hand, at the lowest level distributed controls must be implemented which ensure that the robots maintain safe spacings and do not collide. The following problems are fundamental to multi-robot researchers [15]:

- Multi-robot system design is inherently harder than design of single robots.
- Multiple robots may distract activities of each other, in the extreme precluding the team from achieving the goal of the mission.
- A team may have problems with recognizing the case when one or more team members, or the team as a whole, becomes unproductive.

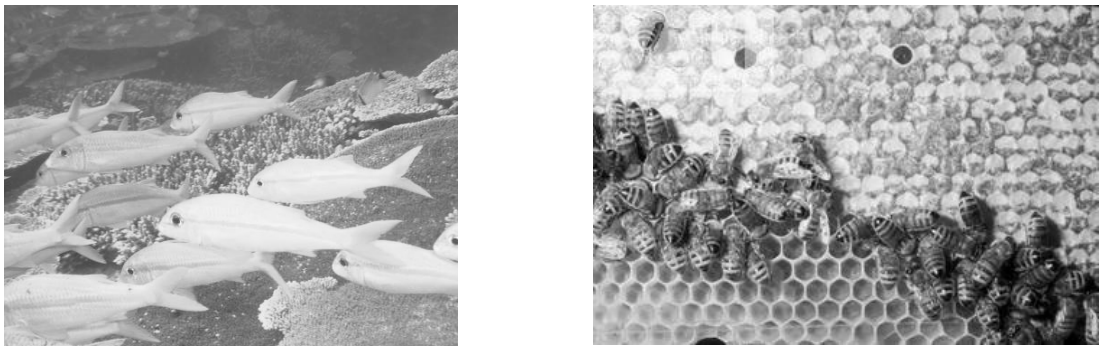
- The communication among the robots is a nontrivial issue.
- The "appropriate" level of individualism and cooperation within a team is problem-dependent.

The autonomous robots forming the group must avoid collisions with other members of the group and any other static or dynamic obstacles. Collision turns out to be one of the most essential problems in the context of coordinated motion [19]. Moreover, collision avoidance is the premier factor in generation of the reference trajectories to yield coordinated motion; i.e. the robots should change their path to avoid collisions even if this will introduce some delay for the achievement of the specified coordinated task.

1.2 Decentralized Systems

Computer science encountered a serious bottleneck with the increasing computational demand of applications such as databases and networks due to limited computational power. The idea of *decentralized systems* emerged in computer science society to fulfill such demands [23].

Flocking birds, schooling fish (see Fig. 1.1(a)) and bees building a honeycomb in the beehive (see Fig. 1.1(b)) are examples of decentralized groupings in nature, where each member works in coordination with the others [3]. In effect, coordinated motion of multiple autonomous mobile robots is an important application area for decentralized systems. In particular, multi-robot systems are different from other



(a) **(b)**

Figure 1.1: Decentralized natural groupings: (a)Schooling fish (b)Honey bees

decentralized systems because of their implicit "real world" environment, which is presumably more difficult to model compared to traditional components of decentralized system environments like computers, databases and networks. As a result of the wide application areas, the research efforts towards developing such systems has been monotonically increasing in the last decade [24] - [30].

The research efforts towards the development of decentralized robotic systems revealed the fact that, there are several tasks that can be performed more efficiently and robustly using distributed multiple robots [10]. The classical example of decentralized robotic systems is space exploration [15]. Another example is the exploration and preservation of the oceanic environments, the interest in which has gained momentum in recent years [25]. Following are the most appealing advantages of decentralized systems over centralized systems for robotics applications:

- Failure of a single robot in centralized systems results in system failure, whereas this will not necessarily jeopardize the whole mission assigned to a team in decentralized systems.

- Economic cost of a decentralized robotic system is usually lower than that of a centralized system that could carry out the same task, especially in the case when component failure is encountered [27].

- A huge single robot, no matter how powerful it is, will be spatially limited while smaller robots could achieve the same goal more efficiently.

- Decentralized systems outclass centralized systems in tasks such as exploration of an area for search and rescue activities [23].

1.3 Computer Vision for Mobile Robots

Sensing of the environment and subsequent control are important features of the navigation of an autonomous mobile robot. Hence, each member in a decentralized robotic system should gather information about its environment via some sensor during the manipulation of a specified coordinated task. This is crucial for a variety of tasks during navigation such as target detection and collision avoidance, which are common in most coordination scenarios. Although numerous types of sensors

exist in the market, two main types have been widely used in the context of coordinated motion. Ultrasonic range sensors mounted around the mobile robot as seen in Fig. 1.2(a) have been used to obtain distance information between the robot and any physical existence in its environment. Onboard camera(s) mounted on the mobile robot as depicted in Fig. 1.2(b) have been applied together with techniques from computer vision for autonomous sensing of the robot's environment.

There has been a significant research interest on vision-based sensing algorithms for the mobile robot navigation task [19], [27], [31] - [46]. In particular, some research was dedicated on the application of vision systems as the sensor basis of the autonomous mobile robots engaged in coordinated behavior [2], [47] - [50]. It has been shown that there are provable visual sensing strategies advantageous over any other sensing techniques for mobile robot navigation [31]. In spite of these accumulated studies on autonomous mobile robots with visual capabilities, there is still great challenge for computer vision systems in the area since such systems require skills for the solution of complex image understanding problems. Existing algorithms are not designed with real-time performance and are too luxurious from the aspect of time consumption. The development of a vision system which can satisfy the needs of both robustness and efficiency is still very difficult [45]. Concentration of computer vision society has been accumulated on estimation of the state of the robot in the environment and the structure of the environment [46].

(a) **(b)**

Figure 1.2: Possible sensors for mobile platforms: (a)Ultrasonic sensors (b)Onboard camera

1.4 Formulation of Coordinated Task

Coordinated behavior among a group of autonomous mobile robots is a hot research area in various disciplines - mechatronics, computer science, robotics, etc - due to various application areas of decentralized robotic systems such as exploration, surveillance, search and rescue, mapping of unknown or partially known environments, distributed manipulation and transportation of large objects, reconnaissance, remote sensing, hazard identification and hazard removal as mentioned at the beginning of this chapter.

In this work, a generic coordinated task explained below will be used as a test bed to verify the validity of the proposed models for the coordinated motion of a group of autonomous mobile robots. The mobile robots engaged in coordinated behavior will be assumed to be nonholonomic, because autonomous nonholonomic mobile robots are low-cost, off-the-shelf and easy to find test beds in the market. A vehicle is nonholonomic if it has a certain constraint on its velocity in moving certain directions. For example, two-wheeled mobile robots are nonholonomic since they can not move sideways unless there is slip between their wheels and the ground. Two-wheeled robots and car-like vehicles are the most appealing examples.

A group of n autonomous *nonholonomic* mobile robots, namely $R_1, R_2, \ldots, R_{n-1}$, R_n, and an object, T, that will serve as a target for the group, are considered. In the sequel, R_i denotes the i^{th} robot in the group.

The coordinated task scenario and the required formations for the coordinated motion in this work can be summarized as follows:

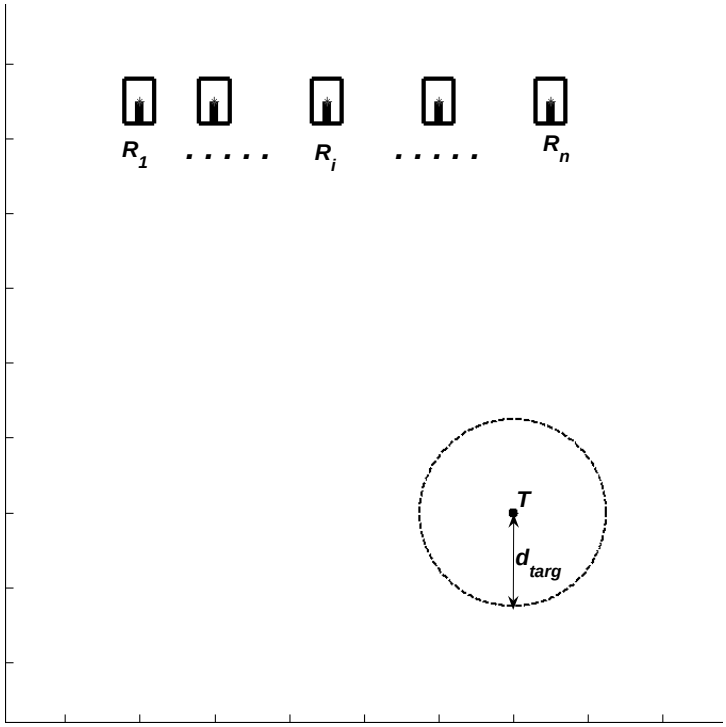
- Starting from any initial setting of the robots and the target, $R_1, R_2, \ldots, R_{n-1}$, R_n should form a circle of certain radius d_{targ}, with T being at the center.
- The robots should move in a coordinated manner maintaining certain mutual distances; i.e. they should approach T as a group.
- The robots should be uniformly distributed on the formation circle, with each robot maintaining a certain distance d_{near} from its closest neighbor.
- Each R_i should orient itself towards T once it achieves the requirements stated in the previous items.

A possible initial configuration for the above defined coordinated task is depicted in Fig. 1.3(a) for a group of n autonomous mobile robots. Fig. 1.3(b) on the other hand, shows the desired state of a group of five robots after the coordinated task is accomplished.

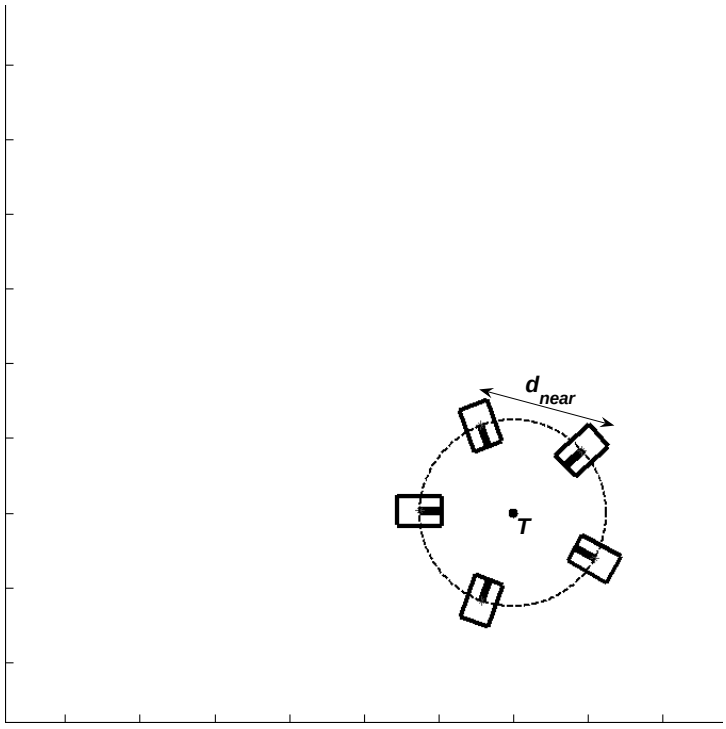
Complicated coordinated tasks can be dealt with in terms of simpler coordinated tasks that are manipulated sequentially. The instant implication of this idea is that the above scenario might serve as a general basis for more complicated coordinated tasks. For example, consider the manipulation of a heavy object, T, by a nonholonomic mobile robot group as the coordinated task. To accomplish such a coordinated task, the robots should first approach the object and grasp it in a formation as uniform as possible for mechanical equilibrium that will provide ease in lifting. Once the robots achieve the desired formation described in the above scenario, they can grasp, lift and move the object to any desired pose (location and orientation) in a coordinated manner. Another example is enclosing and catching a prisoner, T, in a surveillance area by such a nonholonomic mobile robot group. To achieve this goal, the distances d_{targ} and d_{near} should be decreased after the above explained coordinated task has been finalized.

Dealing with coordinated tasks as a sequence of simpler tasks, each of which can be considered as a "phase" of the whole task, the phenomenon of initiation of phases arises. In the first example given above, each R_i should check if the others have taken hold of the object before trying to lift it. On the contrary, the other robots can start attacking the prisoner without checking the state of the other robots in the latter scenario.

In the generic coordinated task investigated in this work, a stationary target, T, the position of which is a priori known by all autonomous nonholonomic mobile robots, is assumed for the sake of simplicity. The final assumption to specify the coordinated task is that the robots communicate their positions and velocities, out of which orientations can be extracted using the nonholonomic constraint, to each other by some communication protocol. This is not trivial, but the design of such communication protocols is out of the scope of this work. Instead, the research effort is more concentrated on establishing models and designing methods to supply coordinated motion of the autonomous nonholonomic mobile robot group.



(a)



(b)

Figure 1.3: The specified coordinated task scenario: (a)A possible initial configuration for n robots (b)Desired final configuration for 5 robots

In this thesis, two novel approaches to the problem of modeling coordinated motion of a group of autonomous nonholonomic mobile robots will be developed. An online collision avoidance algorithm, that will be explained in later chapters, will be inherent in both approaches. Chapter 2 gives a brief literature survey on the issues related to coordinated motion of a group of autonomous mobile robots and outlines the previous studies along with the presentation of the previous results in the area. Chapter 3 is on modeling and control of nonholonomic mobile robots. The first approach developed in this thesis is presented in detail in Chapter 4. In Chapter 5, the details of the second model developed in this thesis are given. The results of the simulations and experiments are given in Chapter 6. In Chapter 7, the thesis is concluded with some remarks on the developed models and some possible future work is presented.

Chapter 2

A Brief Survey on Coordination

Essential aspects of modeling the coordinated motion of a group of autonomous mobile robots have been outlined in Chapter 1. The problem can be summarized as follows: A group of autonomous mobile robots should move in a coordinated fashion for the achievement of a specified task, each member avoiding possible collisions with the other members of the group and the obstacles around. The development of models describing the motion of each autonomous member in the group - hence the motion of the entire group - is an important and nontrivial problem.

The research effort in modeling groups of autonomous mobile robots engaged in coordinated behavior has been growing recently [1] - [19], [23] - [27], [49]. This chapter outlines some methods in the literature that researchers from diverse disciplines have developed to attack this challenging problem; i.e. their interpretation of the problem, approaches developed to end up with good models, etc.

There are several ways in which researchers in different areas interpret coordination. For instance, computer scientists dealing with networks think of coordination as the communication of the computers through a network; i.e. *multi-agent systems* in computer science jargon. A coordinated task in their sense is either a computation requiring very high computational power that can be provided by multiple computers or shared use of a specific hardware among the agents. On the other side of the coin, studies in robotics consider coordination generally among a group of robots, often mobile, that is designed to achieve a predefined coordinated task as described in Section 1.1. Researchers in telecommunications society on the other hand, deal with the problem of data transfer between the autonomous robots in a group of mobile robots performing a coordinated task.

The rest of this chapter introduces the most common approaches to the following three main aspects of the problem:

- *Coordination Constraint*
- *Modeling Approach*
- *Sensory Base*

2.1 Coordination Constraints

The coordinated motion of a group of autonomous mobile robots is defined as the motion of the group maintaining certain formations. There are a variety of different approaches to this maintenance problem in the literature [15].

2.1.1 Leader-Follower Configuration

In this configuration, the group has one or more leader(s) and the motion of the so-called followers is dependent on the motion of the leader(s). In that sense, the system becomes centralized - a direct disadvantage of which is the risk in case of the failure of a leader. Leader-follower configuration is compared with decentralized schemes in [2], [14], [15] and [27].

The simple leader-follower configuration is depicted in Fig. 2.1. In this scenario, R_j follows R_i with a predefined separation l_{ij} and a predefined orientation ψ_{ij}; which is the relative orientation of the follower with respect to the leader as shown.

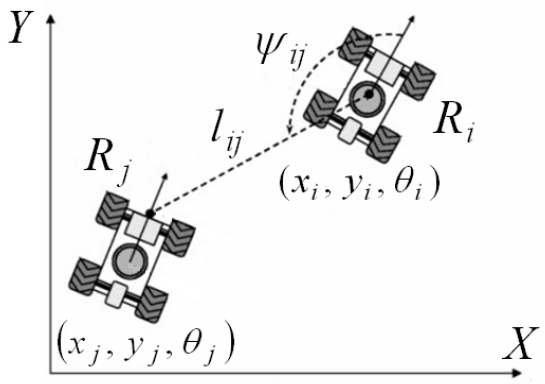


Figure 2.1: Leader-follower configuration

This two-robot system can be modeled if a suitable transformation to a new set of coordinates where leader's state is treated as an exogenous input is carried out. The stability of this system was proven using input-output feedback linearization under suitable assumptions in [2].

For flocking birds, V-shaped formation was shown to be advantageous for aerodynamic and visual reasons [11]. Such a formation depicted in Fig. 2.2 seems as a good example of leader-follower configuration. However, investigations revealed that there's actually no leader and the members are shifted from the leader position to the very back of the V-shape periodically since the members closer to the leader position spend more power. This switching behavior motivates the studies towards decentralized systems.

2.1.2 Leader-Obstacle Configuration

This configuration allows a follower robot to avoid the nearest obstacle within its sensing region while keeping a desired distance from the leader. This is a nice and reasonable property for many practical outdoor applications.

The simple leader-obstacle configuration is depicted in Fig. 2.3. In this scenario, the outputs of interest are l_{ij} and the distance δ between the reference point P_j on the follower, and the closest point O on the object.

A virtual robot R_o moving on the obstacle's boundary is defined with heading θ_o tangent to the obstacle's boundary for modeling purposes. This system was shown to be stable under suitable assumptions by input-output feedback linearization in [2].

This configuration might be considered as a centralized system due to the dependency of the path of the follower on that of the leader. On the other hand, the autonomous behavior of the follower robot in the presence of obstacles introduces some level of decentralization in this system.

Figure 2.2: V-shaped formation of flocking birds

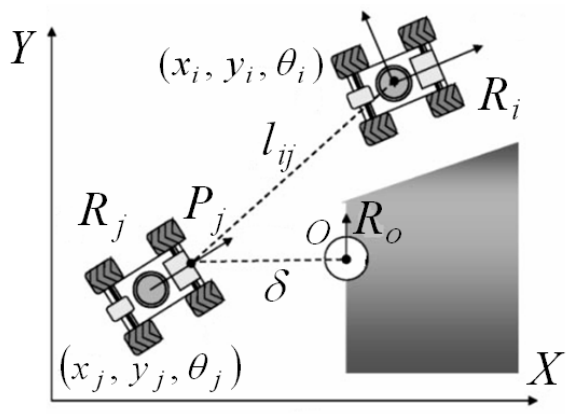


Figure 2.3: Leader-obstacle configuration

2.1.3 Shape-Formation Configuration

When there are three or more robots in the group, two consecutive leader-follower configurations might be used with a random selection of the leaders and followers for each pair. Instead, a shape formation configuration that will enable interaction between all robots may be used to implement a decentralized system.

This configuration is depicted in Fig. 2.4 for a group of three robots. In this scenario, each robot follows the others with desired separations. e.g. R_k follows R_i and R_j with desired distances l_{ik} and l_{jk} respectively as seen in the figure.

This system was also proved to be stable under suitable assumptions by input-output feedback linearization in [2]. The proof is done by the aid of suitable coordinate transformations.

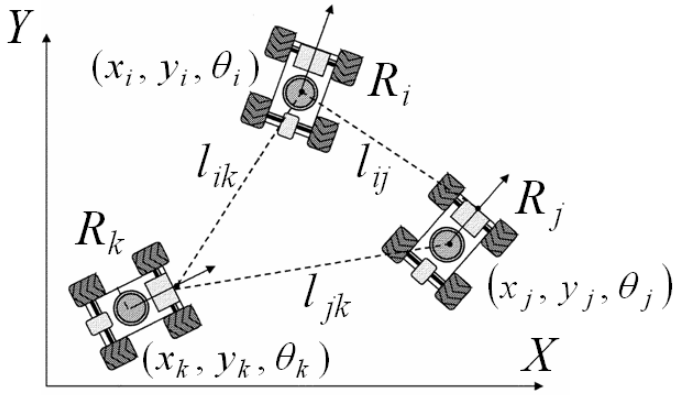


Figure 2.4: Shape-formation configuration

An important property of this configuration is that it allows explicit control of all separation distances; hence minimizes the risk of collisions. This property makes this configuration preferable especially when the distances between the robots are small.

2.2 Modeling Approaches

The mathematical model of a group of autonomous mobile robots has been derived using a variety of different ideas in the literature. In other words, there are diverse approaches to the derivation of mathematical representation of the rules dictating the motions of the robots [30]. Note that a hybrid system that is constructed as a combination of the ideas presented in the following subsections might be used to model coordinated motion of a group of autonomous mobile robots; i.e. the ideas in the following subsections do not fully contradict with each other.

2.2.1 Potential Fields

In this approach, the robot is assumed as a single point and a generally circular virtual potential field is considered around it. The idea of defining navigation path of a robot on the basis of potential fields has been used extensively in the literature [16], [29].

Baras *et. al.* constructed a potential function for each robot consisting of several terms, each term reflecting a goal or a constraint [29]. In that work, the position of the robot i at time t is denoted as $p_i(t) = (x_i(t), y_i(t))$. The potential function $J_{i,t}(p_i)$ for the robot i at time t is then given as:

$$\begin{aligned} J_{i,t}(p_i) &= \lambda_g J^g(p_i(t)) + \lambda_n J^n_{i,t}(p_i(t)) \\ + \quad & \lambda_o J^o(p_i(t)) + \lambda_s J^s(p_i(t)) + \lambda_m J^m_t(p_i(t)) \ , \end{aligned} \tag{2.1}$$

where J^g, $J^n_{i,t}$, J^o, J^s, J^m_t are the components of the potential function while and λ_g, λ_n, λ_o, λ_s, $\lambda_m \geq 0$ are the corresponding weighting coefficients due to the target (goal), neighboring robots, obstacles, stationary threats and moving threats, respectively. The velocity $\dot{p}_i$ that will be used as the reference signal by a low-level velocity controller is calculated by:

$$\dot{p}_i(t) = -\frac{\partial J_{i,t}(p_i)}{\partial p_i} \ . \tag{2.2}$$

The components of the potential function are described as follows:

- The target potential $J^g(p_i) = f_g(r_i)$ where r_i is the distance of the i^{th} robot to the target, and $f_g(\cdot)$ a suitably defined function satisfying $f_n(r) \to 0$ as $r \to 0$. Most researchers in the area defined this function as $f_g(r) = r^2$ motivated by Newton's gravitational force;

- The neighboring potential $J^n_{i,t}(p_i) = f_n(|p_i - p_j|)$ where p_j denotes the position of an effective neighbor, and $f_n(\cdot)$ is an appropriately defined function satisfying $f_n(r) \to \infty$ as $r \to 0$.

- The obstacle potential $J^o(p_i) = f_o(|p_i - O_j|)$ where O_j denotes the position of the obstacle, and $f_o(\cdot)$ is an appropriately defined function satisfying $f_o(r) \to \infty$ as $r \to 0$.

- The potential J^s due to stationary threats. This can be modeled similarly as the obstacle potential.

- The potential $J^m(p_i) = f_m(|p_i - q_j|)$ due to moving threats where q_j denotes the position of the threat, and $f_m(\cdot)$ is an appropriately defined function satisfying $f_m(r) \to \infty$ as $r \to 0$. $f_m(\cdot)$ might be a piecewise continuous function depending on the sensing range of the robot.

A sliding mode controller was used in [16] to track similarly obtained references to provide collision-free trajectories for the autonomous mobile robots. In that work, the notion of "behavior arbitration" is introduced for the adjustment of the weighting coefficients of the potentials due to the target and the obstacle. A result of the algorithm developed in that work is depicted in Fig. 2.5.

2.2.2 Formation Vectors

Yamaguchi introduced "formation vectors" to model coordinated motion of mobile robot troops aimed for hunting invaders in surveillance areas in [12]. In that work,

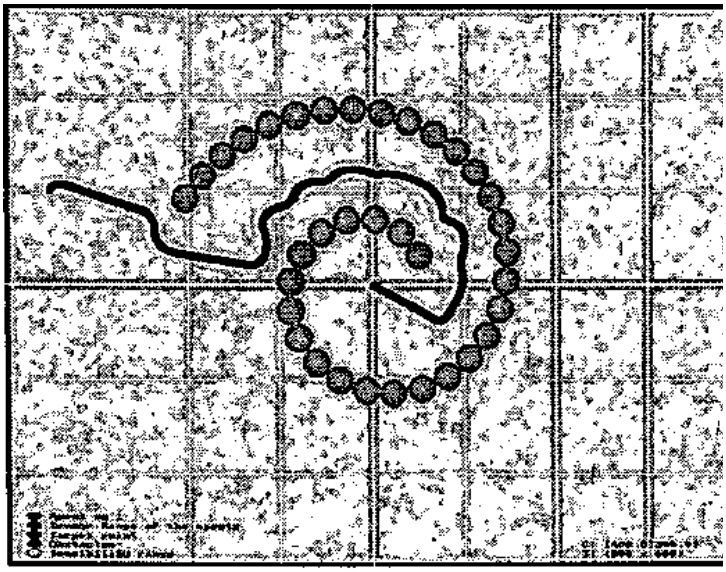


Figure 2.5: A simulation result using potential fields; the robot moves inwards

each robot in the group controls its motion autonomously and there's no centralized controller. To make formations enclosing the target, each robot especially has a vector called "formation vector" and formations are controlled by these vectors. These vectors are determined by a reactive control framework heuristically designed for the desired hunting behavior of the group.

Under the assumption of n mobile robots forming a strongly connected configuration initially - i.e. each robot senses at least one neighboring robot - each robot at the start of an arrow keeps a certain relative position to the robot at the end of the arrow to form formations. The velocity controller of the i^{th} robot in the troop, R_i, implements the control strategy:

$$\begin{aligned}\begin{pmatrix} \dot{x}_i \\ \dot{y}_i \end{pmatrix} &= \sum_{j\epsilon L_i} \tau_{ij} \left\{ \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right\} + \tau_i \left\{ \begin{pmatrix} x_t \\ y_t \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right\} \\ &+ \begin{pmatrix} d_{xi} \\ d_{yi} \end{pmatrix} + \sum_{j\epsilon OBJECTS} \delta_{ij} \left[\begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right] \\ &+ \sum_{j\epsilon OBJECTS} \delta_{ij} D \left[\left\{ \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right\} / \left| \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right| \right] ,\end{aligned} \tag{2.3}$$

with:

$$\delta_{ij} = \begin{cases} \delta, & \text{if } \left| \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right| \leq D \\ \\ 0, & \text{if } \left| \begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right| > D \end{cases} ,$$

$$j \epsilon OBJECTS = L_i \cup M_i \cup N_i \cup TARGET ,$$

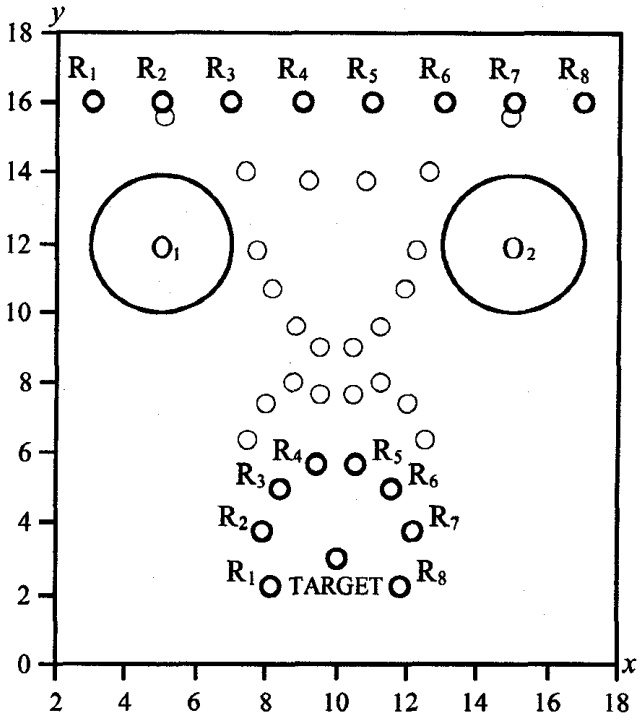
where $[\dot{x}_i,\ \dot{y}_i]^t$ is the velocity of R_i, L_i is the set of robots that are considered as neighbors by R_i, M_i is the set of robots and N_i is the set of obstacles that are sensed by R_i for collision avoidance, $[\dot{x}_j,\ \dot{y}_j]^t$ is the position of the j^{th} robot in the group, R_j, $[\dot{x}_t,\ \dot{y}_t]^t$ is the position of the $TARGET$, τ_{ij} is the attraction coefficient of R_i to R_j, τ_i is the attraction coefficient of R_i to $TARGET$, δ_{ij} is the repulsion coefficient of R_i from the obstacles and $[d_{xi},\ d_{yi}]^t$ is the formation vector associated with R_i.

As implied by (2.3), each robot, R_i, is repelled by its neighbors when they are considered as obstacles (the distance between R_i and that robot is below D); hence collisions are avoided.

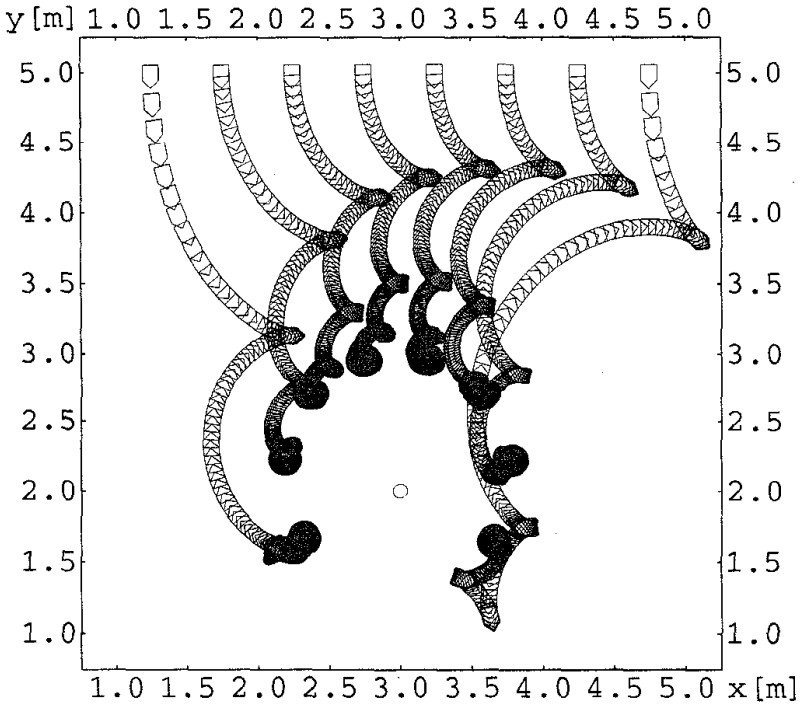
The formation vector associated with R_i is determined according to the relative position of R_i to the $TARGET$ and its neighbors, i.e. robots in L_i. Examples explaining the determination of the formation vector are given in [12].

A simulation result from [12] is given in Fig. 2.6(a). In this scenario, eight holonomic(omnidirectional) robots successfully avoid collisions with static obstacles, O_1 and O_2, as well as the other members of the group, and form a circular formation around the invader, $TARGET$.

Similar ideas could be extended for multiple nonholonomic mobile robots by adding an extra term to (2.3) due to the nonholonomic constraint on the velocity of the mobile robot [13]. A simulation result given by Yamaguchi for the case of eight nonholonomic mobile robots can be seen in Fig. 2.6(b)



(a)



(b)

Figure 2.6: Simulation results using formation vectors: (a)Eight holonomic mobile robots (b)Eight nonholonomic mobile robots

2.2.3 Nearest Neighbors Rule

In 1995, Vicsek *et. al.* proposed the simple "nearest neighbors" method in order to investigate the emergence of autonomous motions in systems of particles with biologically motivated interaction in a *Physical Review Letters* article [59]. The model can be summarized as follows: Particles are driven with a constant absolute velocity and at each time step they assume the average direction of motion of the particles in their neighborhood with some random perturbation added. The developed model was able to mimic the motion of bacteria that exhibit coordination motion in order to survive under unfavorable conditions with a good approximation. This idea has then been widely used in the literature to attack the problem of modeling the coordinated motion of a group of autonomous mobile robots [13], [14], [17], [6].

Jadbabaie *et. al.* provided a theoretical explanation for the observed behavior of such a system moving according to nearest neighbors rule both for a leaderless configuration and a leader-following configuration [14]. In that work, they showed that Vicsek model is a graphic example of a switched linear system which is stable, but for which a common quadratic Lyapunov function doesn't exist.

In [18], it has been qualitatively shown that a group of autonomous mobile robots will always break into several separated groups and all robots from each of those groups will go in the same direction. However, this proof was done in the absence of a global attraction to some direction, e.g. attraction to a possible target.

2.3 Sensory Bases

The information about the robot's environment might be based on different sensors as explained in Section 1.3. The following sections introduce some of the possible problems and their solutions encountered in the literature.

2.3.1 Sensor Placement

Placement of a number of sensors on a mobile robot is crucial for the gathering of maximum information about the environment, which in turn utilizes the implementation of developed coordination models. This problem has been addressed in several studies such as [2], [47] - [50].

A new method that uses Christofides algorithm along with graphical methods to determine the shortest necessary path between the viewpoints for planning the viewpoints for 3D modeling of the environment has been developed in [47].

Salomon developed a force model for the appropriate placement of sensors on the robots in [4]. In that work, dynamically evolvable hardware is used; i.e. the poses of the sensors are functions of interest areas that might be time-variant. The idea is illustrated in Fig. 2.7 and the algorithm is summarized as follows:

- The sensing mechanism of the robot consists of n sensors among which the first and the last are rigidly connected to the robot's body.
- A robot has some interest regions which might be considered as attraction forces. Due to these forces, $n-2$ sensors evolve to their final poses.
- The final poses of the sensors are decided by a simple network of springs connected between the sensors.

Salomon's model draws from some biological observations. The insertion of a new cell into a bunch of cells at some particular place would have a strong effect in its vicinity but a rather small effect globally. Similarly, a particular force F_i twice as strong as before due to the placement of an object of high interest in the area sensed by the sensors i and $i+1$ would almost double the angle between those sensors, whereas it would decrease the other angles by a small amount dependent on the number of sensors.

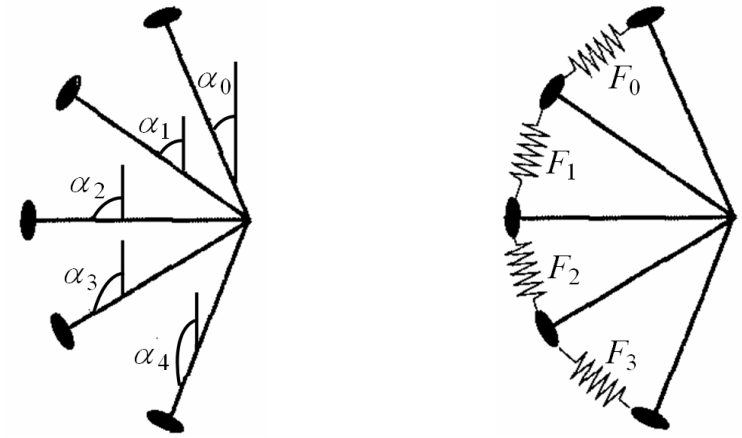


Figure 2.7: Sensor placement techniques: (a)Conventional (b)Salomon's force model

2.3.2 Ultrasonic Sensors

Ultrasonic sensors are mostly used for the measurement of distances that will be utilized for self-localization of the robot or collision avoidance algorithms. Compared with other detection modes, ultrasonic sensing is a favorable distance measurement mode due to its robustness against changes in environmental factors such as temperature, color, etc. Most ultrasonic sensors have the transmitter and receiver on the same side. For continuous distance measurements using such sensors, ultrasonic pulse echo technique is widely used. The working principle is: the sender sends an ultrasonic pulse - a sound wave transmits in the medium - that is reflected when it hits physical objects. Recording the duration between sending and receiving, the distance from the sensor to reflection point is calculated based on the velocity of sound wave in the medium.

An ultrasonic sensor ring attached to the robot base is used to sense the distance to the physical objects in the environment to avoid the collisions of the robots with the walls and other robots in the robotic soccer examples of [20] - [22].

2.3.3 Vision Sensors

Onboard cameras mounted on the basis of the robots have been widely used as the sensory base for mobile robots in the literature. Many researchers proposed diverse methods for the problems that arise when a camera is used on a mobile platform such as self-localization of the robots, vision-based control of mobile robots, 3D modeling of the environment, collision avoidance, etc for the last two decades [8], [19] - [22], [32] - [39], [41], [43] - [46]. The reason behind this high interest in using cameras as sensors is the fact that they are cheap and off-the-shelf components, which can be used for various goals.

The characters of integration of traffic control systems and dynamic route guidance systems based on visual sensing are analyzed and a two kind agent model is developed [8]. In that work, route guidance and traffic control are addressed separately and the mobile robots are assigned as route guidance agents or traffic control agents dynamically according to the circumstances.

A new geometric method for the estimation of the camera angular and linear velocities with respect to a planar scene was developed by Shakernia *et. al.* [34].

In that article, the problem of controlling the landing of an unmanned air vehicle via computer vision was presented. Differential flatness of the planar surface in the image was utilized in the control loop of the vehicle.

Hai-bo *et. al.* developed a fast and robust vision system for autonomous mobile robots equipped with a pan-tilt camera [45]. The performance of the system in terms of its speed and robustness is increased through appropriate choice of color space, algorithms of mathematical morphology and active adjustment of parameters. In that work, preprocessing of images and intelligent subsampling are used to facilitate high sampling rates of around 25Hz.

Most of these studies attack the problem of developing a robust model for a single pan-tilt camera attached to the base of the mobile robot. The following sections present other possibilities commonly encountered in literature.

Omnidirectional Cameras

An omnidirectional camera captures images of the environment in a radial form. Actually, the lens is mounted pointing upwards on the robot and captures the image of a circle of certain radius around the robot with the help of a mirror fixed across the lens. A sample image captured by such a camera is shown in Fig. 2.8.

Omnidirectional cameras were used as sensors for mobile robots in a group performing coordinated tasks in [20], [27], [44], and [48]. The use of such cameras is advantageous for some tasks such as 3D modeling of the ground plane and self-localization of the robots. Spletzer *et. al.* developed a framework for the coordina-

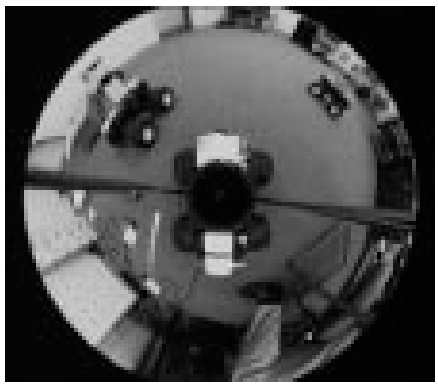


Figure 2.8: Sample image from an omnidirectional camera

tion of multiple mobile robots, which use vision for extracting the relative position and orientation information [27]. In that work, a centralized localization method is used although every robot has its own onboard camera. A group of three mobile robots in a box-pushing task was demonstrated using "shape-formation configuration" described in Section 2.1.3.

Similar ideas were applied for a catadioptric (using refracted and reflected light) omnidirectional camera system, depicted in Fig. 2.9 for a robotic soccer team in [20]. In that work, ultrasonic sensors are used together with the omnidirectional camera to avoid collisions with the other members of the group and the walls of the soccer field.

Multiple Vision Agents

Robotic applications often need simultaneous execution of different tasks using different camera motions. Most common tasks in such a mobile robot navigation are following the road, finding moving obstacles and possible traffic signs, viewing and memorizing interesting patterns along the route to create a model of the environment, etc. Robots that are equipped with a single camera allocate their camera and computational power for these tasks sequentially; hence the sampling rate for the control algorithm of the robot is decreased.

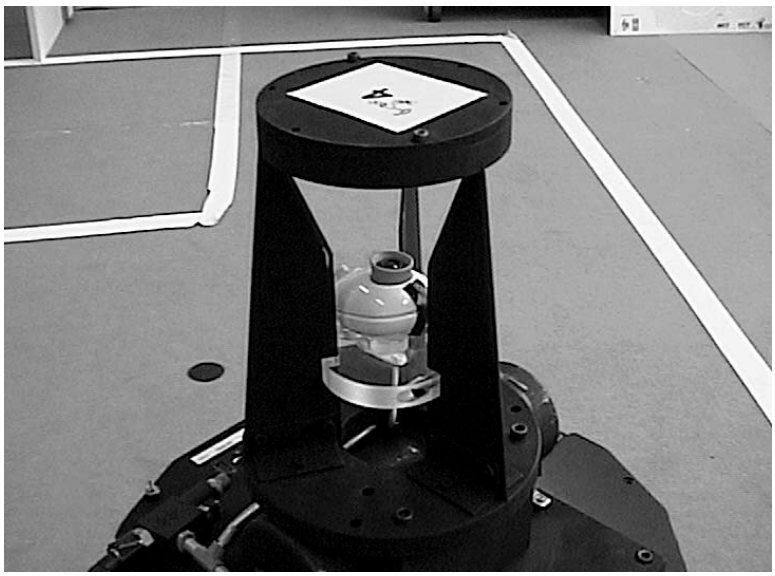


Figure 2.9: Catadioptric omnidirectional vision system mounted on a robot

The concept of *Multiple Vision Agents* (MVA) were introduced to attack this problem in [40]. In that work, a system with 4 cameras moving independently of each other was developed. Each agent analyzes the image data and controls the motion of the camera it is connected to. The various visual functions are assigned to the agents for the achievement of the task in the real world. The following three properties are common to each agent of the MVA:

- Each agent corresponds to a camera and controls the motion of that camera independently.
- Each agent has a computing resource of its own and processes the image taken by its camera using that resource.
- Each agent behaves according to three instincts of visual perception:
  - Moving obstacle tracking.
  - Goal searching.
  - Free region detection.

  These instincts are activated in the assigned order according to the priority of the instinct given in Fig. 2.10.

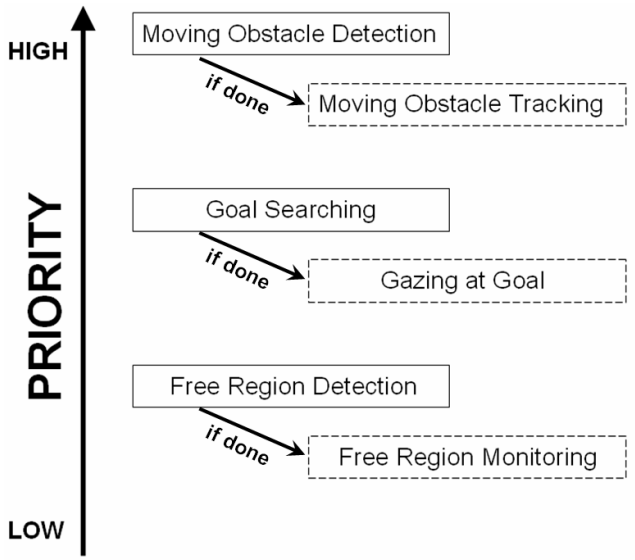


Figure 2.10: Visual perception instincts in a hierarchical design

The idea was extended in [35] for panoramic sensing of mobile robots. In that work, the robot builds up an outline structure of the environment, as a reference frame for gathering the details and acquiring the qualitative model of the environment, before moving.

Chapter 3

Nonholonomic Mobile Robots: Modeling & Control

A vehicle is nonholonomic if it has a certain constraint on its velocity in moving certain directions. For example, two-wheeled mobile robots are nonholonomic since they can not move sideways unless there is slip between their wheels and the ground. Two-wheeled robots and car-like vehicles are the most appealing examples in daily life. The nonholonomic constraint complicates the development of mathematical representations and control laws for such robots.

Systems with nonholonomic constraints - and consequently chained systems - have received significant research interest in robotics society, especially in the last two decades. Results of studies in the area can be found in [51] - [58].

In this work, the coordinated motion of a group of nonholonomic mobile robots is investigated. Two-wheeled robots, often referred as "*unicycle*", are used as test beds to test the performance of the developed methods. The following sections outline the basics of mathematical modeling and control laws for these specific type of nonholonomic mobile robots.

3.1 Modeling

The model of a physical system can be dynamic or kinematic. However, the nonholonomy of a unicycle type mobile robot introduces the following constraint on the velocity of the robot: the robot can't perform any sideways motion. Due to this fact, dynamic modeling of unicycle robots is very complicated; e.g. an attractive force acting on the robot will not be able to move the robot if its orientation coincides with the axis of wheels. Instead, nonholonomic mobile robots are represented by

their kinematic models. It's widely known that the kinematic model for unicycle robot is given by the equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_1 \cos\theta \\ u_1 \sin\theta \\ u_2 \end{bmatrix} , \tag{3.1}$$

where x and y are the Cartesian coordinates of the center of mass of the robot, θ is its orientation with respect to the horizontal axis, u_1 and u_2 are its linear and angular velocities, respectively.

The pose of a robot in cartesian coordinates is represented by the three variables x, y and θ. In the above equation, these variables can be treated as outputs while u_1 and u_2 can be utilized as inputs. In other words, the linear and angular velocities of the robot should be designed appropriately for the robot to achieve a specified pose. The mathematical representation of the system can be rewritten for such an approach as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_2 \, . \tag{3.2}$$

The velocities u_1 and u_2 in the above equations are related to the linear velocities of the centers of the right and left wheels with:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} (u_R + u_L)/2 \\ (u_R - u_L)/(2\lambda) \end{bmatrix} , \tag{3.3}$$

where λ is the half length of the wheel axis as shown in Fig. 3.1.

3.2 Control

The difficulty of the control problem for nonholonomic mobile robots originates from the nature of (3.2). In that system, only two controls, the linear and angular velocities of the robot, are used to control three outputs for the pose of the robot.

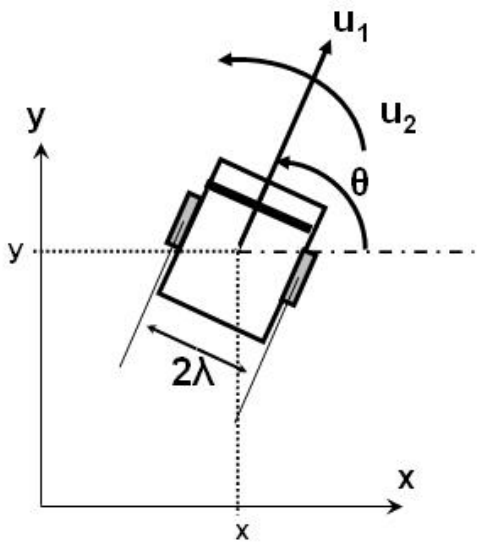


Figure 3.1: A unicycle robot and its variables of interest

Nonholonomic mobile robots cannot be stabilized to a desired pose by using smooth state-feedback control although they are completely controllable in their configuration space [52]. However, feedback stabilization of a point on a nonholonomic mobile robot was shown to be possible in [53]. In that work, C. Samson and K.Ait-Abderrahim proved that feedback stabilization of the robot's pose around the pose of a "*virtual reference robot*" is possible provided the reference robot keeps moving. Consequently, *tracking of time-variant reference trajectories* and *parking* should be considered as different control problems as pointed out in [54].

The well known problem of switching between controllers arises here. For most tasks about nonholonomic mobile robots, the designed controllers switch between the proposed solutions for trajectory tracking and parking. If a high frequency switching occurs, this might risk the stability of the system. However, there's no better solution as of today in the literature, so similar switching of controllers will be used in this work.

3.2.1 Trajectory Tracking Problem

For time-variant reference trajectory tracking, the reference trajectory must be selected to satisfy the nonholonomic constraint. This is ensured by tracking a *virtual reference robot* which moves according to the model:

$$\begin{bmatrix} \dot{x_r} \\ \dot{y_r} \\ \dot{\theta_r} \end{bmatrix} = \begin{bmatrix} \cos\theta_r \\ \sin\theta_r \\ 0 \end{bmatrix} u_{1r} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_{2r} \ . \tag{3.4}$$

where x_r and y_r are the Cartesian coordinates of the center of mass of the virtual reference robot, θ_r is its orientation with respect to the horizontal axis, u_{1r} and u_{2r} are its linear and angular velocities, respectively.

If x_r, y_r and θ_r are continuously differentiable and bounded as $t \to \infty$ one can easily show that

$$\begin{bmatrix} u_{1r} \\ u_{2r} \\ \theta_r \end{bmatrix} = \begin{bmatrix} \dot{x_r}\cos\theta_r + \dot{y_r}\sin\theta_r \\ \left(\ddot{y_r}\dot{x_r} - \ddot{x_r}\dot{y_r}\right) / \left(\dot{x_r}^2 + \dot{y_r}^2\right) \\ \arctan\left(\dot{y_r}/\dot{x_r}\right) \end{bmatrix} \ . \tag{3.5}$$

The tracking errors $\tilde{x}$, $\tilde{y}$ and $\tilde{\theta}$ are defined as the difference between the actual robot's pose, $\begin{bmatrix} x & y & \theta \end{bmatrix}^t$, and pose of the virtual reference robot as follows:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{\theta} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} - \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix} \ . \tag{3.6}$$

To facilitate the generation of a control law, the transformed tracking errors (e_1, e_2 and e_3) are obtained using an invertible transformation as follows:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{\theta} \end{bmatrix} \ . \tag{3.7}$$

Based on the inverse transformation of (3.7), it is clear that $\begin{bmatrix} \tilde{x} & \tilde{y} & \tilde{\theta} \end{bmatrix}^t \to 0$ if $\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^t \to 0$ as $t \to \infty$ as seen below:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \ . \tag{3.8}$$

It is shown in [58] that the following controls, u_1 and u_2, with proper selection of constant control gains, $k_1 > 0$ and $k_2 > 0$, regulate all tracking errors to zero for time-variant reference trajectories:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 + u_{1r} \cos e_3 \\ -u_{1r} \left(\sin e_3 / e_3\right) - k_2 e_3 + u_{2r} \end{bmatrix} . \tag{3.9}$$

[57] provides an overview of trajectory tracking problems for nonholonomic vehicles. Several control objectives for such a system are discussed and reviewed from the aspects of robotics and control in that article. This would work as a good reference for further information about the time-variant trajectory tracking problem and the proposed solutions both for unicycle and car-like vehicles.

3.2.2 Parking Problem

Parking the robot at a fixed reference position, $\begin{bmatrix} x_p & y_p \end{bmatrix}^t$, with a fixed reference orientation, θ_p, is a different problem for nonholonomic vehicles from the control point of view, as explained above. The control law given by (3.9) doesn't regulate all of the three errors to zero in this case. In other words, there's no stabilizing time-invariant feedback law for fixed-point references.

When smooth state-feedback fails to stabilize the system in a control problem, it is common to search for a stabilizing discontinuous feedback. However, another approach might be to use a time-varying feedback law to have a smooth response [52].

For this problem, the parking errors $\tilde{x_p}$, $\tilde{y_p}$ and $\tilde{\theta_p}$ are defined as the difference between the actual robot's pose, $\begin{bmatrix} x & y & \theta \end{bmatrix}^t$, and the desired fixed pose as follows:

$$\begin{bmatrix} \tilde{x}_p \\ \tilde{y}_p \\ \tilde{\theta}_p \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} - \begin{bmatrix} x_p \\ y_p \\ \theta_p \end{bmatrix} . \tag{3.10}$$

A similar transformation as given in (3.7) can be applied to obtain transformed parking errors (e_{1p}, e_{2p} and e_{3p}) for easier construction of the control law as follows:

$$\begin{bmatrix} e_{1p} \\ e_{2p} \\ e_{3p} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_p \\ \tilde{y}_p \\ \tilde{\theta}_p \end{bmatrix} . \tag{3.11}$$

It is clear that $\begin{bmatrix} \tilde{x}_p & \tilde{y}_p & \tilde{\theta}_p \end{bmatrix}^t \to 0$ if $\begin{bmatrix} e_{1p} & e_{2p} & e_{3p} \end{bmatrix}^t \to 0$ as $t \to \infty$ based on the inverse transformation of (3.11).

It is shown in [58] that the following smooth time-varying feedback controls, u_{1p} and u_{2p}, with proper selection of constant control gains, $k_{1p} > 0$ and $k_{2p} > 0$, regulate all tracking errors to zero for fixed-point references:

$$\begin{bmatrix} u_{1p} \\ u_{2p} \end{bmatrix} = \begin{bmatrix} -k_{1p}e_{1p} \\ -k_{2p}e_{3p} + e_{2p}^2 \sin(t) \end{bmatrix} . \tag{3.12}$$

Note that there are other approaches to this problem. For example, Lee *et. al.* proposed a new method for the parking problem of nonholonomic mobile robots in 2004 [51]. In that work, the problem of switching between the two controllers given above by (3.9) and (3.12) is addressed. The proposed idea is to add a virtual trajectory to the original trajectory to create a reference trajectory for the parking problem and applying a smooth, time-invariant control that is derived by linearization and pole-placement techniques. Despite the fact that this is a solution for the controller-switching problem, the performance of switched controllers is still better.

3.3 Simulations for Gain Adjustments

The performance of the control laws presented above were investigated by simulations that were run in *Simulink* 6.1 and *MATLAB* 7.0.1. The block diagram generated for these simulations in *Simulink* is depicted in Fig. 3.2.

A single nonholonomic mobile robot, with a maximum linear speed of $1.0m/sec$ and a maximum angular speed of $(\pi/2)rad/sec$ was used in the simulations.

3.3.1 Trajectory Tracking Simulations

A circle of radius $r = 2.0m$ to be followed by $\omega = (\pi/6)rad/sec$ was used as a sample time-varying reference trajectory to investigate the performance of (3.9).

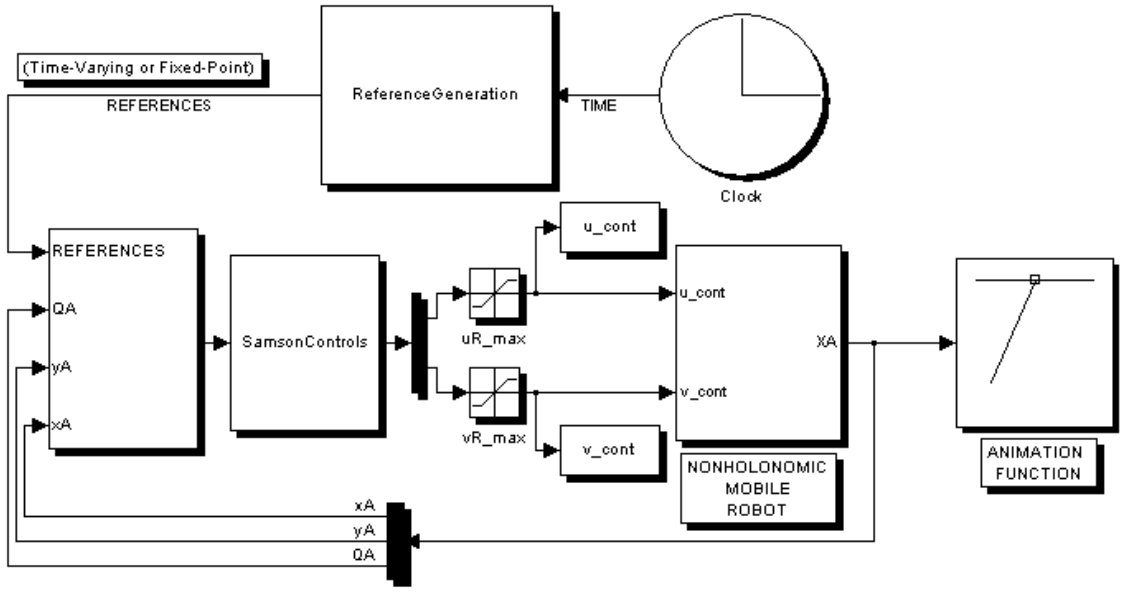
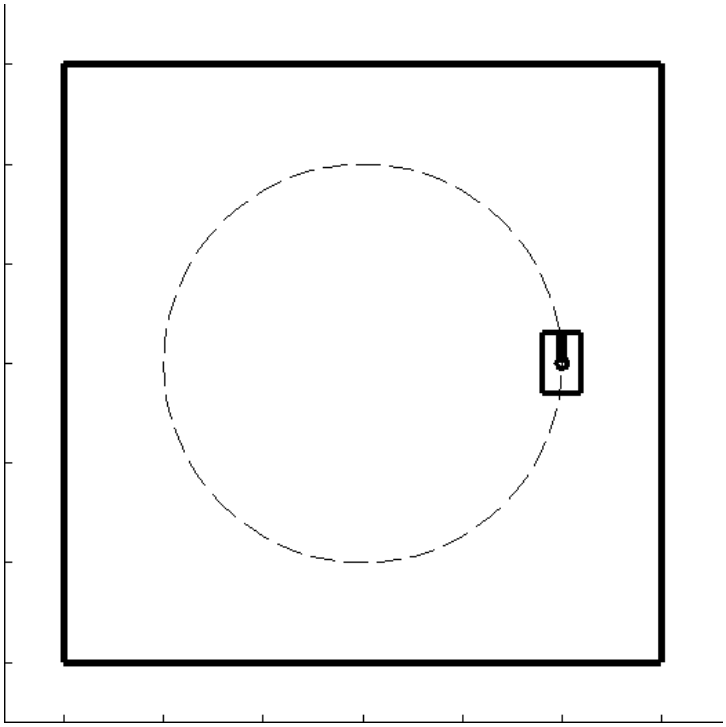


Figure 3.2: *Simulink* block diagram for the simulations of control laws

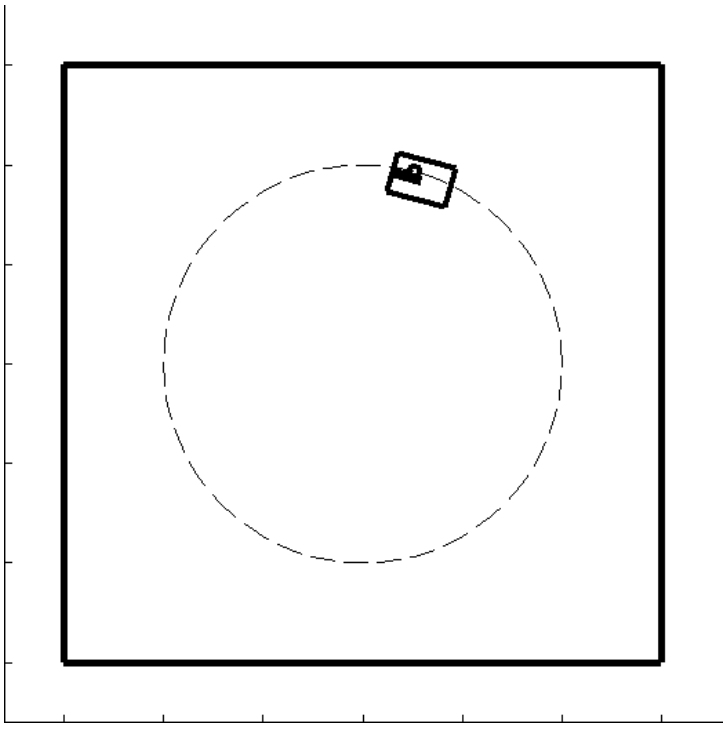
A variety of values for the control gains k_1 and k_2 in that control law were simulated and the average tracking errors were recorded for each case.

The simulations were run only for the upper-left quarter of the circle for time-consumption considerations. The initial and final poses of the nonholonomic mobile robot along with the reference circular trajectory are shown in Fig. 3.3. The little circle inside the rectangle depicting the robot is the reference position input to the robot; i.e. the position of the *virtual reference robot*.

The average tracking errors for each value of the control gains, k_1 and k_2, are given in Table 3.1.



(a)



(b)

Figure 3.3: Trajectory tracking scenario: (a)Initial pose (b)Final pose

Table 3.1: Average tracking errors for different values of control gains

$\mathbf{k_1}$	$\mathbf{k_2}$	$\tilde{\mathbf{x}}(m)$	$\tilde{\mathbf{y}}(m)$	$\tilde{\theta}(rad)$
3	3	0.025740	0.048942	0.00508680
3	6	0.025007	0.049741	0.00315590
3	9	0.024658	0.050110	0.00228530
3	12	0.024454	0.050321	0.00179090
3	15	0.024321	0.050458	0.00147210
3	18	0.024227	0.050554	0.00124970
3	21	0.024157	0.050625	0.00108560
3	24	0.024103	0.050679	0.00095963
6	3	0.025740	0.048942	0.00508680
6	6	0.025007	0.049741	0.00315590
6	9	0.024658	0.050110	0.00228530
6	12	0.024454	0.050321	0.00179090
6	15	0.024321	0.050458	0.00147210
6	18	0.024227	0.050554	0.00124970
6	21	0.024157	0.050625	0.00108560
6	24	0.024103	0.050679	0.00095963
9	3	0.025740	0.048942	0.00508680
9	6	0.025007	0.049741	0.00315590
9	9	0.024658	0.050110	0.00228530
9	12	0.024454	0.050321	0.00179090
9	15	0.024321	0.050458	0.00147210
9	18	0.024227	0.050554	0.00124970
9	21	0.024157	0.050625	0.00108560
9	24	0.024103	0.050679	0.00095963
12	3	0.025740	0.048942	0.00508680
12	6	0.025007	0.049741	0.00315590

Continued on next page

Table 3.1 – continued from previous page

$\mathbf{k_1}$	$\mathbf{k_2}$	$\tilde{\mathbf{x}}(m)$	$\tilde{\mathbf{y}}(m)$	$\tilde{\theta}(rad)$
12	9	0.024658	0.050110	0.00228530
12	12	0.024454	0.050321	0.00179090
12	15	0.024321	0.050458	0.00147210
12	18	0.024227	0.050554	0.00124970
12	21	0.024157	0.050625	0.00108560
12	24	0.024103	0.050679	0.00095963
15	3	0.025740	0.048942	0.00508680
15	6	0.025007	0.049741	0.00315590
15	9	0.024658	0.050110	0.00228530
15	12	0.024454	0.050321	0.00179090
15	15	0.024321	0.050458	0.00147210
15	18	0.024227	0.050554	0.00124970
15	21	0.024157	0.050625	0.00108560
15	24	0.024103	0.050679	0.00095963
18	3	0.025740	0.048942	0.00508680
18	6	0.025007	0.049741	0.00315590
18	9	0.024658	0.050110	0.00228530
18	12	0.024454	0.050321	0.00179090
18	15	0.024321	0.050458	0.00147210
18	18	0.024227	0.050554	0.00124970
18	21	0.024157	0.050625	0.00108560
18	24	0.024103	0.050679	0.00095963
21	3	0.025740	0.048942	0.00508680
21	6	0.025007	0.049741	0.00315590
21	9	0.024658	0.050110	0.00228530
21	12	0.024454	0.050321	0.00179090
21	15	0.024321	0.050458	0.00147210
21	18	0.024227	0.050554	0.00124970

Continued on next page

Table 3.1 – continued from previous page

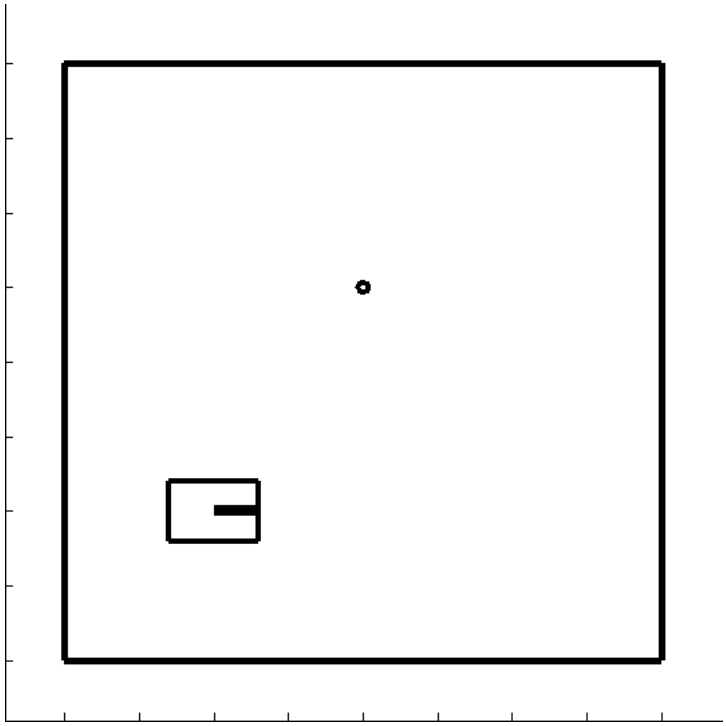
$\mathbf{k_1}$	$\mathbf{k_2}$	$\tilde{\mathbf{x}}(m)$	$\tilde{\mathbf{y}}(m)$	$\tilde{\theta}(rad)$
21	21	0.024157	0.050625	0.00108560
21	24	0.024103	0.050679	0.00095963
24	3	0.025740	0.048942	0.00508680
24	6	0.025007	0.049741	0.00315590
24	9	0.024658	0.050110	0.00228530
24	12	0.024454	0.050321	0.00179090
24	15	0.024321	0.050458	0.00147210
24	18	0.024227	0.050554	0.00124970
24	21	0.024157	0.050625	0.00108560
24	24	0.024103	0.050679	0.00095963

3.3.2 Parking Simulations

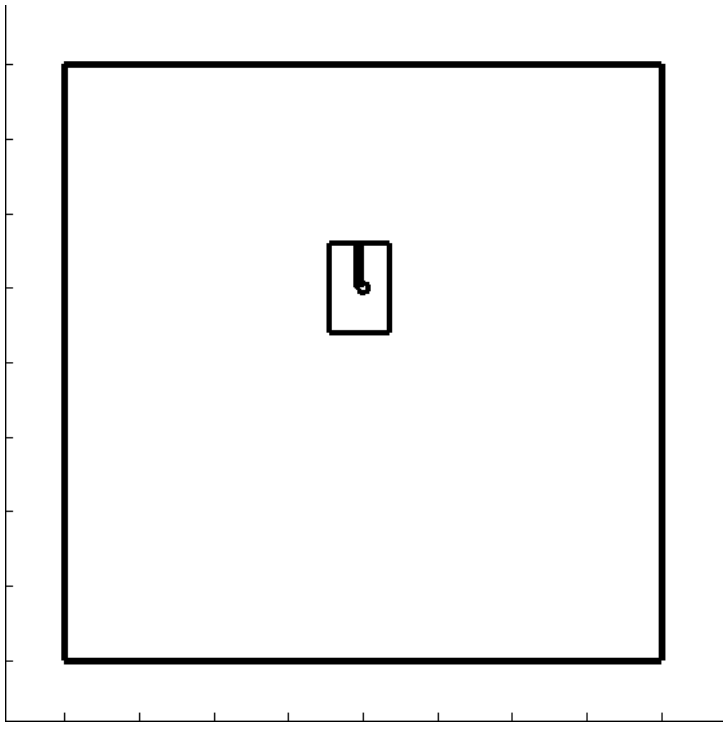
A sample fixed point reference to investigate the performance of the control law given by (3.12) was used in these simulations. A variety of values for the control gains k_{1p} and k_{2p} in that control law were simulated and the final parking errors were recorded for each case.

The initial and final poses of the nonholonomic mobile robot along with the fixed point reference are shown in Fig. 3.4. The little circle is the reference fixed position input to the robot.

The final parking errors for each value of the control gains, k_{1p} and k_{2p}, are given in Table 3.2.



(a)



(b)

Figure 3.4: Parking scenario: (a)Initial pose (b)Desired final pose

Table 3.2: Final parking errors for different values of control gains

$\mathbf{k_{1p}}$	$\mathbf{k_{2p}}$	$\tilde{\mathbf{x}}_\mathbf{p}(m)$	$\tilde{\mathbf{y}}_\mathbf{p}(m)$	$\tilde{\theta}_\mathbf{p}(rad)$
1	1	0.019550	0.002290	0.00120000
1	6	0.394650	0.000440	0.01341800
1	11	0.402980	0.001440	0.00861600
1	16	0.403520	0.001730	0.00618200
1	21	0.403260	0.001860	0.00479900
1	26	0.402920	0.001920	0.00391600
1	31	0.402620	0.001960	0.00330500
6	1	0.045049	0.000079	0.00128000
6	6	0.394560	0.003760	0.01341500
6	11	0.404750	0.002630	0.00869200
6	16	0.405080	0.001920	0.00623000
6	21	0.404550	0.001500	0.00483000
6	26	0.404000	0.001230	0.00393700
6	31	0.403540	0.001040	0.00332000
11	1	0.046686	0.000072	0.00129000
11	6	0.394630	0.004480	0.01342000
11	11	0.404810	0.003050	0.00869500
11	16	0.405120	0.002210	0.00623200
11	21	0.404580	0.001720	0.00483000
11	26	0.404030	0.001400	0.00393700
11	31	0.403560	0.001180	0.00332000
16	1	0.047077	0.000069	0.00129000
16	6	0.394650	0.004740	0.01342200
16	11	0.404820	0.003200	0.00869600
16	16	0.405130	0.002310	0.00623200
16	21	0.404590	0.001790	0.00483000

Continued on next page

Table 3.2 – continued from previous page

$\mathbf{k_{1p}}$	$\mathbf{k_{2p}}$	$\tilde{\mathbf{x}}_\mathbf{p}(m)$	$\tilde{\mathbf{y}}_\mathbf{p}(m)$	$\tilde{\theta}_\mathbf{p}(rad)$
16	26	0.404030	0.001460	0.00393700
16	31	0.403560	0.001230	0.00332000
21	1	0.047230	0.000067	0.00129000
21	6	0.394660	0.004880	0.01342200
21	11	0.404830	0.003280	0.00869600
21	16	0.405130	0.002360	0.00623200
21	21	0.404590	0.001830	0.00483100
21	26	0.404030	0.001490	0.00393700
21	31	0.403560	0.001260	0.00332000
26	1	0.047305	0.000066	0.00129000
26	6	0.394660	0.004960	0.01342300
26	11	0.404830	0.003330	0.00869600
26	16	0.405130	0.002390	0.00623200
26	21	0.404590	0.001860	0.00483100
26	26	0.404030	0.001510	0.00393700
26	31	0.403560	0.001280	0.00332000
31	1	0.047349	0.000065	0.00129000
31	6	0.394670	0.005020	0.01342300
31	11	0.404830	0.003360	0.00869600
31	16	0.405130	0.002420	0.00623200
31	21	0.404590	0.001870	0.00481300
31	26	0.404030	0.001530	0.00393700
31	31	0.400356	0.001290	0.00332000

Since this thesis is essentially on modeling and control of coordinated motion, the control problems of nonholonomic mobile robots will not be discussed any further. The simulation results presented above are promising. In the developed models of Chapter 4 and Chapter 5, the control laws given by (3.9) and (3.12) will be used appropriately.

Chapter 4

Dynamic Coordination Model

Coordinated motion of a group of autonomous mobile robots requires each member of the group to follow specific trajectories that are dependent on the motions of the other members of the group. Hence, *coordinated motion of a group of autonomous mobile robots will be modeled by the generation of a reference trajectory for each member in the group, that is dependent on the positions and orientations of some or all of the others.* Besides, the generated reference trajectories should enable achievement of a possible goal defined for the group. A direct result of the definition of such a goal is the fact that; the generated models of coordinated motion and coordinated task manipulation are scenario dependent; i.e. models for different coordinated tasks will be different.

Throughout the rest of this thesis, models for the coordinated motion of a group of autonomous nonholonomic mobile robots will be developed. Since the generated model is scenario dependent, the scenario described in Section 1.4 will be used. However, the developed model will be easy to modify for other coordinated tasks.

In this chapter, coordinated motion of a group of autonomous nonholonomic mobile robots is defined and modeled by the definition of forces; hence the title of the chapter. However, the control of a nonholonomic mobile robot is based on its kinematic model as described in Chapter 3. To achieve coordinated motion using forces and dynamics, a *virtual reference model* that in turn implies online collision-free trajectories for the autonomous robots is introduced. The model consists of virtual references - which move under the effect of coordination forces between the robot and the others in the group, and an attraction force between the robot and the target - for each autonomous nonholonomic mobile robot. Then, actual non-

holonomic mobile robot is forced to track the reference trajectory generated by the virtual reference model by implementing the controllers described in Chapter 3.

The problem of modeling the coordinated motion of a group of autonomous non-holonomic mobile robots is attacked by the hierarchical approach depicted in Fig. 4.1. Note that there's a feedback path to compensate for any disturbance such as weak performance of the low-level controller, any variations in the parameters of the robot's models, etc. between the actual poses of the autonomous mobile robots and the virtual reference system, implying online generation of reference trajectories for the robots.

The rest of the chapter describes the components of the *virtual reference model* and essential factors contributing to the generation of reference trajectories.

4.1 Virtual Reference System

There are several possibilities to model the virtual reference system for the robots. However, the generated virtual reference system should be kept as simple as possible for easier design and manipulation of the inner dynamics.

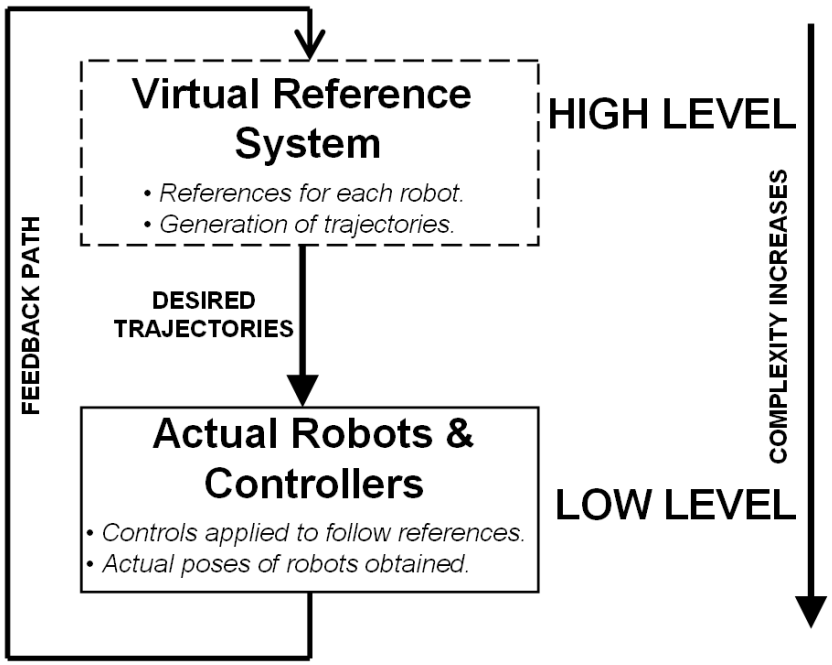


Figure 4.1: Hierarchical approach of dynamic coordination model for the group

Electrical charges can be used as virtual references for the autonomous robots along with *electrostatic forces* that model the forces responsible for coordination and target attraction. In such a case, the mutual forces between two charged objects i and j are given by:

$$\mathbf{F}_{ij} = k_e \frac{q_i q_j}{d_{ij}^2} \mathbf{r}_{ij} \ , \tag{4.1}$$

where $k_e > 0$ is a constant, q_i and q_j are the signed charges of i and j, d_{ij} is the absolute distance between i and j, whereas $\mathbf{r}_{ij}$ is the vector from i to j, and $\mathbf{F}_{ij}$ is the force exerted on j by i. The virtual reference system generated for a group of three autonomous mobile robots using electrical charges and forces is depicted in Fig. 4.2(a).

Gravitational forces can also be used to model such forces whereas *masses* are used as the virtual references for the autonomous robots and the target. In such a case, the forces between two virtual reference objects i and j are given by:

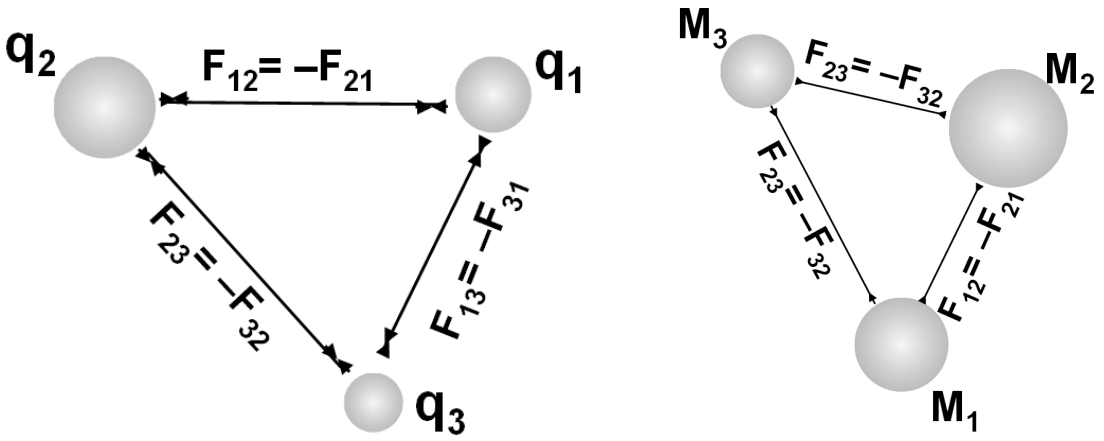
$$\mathbf{F}_{ij} = G_g \frac{M_i M_j}{d_{ij}^2} \mathbf{r}_{ij} \ , \tag{4.2}$$

where $G_g > 0$ is a constant, $M_i > 0$ and $M_j > 0$ are the masses of the virtual references, d_{ij} is the absolute distance between i and j, whereas $\mathbf{r}_{ij}$ is the vector from i to j, while $\mathbf{F}_{ij}$ is the force exerted by j on i. Note that $\mathbf{F}_{ij}$ acts only as an *attractive* force. The virtual reference system generated for a group of three autonomous mobile robots using gravitational attraction forces is depicted in Fig. 4.2(b).

4.1.1 Virtual Masses

The virtual reference model proposed in this work consists of *virtual masses* that are interconnected via virtual springs and dampers; i.e. *virtual bonds*. The well-known dynamics of such a system facilitates the generation of reference trajectories.

The virtual mass-spring-damper model proposed in this work is analogous to the molecules formed by atoms that are tied by chemical bonds (see Fig. 4.3). The relative positions and orientations of the atoms in such a molecule are almost constant when the molecule is moving. Similarly, the reference virtual masses should



(a) (b)

Figure 4.2: Possibilities for virtual reference systems: (a)Electrostatic (b)Gravitational

be at certain positions with respect to each other when the entire system is moving towards a target.

The forces generated on the *virtual bonds* are responsible for the coordinated motion of the virtual masses. The attraction of the masses to the target, T, is defined on the basis of a virtual bond between each virtual mass and T. Positions and velocities of the virtual masses moving under the effect of these forces are then input as references to the low level controllers of the autonomous robots as illustrated in Fig. 4.1. Virtual masses in the virtual reference system can be modeled either as point particles or as finite size geometric shells, depending on the requirements.

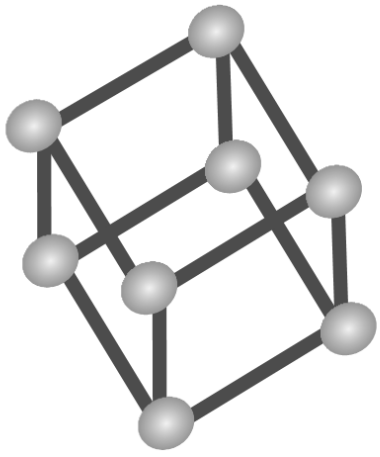


Figure 4.3: Analogy to a molecule where atoms are tied by chemical bonds

Point Particles

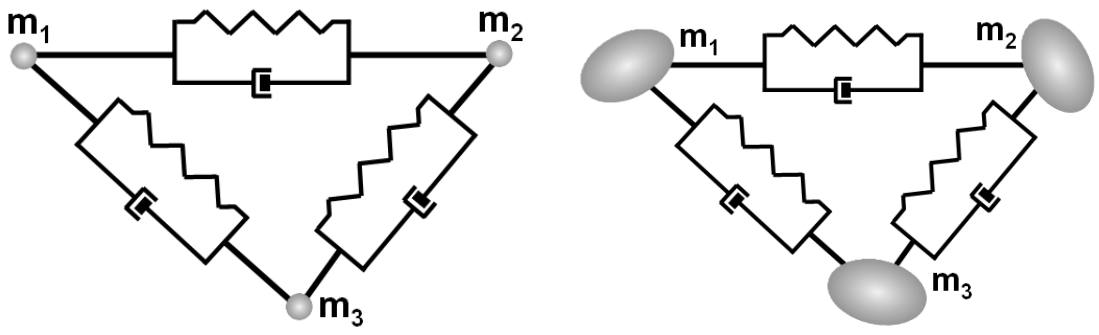
A point particle of finite mass that is connected to other members of the group and the target via virtual bonds as described above can be used to generate a reference trajectory for each of the nonholonomic mobile robots.

Since a point particle is holonomic, this approach relaxes the nonholonomic constraint. In other words, forces can cause point particles to move in any direction. On the other hand, orientation is not defined for point particles. Instead, the orientation reference will be obtained from the velocity of the point particle. A virtual reference system generated for a group of three autonomous nonholonomic mobile robots is shown in Fig. 4.4(a).

Geometric Shells

Geometric shells, i.e. circular or elliptical in 2D and spherical or ellipsoidal in 3D, of finite size and mass can also be used as references for the robots. In this case, the system becomes more complicated. However, the reference orientations exist in the model since orientations of finite-size shells can be defined. This approach is depicted in Fig. 4.4(b) for a group of three robots.

In this work, point masses are used to create virtual references for the robots. Since there are n robots, n virtual masses, $(m_1, m_2, ..., m_{n-1}, m_n)$, are generated on the computer. The dynamics of each m_i will be dictated by the virtual forces exerted on it by the virtual bonds connected between itself and other masses or the target.



(a) **(b)**

Figure 4.4: Possible virtual masses: (a)Point particles (b)Geometric shells

4.1.2 Virtual Forces

A biologically inspired coordination scheme where virtual bonds are constructed between each m_i and its two closest neighbors as depicted in Fig. 4.5 is used in this work. In such a scenario, virtual bonds are not constructed between m_i and m_p, for the calculation of forces that generate the dynamics of m_i.

The closest two neighbors of m_i exert a force on it via the virtual bonds to keep a certain distance between themselves and m_i. Physically, this distance can be interpreted as the equilibrium length of the springs that form the virtual bonds.

Let m_j be the closest and m_k be the second closest neighbors of m_i. Then, the coordination force exerted on m_i by m_j and m_k, is defined as:

$$\begin{aligned}\mathbf{F}_{coord} = &-\left[k_{coord}(d_{i2j} - d_{coord}) + c_{coord}((\dot{\mathbf{X}}_i - \dot{\mathbf{X}}_j) \bullet \mathbf{n}_{i2j})\right]\mathbf{n}_{i2j} \\ &-\left[k_{coord}(d_{i2k} - d_{coord}) + c_{coord}((\dot{\mathbf{X}}_i - \dot{\mathbf{X}}_k) \bullet \mathbf{n}_{i2k})\right]\mathbf{n}_{i2k}\end{aligned}, \tag{4.3}$$

where $\bullet$ denotes vector dot product, k_{coord} and c_{coord} are the coefficients of the spring and damper, respectively, d_{i2j} is the signed distance between m_i and m_j, $\mathbf{n}_{i2j}$ is the unit vector from m_j to m_i, $\dot{\mathbf{X}}_i = \begin{bmatrix} \dot{x}_i & \dot{y}_i \end{bmatrix}^t$ is the velocity vector of virtual mass m_i, $\dot{\mathbf{X}}_j = \begin{bmatrix} \dot{x}_j & \dot{y}_j \end{bmatrix}^t$ is the velocity vector of virtual mass m_j, d_{i2k} is the signed distance between m_i and m_k, $\mathbf{n}_{i2k}$ is the unit vector from m_k to m_i, $\dot{\mathbf{X}}_k = \begin{bmatrix} \dot{x}_k & \dot{y}_k \end{bmatrix}^t$ is the velocity vector of virtual mass m_k, and d_{coord} is the coordination distance to be maintained among the masses.

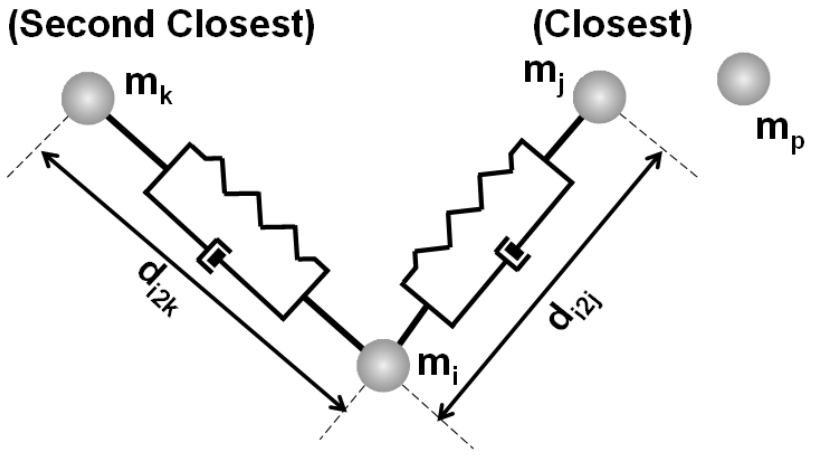


Figure 4.5: Closest two neighbors under interest for definition of coordination force

Similarly, the force exerted on each m_i by the target, T, is modeled as the sum of a spring force with constant k_{targ} and a damping force with constant c_{targ}. In other words, virtual bonds are also constructed between the target and the virtual masses.

The force exerted by T on m_i, to keep m_i at distance d_{targ} from T, is defined as:

$$\mathbf{F}_{targ} = -\left[k_{targ}(d_{i2T} - d_{targ}) + c_{targ}(\dot{\mathbf{X}}_i \bullet \mathbf{n}_{i2T})\right]\mathbf{n}_{i2T} \ , \tag{4.4}$$

where $\bullet$ denotes vector dot product, k_{targ} and c_{targ} are the coefficients of the spring and damper, respectively, d_{i2T} is the signed distance between m_i and T, $\dot{\mathbf{X}}_i = \begin{bmatrix} \dot{x}_i & \dot{y}_i \end{bmatrix}^t$ is the velocity vector of virtual mass m_i, $\mathbf{n}_{i2T}$ is the unit vector from m_{cl2} to m_i, and d_{targ} is the distance to be maintained to T.

The total force on m_i is the sum of $\mathbf{F}_{coord}$ in (4.3) and $\mathbf{F}_{targ}$ in (4.4). Consequently, the dynamics of the virtual mass m_i is given by:

$$m_i \begin{bmatrix} \ddot{x}_i \\ \ddot{y}_i \end{bmatrix} = \mathbf{F}_{coord} + \mathbf{F}_{targ} \ . \tag{4.5}$$

The position vector of m_i, $\mathbf{X}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^t$ that is obtained from the solution of (4.5) is used as the reference position for the actual robot, R_i. On the other hand, the reference orientation for R_i is extracted from $\dot{\mathbf{X}}_i = \begin{bmatrix} \dot{x}_i & \dot{y}_i \end{bmatrix}^t$, the velocity vector of m_i, which is also obtained from the solution of (4.5) as follows:

$$\theta_i = \arctan\left(\dot{y}_i / \dot{x}_i\right) \ . \tag{4.6}$$

4.2 Adaptable Model Parameters

Coordinated task manipulation by a group of mobile robots is defined as the accomplishment of a specified task together in certain formations. The necessary formation may vary based on the specifications of the coordinated task.

To achieve coordinated tasks by the achievement of certain formations, adaptable springs and dampers between virtual masses might be required. For example, the

equilibrium lengths of the virtual springs should be changed when the robots are near the target to achieve a uniform distribution on the final formation. In the specified scenario of this work, the coordinated task is split into two main phases:

- (1) Approaching T starting from an initial setting.
- (2) Achieving a uniform circular formation with radius d_{targ} with T at the center, towards which all robots headed.

In Phase (1), i.e. the virtual masses approach the target from some initial formation, the priority is given to the coordinated motion of the virtual masses. $\mathbf{F}_{targ}$ acts on the virtual masses and attracts all of them towards T. However, $\mathbf{F}_{coord}$ in this phase is dominant so they move together. As long as the distance between m_i and T remains greater than a certain value d_{break}, below which the achievement of final formation will be of higher interest, k_{coord} is set to k_{far}, which is higher than k_{targ} to maintain this dominance. $c_{coord} > 0$ in this phase so that the virtual masses are forced to move in the same direction.

When the distance between the virtual mass m_i and T is lower than d_{break}, it enters Phase (2). Constants of coordination forces are decreased in this phase. Moreover, k_{coord} is decreased to k_{near}, a lower value than k_{targ}; hence the dominance of $\mathbf{F}_{targ}$ and achievement of the final formation are enabled. Since the main property of the final formation is to keep m_i at distance d_{targ} from T for all virtual masses, the constraint dictated by $\mathbf{F}_{coord}$ should be relaxed. For example, c_{coord} is reduced to zero so that m_i need not move in the same direction with its neighbors in this phase. The equilibrium length of the spring between the masses is changed from the initial value of d_{far} to the final value of d_{near}.

It follows from *Law of Cosines* that, d_{near} is given as in (4.7) for a uniform placement of n masses on a circle of radius d_{targ} as illustrated in Fig. 4.6.

$$d_{near} = d_{targ}\sqrt{2(1-\cos(2\pi/n))}\,. \tag{4.7}$$

When m_i is closer than d_{break} to T, it performs circular motion because $\mathbf{F}_{targ}$ acts on it in the radial direction. k_{coord} is reduced to some finite value, k_{near}, rather than zero so that the virtual masses are forced to distribute on the formation circle

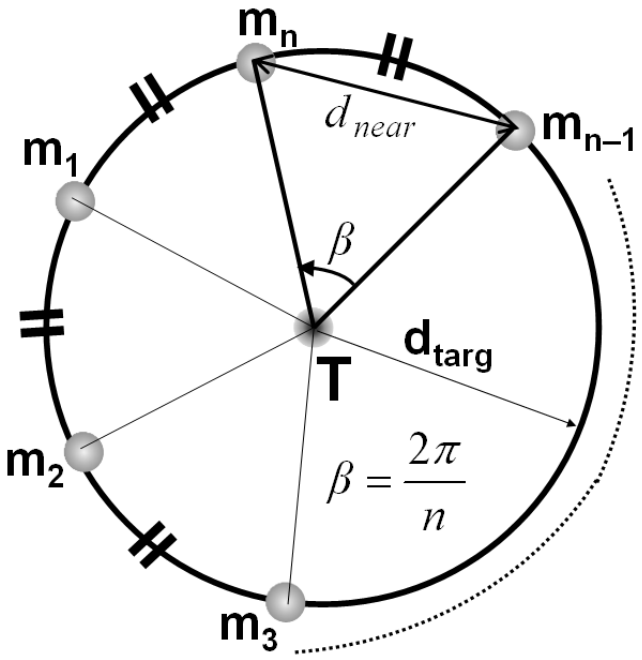


Figure 4.6: Uniform distribution of masses on the formation circle around T

uniformly. Otherwise, m_i would stop when it achieves its distance from T equal to d_{targ}. For $\mathbf{F}_{targ}$ to be the dominant force, k_{near} must be less than k_{targ} in this phase.

k_{coord} has to be reduced as a smooth continuous function of the distance of m_i to T. This results in a smooth change in $\mathbf{F}_{coord}$ and discontinuity is avoided. Otherwise the discontinuity in $\mathbf{F}_{coord}$ might yield a big jump in the velocity of m_i, and R_i may not be able to follow such a change due to its nonholonomic nature. To enable a smooth change in $\mathbf{F}_{coord}$, k_{coord} is modeled by the sigmoid function defined by:

$$k_{coord} = k_{near} + \frac{k_{far} - k_{near}}{1 + \exp(\mu(d_{break} - d_{i2T} + \phi))} , \tag{4.8}$$

where μ and ϕ are positive constants of the sigmoid function, d_{i2T} is the signed distance between m_i and T, k_{far} and k_{near} are the spring coefficients used in Phase 1 and Phase 2, respectively. The change of k_{coord} versus d_{i2T} is depicted in Fig. 4.7.

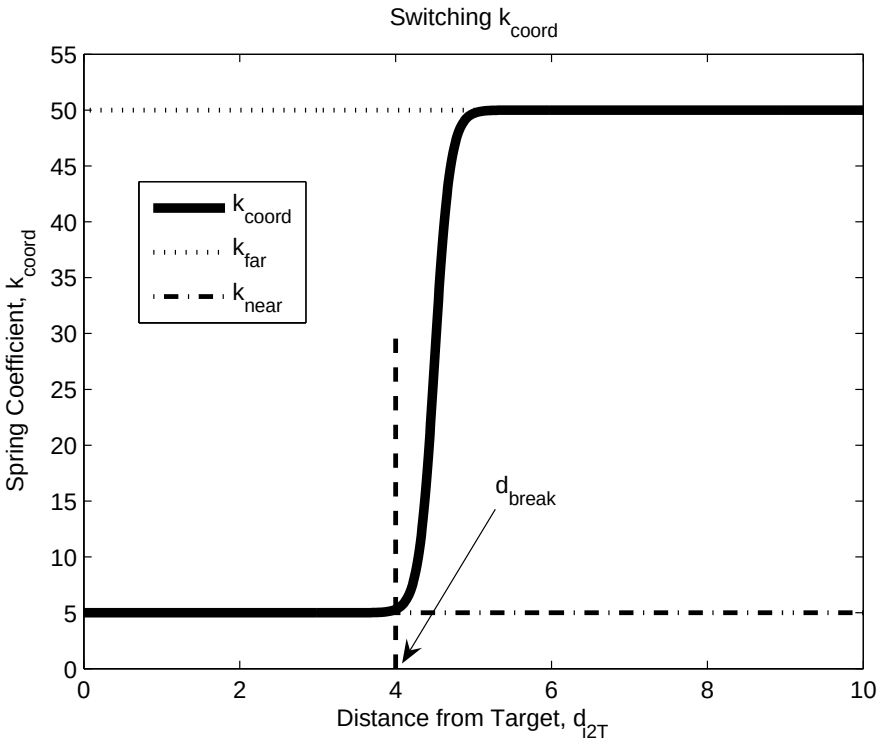


Figure 4.7: Adaptive spring coefficient, k_{coord} vs distance from T, d_{i2T}

4.3 Collision Avoidance by Velocity Update

As discussed in Chapter 1, collision avoidance is one of the essential problems in modeling coordinated motion of a group of autonomous mobile robots. If the robots are nonholonomic, avoiding collisions is even more complicated, since they cannot arbitrarily change their orientations. In this section, a simple algorithm to avoid collisions is proposed. In this work, neither static obstacles are considered nor T is considered as an obstacle since $\mathbf{F}_{targ}$ keeps the robot at distance d_{targ} from T in Phase (2), when R_i is around T.

The developed algorithm uses sensory information coming from the robots to predict collisions ahead of time, updates the velocities of the virtual masses to avoid the possible collisions *online*. For each R_i, we define a *virtual collision prediction region(VCPR)*, Ω_i, given by a circular arc of radius r_{coll} and angle θ_{coll}, symmetric with respect to its velocity as depicted in Fig. 4.8.

R_i, detects a collision risk when any of the other members of the group touches its virtual region, Ω_i. In case of such a collision prediction, the velocity of m_i is updated

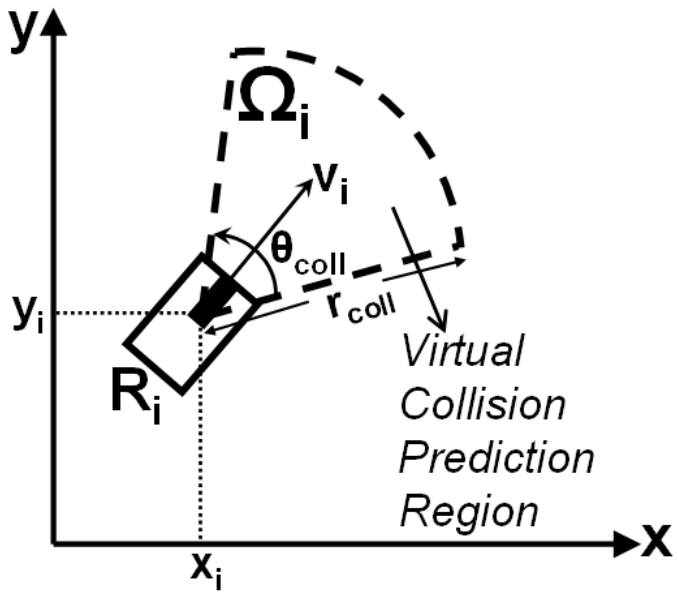


Figure 4.8: Virtual collision prediction region($VCPR$), Ω_i, for the robot, R_i

to safely avoid the collision. After the velocity of m_i is updated, it moves with that constant velocity for a user-defined short period of time. The forces described above start to act on m_i after this time delay. This delay enables avoidance of the collision with the robot sensed as an obstacle.

The final velocity of m_i after the velocity update is designed based on the relative velocity of the sensed robot, R_j, with respect to R_i. The coordinate frame attached to R_i and two parts, R and L, as shown in Fig. 4.9 are generated for the collision avoidance algorithm. R_i checks if any of the sensed robots touches Ω_i; i.e. if there's a collision risk, at each computational step. If a collision risk is detected, the following algorithm is run:

1. Calculate the component of the velocity of R_j projected on the axis y_i.
2. Rotate counter-clockwise if that component is positive.
3. Rotate clockwise if that component is negative.
4. If that component is zero:
    - Rotate counter-clockwise if R_j touches the region R.
    - Rotate clockwise if R_j touches the region L.

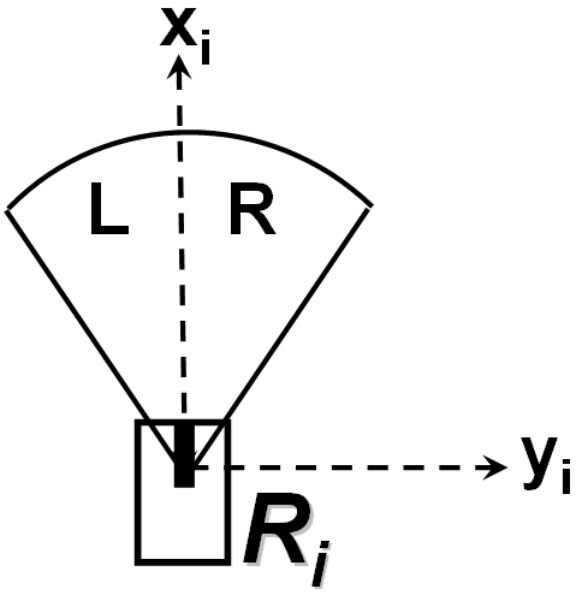


Figure 4.9: R_i's coordinate frame and parts of $VCPR$

The collision avoidance algorithm described above was implemented as two sequential functions:

1. Collision prediction if R_i senses another robot, R_j.

2. If a collision with R_j is predicted, update the velocity to avoid collisions.

Note that the second function is run if and only if the result of the first function is positive; i.e. R_i predicts a collision with R_j. The details of these functions are explained in the following subsections.

4.3.1 Collision Prediction Algorithm

If R_i with position vector $\mathbf{X}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^t$ and orientation $0 < \theta_i < 2\pi$ senses another robot R_j, it runs the following algorithm to predict any possible collision:

1. Extract the position vector and orientation of the sensed robot, R_j, from the sensory information.

   $\Rightarrow \mathbf{X}_j = \begin{bmatrix} x_j & y_j \end{bmatrix}^t$ and θ_j.

2. Calculate the coordinates of a specific number of points on the edges of R_j.

   $\Rightarrow$ Points of the form $\mathbf{e}_j = \begin{bmatrix} e_{jx} & e_{jy} \end{bmatrix}^t$.

3. For each of the calculated edge points; $\mathbf{e}_j$:

   - Calculate the vector from the center of mass of R_i to $\mathbf{e}_j$.

     $\Rightarrow \mathbf{r}_{i2j} = \mathbf{e}_j - \mathbf{X}_i$.

   - If R_j is close enough for a collision risk; $|\mathbf{r}_{i2j}| \leq r_{coll}$:

     – Calculate the limiting angles of Ω_i.

       $\Rightarrow \theta_{min} = \theta_i - (r_{coll}/2)$,

       $\Rightarrow \theta_{max} = \theta_i + (r_{coll}/2)$.

     – Calculate the angle of $\mathbf{r}_{i2j}$.

       $\Rightarrow \theta_{i2j} = \arctan\left((e_{jy} - y_i) \,/\, (e_{jx} - x_i)\right)$.

     – If $\mathbf{r}_{i2j}$ is inside Ω_i; $\theta_{min} < \theta_{i2j} < \theta_{max}$:

       * A collision risk with R_j exists.

       * Run the *velocity update* algorithm.

4.3.2 Velocity Update Algorithm

If a collision is predicted by the *collision prediction algorithm*, run the following algorithm to decide direction of velocity update to avoid the possible collision:

1. Calculate the angular distances to the limits of Ω_i.

   $\Rightarrow \alpha_{max} = |\theta_{i2j} - \theta_{max}|$,

   $\Rightarrow \alpha_{min} = |\theta_{i2j} - \theta_{min}|$.

2. Decide the default velocity update direction.

   - If the angle to θ_{min} is smaller; $\alpha_{min} < \alpha_{max}$:

     – The default direction is a counter-clockwise rotation of the velocity of m_i by an angle of α_{min}.

   - If the angle to θ_{max} is smaller; $\alpha_{min} \geq \alpha_{max}$:

     – The default direction is a clockwise rotation of the velocity of m_i by an angle of α_{max}.

3. Check the direction of R_j for the final decision.

   - If R_j moves towards the default direction of velocity update obtained in step 2:

  - The direction of velocity update is opposite to the default direction.

- If R_j moves in the opposite direction to the default direction of velocity update in step 2:
  - The direction of velocity update is the default direction.

In the light of the above described algorithms, the following examples illustrate the dependence of the velocity update direction on the velocity of R_j.

Example-1

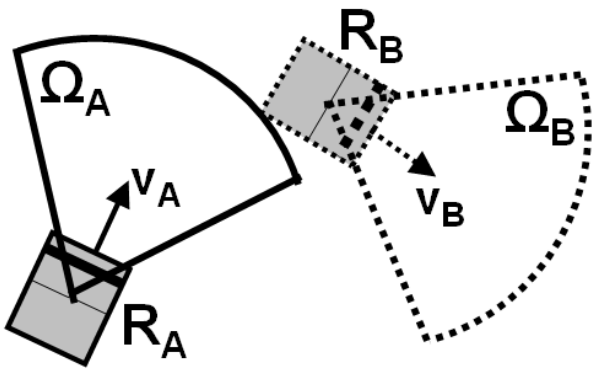
Since R_B hits the virtual arc, Ω_A, R_A predicts a collision in Fig. 4.9(a). In this case, disregarding the dynamics given by (4.5), the velocity of m_A - hence the reference for R_A - is updated depending on the relative velocity of R_B with respect to R_A. The shortest rotational path to avoid the predicted collision in this case is to change the orientation of R_A in counter-clockwise direction until R_B loses contact with Ω_A. However, the velocity of R_B should also be considered. Since R_B is moving to the right, the counter-clockwise rotation avoids the predicted collision and the path of R_A for the time delay described above is free of collisions with R_B. The velocity of m_B is not changed since R_B doesn't predict a collision.

Example-2

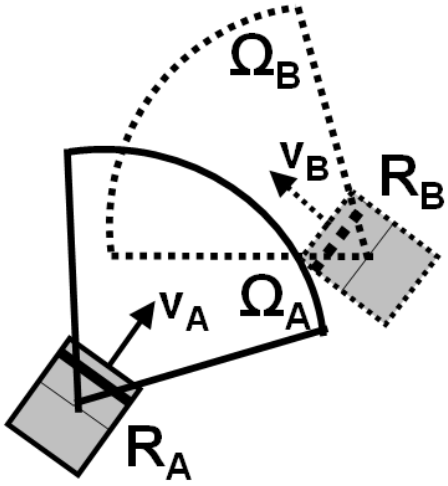
Since R_B hits the virtual arc, Ω_A, R_A predicts a collision in Fig. 4.9(b). The easiest way of avoiding the predicted collision in this case is to change the orientation of R_A in counter-clockwise direction until R_B loses contact with Ω_A. This would be useful if R_B were stationary. However, since R_B is moving to the left, it will be hitting Ω_A again just after the predicted collision is avoided. To avoid this occurrence, the orientation of R_A is changed in clockwise direction, taking the relative velocity of R_B with respect to R_A into account. The velocity of m_B is not changed since R_B doesn't predict a collision at the shown instant.

4.4 Controller Switching

When the virtual masses and therefore robots are uniformly distributed around T on a circle of radius d_{targ} based on the framework given above, the last maneuver



(a)



(b)

Figure 4.10: Collision avoidance examples explaining relative velocity dependence: (a)Example-1 (b)Example-2

that robots should do is to orient themselves towards the target.

The reference trajectories generated by the positions of the virtual masses are tracked by the control law given by (3.9). After the virtual masses are fully distributed on the circle of radius d_{targ} around T, the control law of (3.12) is used to park the robots on the circle with necessary orientations so that they head towards T. The fixed-point reference position for R_i to park at is used as the position of m_i on the circle, whereas the reference orientation for R_i to be headed towards T is calculated artificially for that specific reference position.

Chapter 5

Kinematic Coordination Model

Coordinated motion of a group of autonomous nonholonomic mobile robots is usually modeled and controlled on the basis of virtual forces. In this approach, the reference linear and angular velocities are derived from the reference position and orientation obtained by solving the dynamic equations of the system. The desired kinematics of a "*virtual reference robot*" so obtained is given as a reference to the actual robot. This method fails to take the nonholonomic constraint into consideration during the generation of reference trajectories for a group of autonomous nonholonomic mobile robots aimed for a specific task. The actual robot might not be able to follow the virtual reference robot if abrupt changes in the reference position and orientation occur.

In this chapter, coordinated motion of a group of autonomous nonholonomic mobile robots is modeled and controlled by proper selection of the linear and angular velocities of the virtual reference robot; hence the title of the chapter. The nonholonomic constraint is taken into consideration from the beginning; so the generated reference trajectories are named *nonholonomic*. Consequently, the regulation of the errors between the virtual reference robot's pose and the actual robot's pose to zero are guaranteed with an appropriate feedback controller. Once the reference velocities that will yield coordinated motion are obtained, they are integrated to get the reference positions and orientations of the robots. The first section of the chapter is devoted to explain the details of such a trajectory generation.

The hierarchical approach, depicted in Fig. 5.1 for a single robot, is used in this model. Note that a closed loop is formed between the *virtual reference robots* and the actual robots, implying the online nature of reference velocity generation.

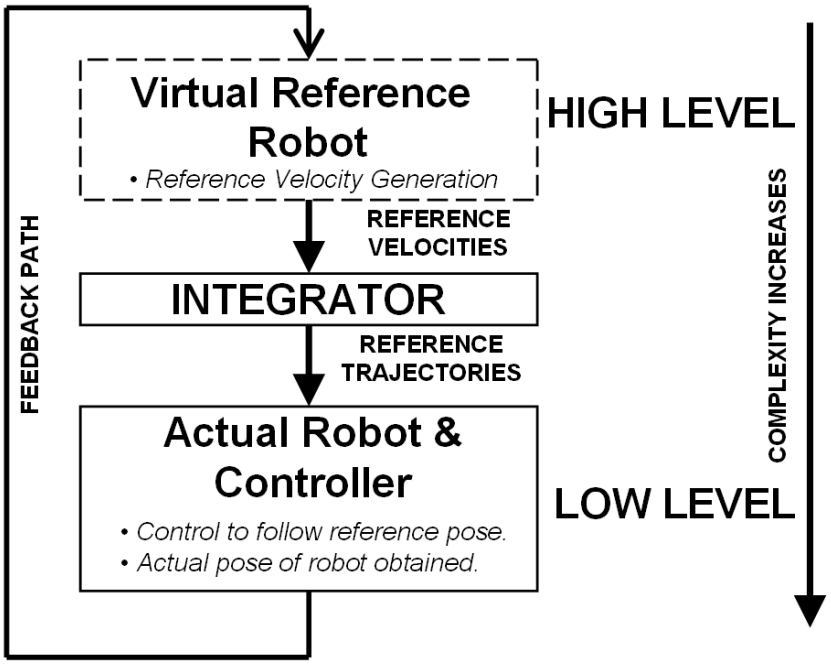


Figure 5.1: Hierarchical approach of kinematic coordination model for a single robot

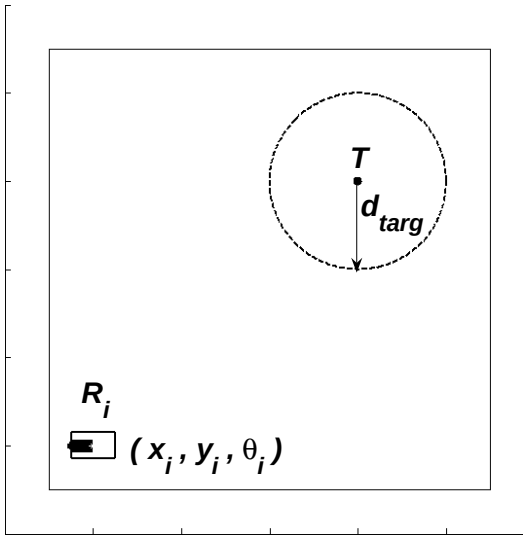
5.1 Kinematic Reference Generation

The proposed model generates kinematic references instead of dynamic references; i.e. reference linear and angular velocities instead of reference trajectories. For the sake of simplicity in analysis, possibilities about the generation of such references will be investigated for a single robot in this section. Coordinated motion and coordinated task manipulation by a group of autonomous nonholonomic mobile robots requires each member in the group to achieve certain poses so that the necessary formations can be maintained by the entire group. The desired pose of each R_i is dependent on the specified scenario.

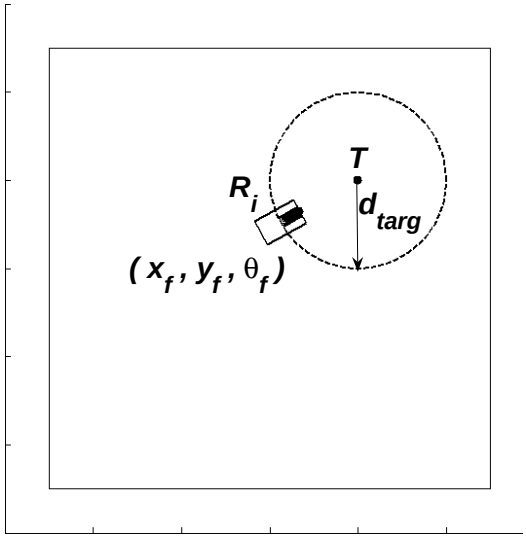
In this section, the scenario depicted in Fig. 5.2(a) is used as a test bed to investigate the performance of different approaches to the problem of generating reference velocities. The desired final configuration is shown in Fig. 5.2(b).

The pose errors, e_x, e_y and e_θ, are defined as:

$$\begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} x_f \\ y_f \\ \theta_f \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}, \tag{5.1}$$



(a)



(b)

Figure 5.2: Scenario for analysis: (a)Initial pose (b)Desired pose

where $\mathbf{X}_i = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^t$ is the current pose of R_i, and $\mathbf{X}_f = \begin{bmatrix} x_f & y_f & \theta_f \end{bmatrix}^t$ is the desired final pose with distance d_{targ} to T and orientation equal to the angle of the vector from R_i to T makes with the axis of abscissas.

The reference velocities are then generated as functions of the errors in position and orientation of the robot. The approach is similar to the proportional velocity controllers in control theory in the sense that reference angular velocity is modeled proportional to the error in orientation and reference linear velocity is defined

proportional to the error in position. The reference velocities may be generated in a variety of ways. Two essential examples of these possibilities are explained and compared in the sequel.

5.1.1 Discontinuous Linear Velocity Reference

For the given scenario, R_i should correct its orientation to head towards T, and maintain distance d_{targ} from T. This can be achieved by the introduction of a discontinuity in the linear velocity. In this approach, R_i corrects its orientation to some limiting value, θ_{tol}, before it starts moving towards T. After the error in orientation falls below θ_{tol}, it starts moving towards T with a linear velocity proportional to its distance to T, namely d_{i2T}.

To achieve the correct orientation, the reference angular velocity, u_{2r}, is always calculated as:

$$u_{2r} = k_1 e_\theta \ , \tag{5.2}$$

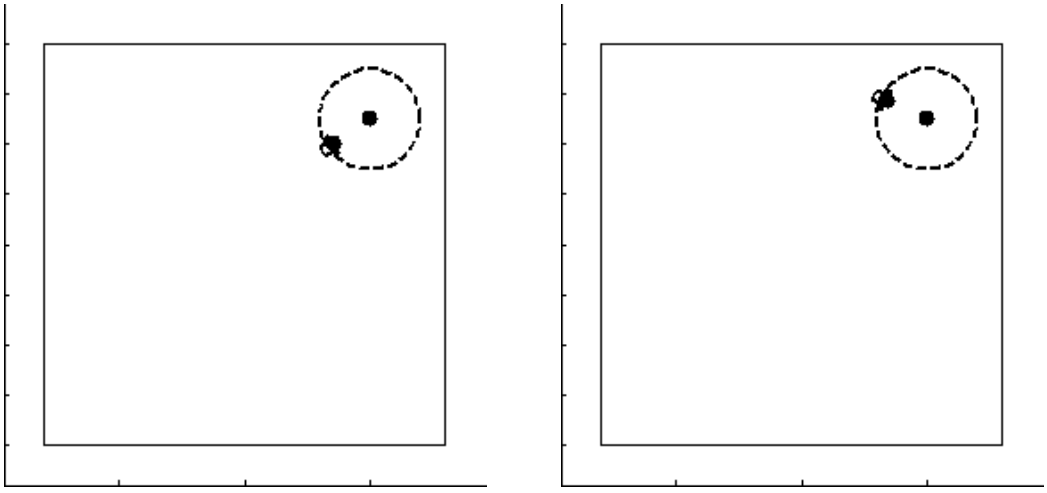
where $k_1 > 0$ is the proportionality constant and e_θ is the error in orientation.

On the other hand, the linear velocity reference, u_{1r}, is given by:

$$u_{1r} = \begin{cases} 0, & \text{if} \quad |e_\theta| \geq \theta_{tol} \\ k_2 \left(d_{i2T} - d_{targ} \right), & \text{if} \quad |e_\theta| < \theta_{tol} \end{cases} \ , \tag{5.3}$$

where $k_2 > 0$ is the proportionality constant, θ_{tol} is the limiting orientation error below which R_i moves towards T, d_{i2T} is the signed distance between R_i and T, and d_{targ} is the distance to be maintained from T.

The system was simulated for two different values of θ_{tol}. Proper selection of the constants k_1 and k_2 results in achievement of the specified task as seen in Fig. 5.3(a) and Fig. 5.3(b). The final configurations is not the same in these two cases as expected. Despite the fact that the generated reference velocities suffice to achieve the desired configuration in the simulations, the discontinuity introduced by the piecewise definition of u_{1r} is problematic. Due to this discontinuity, the reference linear velocity exhibits high-frequency switching behavior. The reference velocities generated for these cases are plotted in Fig. 5.4 and Fig. 5.5.



(a) **(b)**

Figure 5.3: Discontinuous linear velocity final poses: (a)Low tolerance, $\theta_{tol} = 0.0873$ (b)High tolerance, $\theta_{tol} = 0.5236$

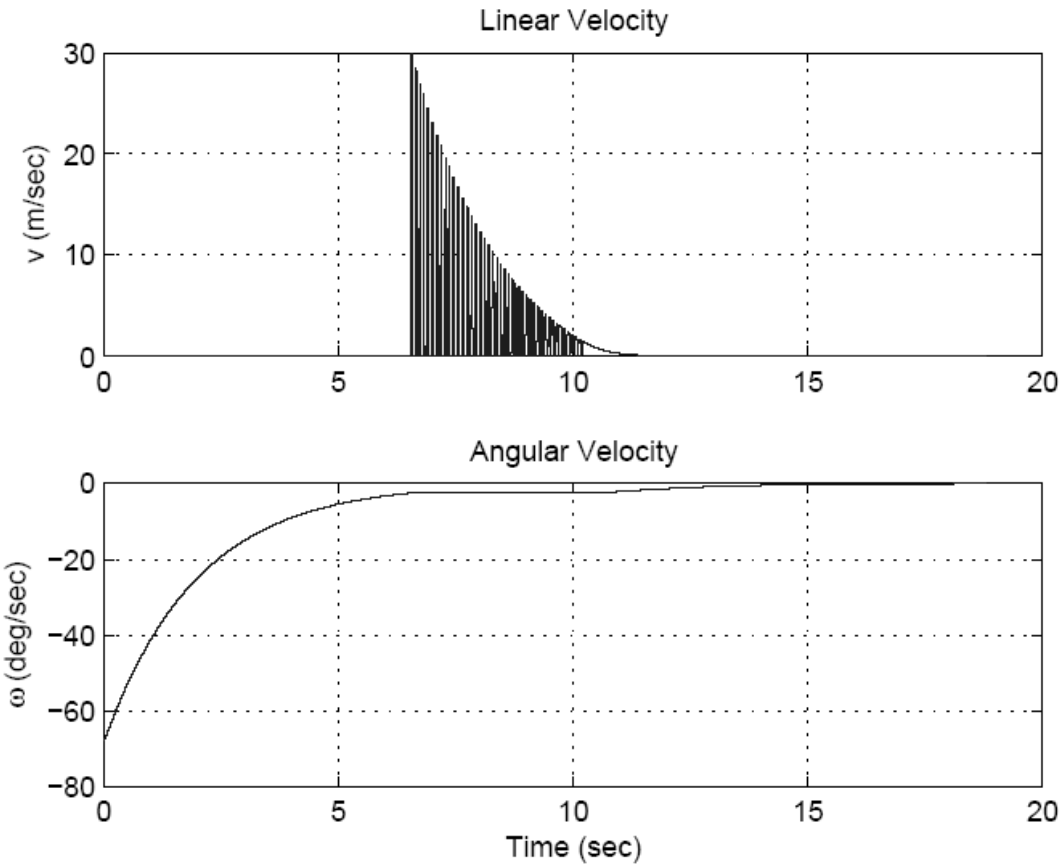


Figure 5.4: Discontinuous reference velocities with low tolerance, $\theta_{tol} = 0.0873$

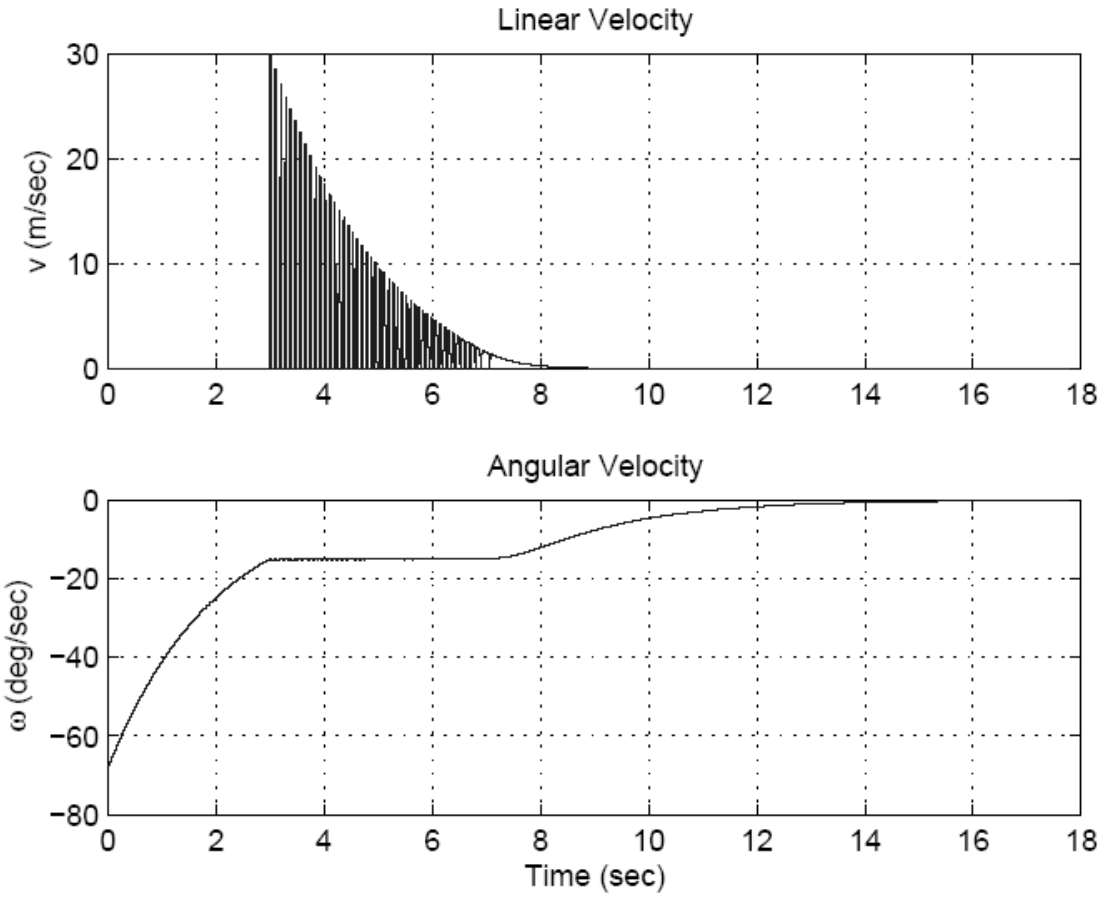


Figure 5.5: Discontinuous reference velocities with high tolerance, $\theta_{tol} = 0.5236$

A direct comparison between these results reveals that a higher value of θ_{tol} causes earlier start of linear motion. However, the time for the overall motion to be completed is the same in both cases. In other words, a higher θ_{tol} causes R_i to reach the neighborhood of T earlier.

Note that the above explained generation of reference trajectories might be problematic for real-time applications due to high-frequency switching of the references; e.g. the low-level controller might fail to follow such references, power consumption would be higher, etc. Inspired by this fact, a continuous linear velocity reference is proposed for faster achievement of the goal in the next approach.

5.1.2 Continuous Linear Velocity Reference

The linear velocity reference is initialized with a low value and is increased as the error in orientation decreases. The aim is to have a smoother response and reach

the neighborhood of T earlier. A low limiting value of the orientation error, θ_{tol}, is specified to saturate the linear velocity reference since they are inversely proportional.

In this approach, the correct orientation is achieved in the same manner as the discontinuous case, i.e. the model for generation of angular velocity is (5.2).

On the other hand, the linear velocity reference, u_{1r}, is given by:

$$u_{1r} = \begin{cases} (1/|e_\theta|)\, k_2\, (d_{i2T} - d_{targ})\,, & \text{if} \quad |e_\theta| \geq \theta_{tol} \\ (1/\theta_{tol})\, k_2\, (d_{i2T} - d_{targ})\,, & \text{if} \quad |e_\theta| < \theta_{tol} \end{cases}\,, \tag{5.4}$$

where $k_2 > 0$ is the proportionality constant, $\theta_{tol} > 0$ is the saturating value for the orientation error, d_{i2T} is the signed distance between R_i and T, and d_{targ} is the distance to be maintained from T.

Appropriate selection of the constants k_1 and k_2 results in achievement of the specified task as seen in Fig. 5.6. The final configurations is not the same in these two cases as expected.

The generated reference velocities suffice to achieve the desired configuration in the simulations. Moreover, the velocity references are changing smoothly in time; hence, a desirable property in any control system is satisfied. Note that u_{1r} decreases although orientation error is below the specified value of 0.01. The reason is the fact that u_{1r} is dependent on d_{i2T}.

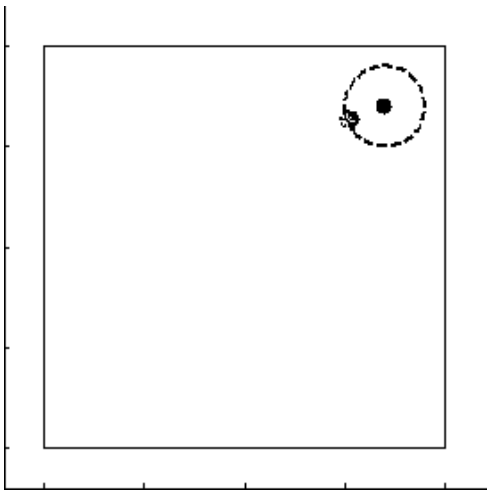


Figure 5.6: Continuous linear velocity final pose, $\theta_{tol} = 0.01$

The smooth reference velocities generated using (5.4) is plotted in Fig. 5.7.

In this chapter, coordinated motion of a group of autonomous nonholonomic mobile robots is modeled based on the proper selection of the linear and angular velocities of the virtual reference robot for each R_i. In this case, the nonholonomic constraint is automatically satisfied. Hence, the regulation of the errors between the virtual reference robot's pose and the pose of the actual robot are guaranteed. Once the velocities to yield coordinated motion are defined, they are integrated to generate the reference pose for the robot.

5.2 Desired Velocities

Coordinated motion of a group of autonomous nonholonomic mobile robots to accomplish the coordinated task described in this thesis is achieved by the definition of desired velocities due to closest neighbors of R_i and due to T as explained below.

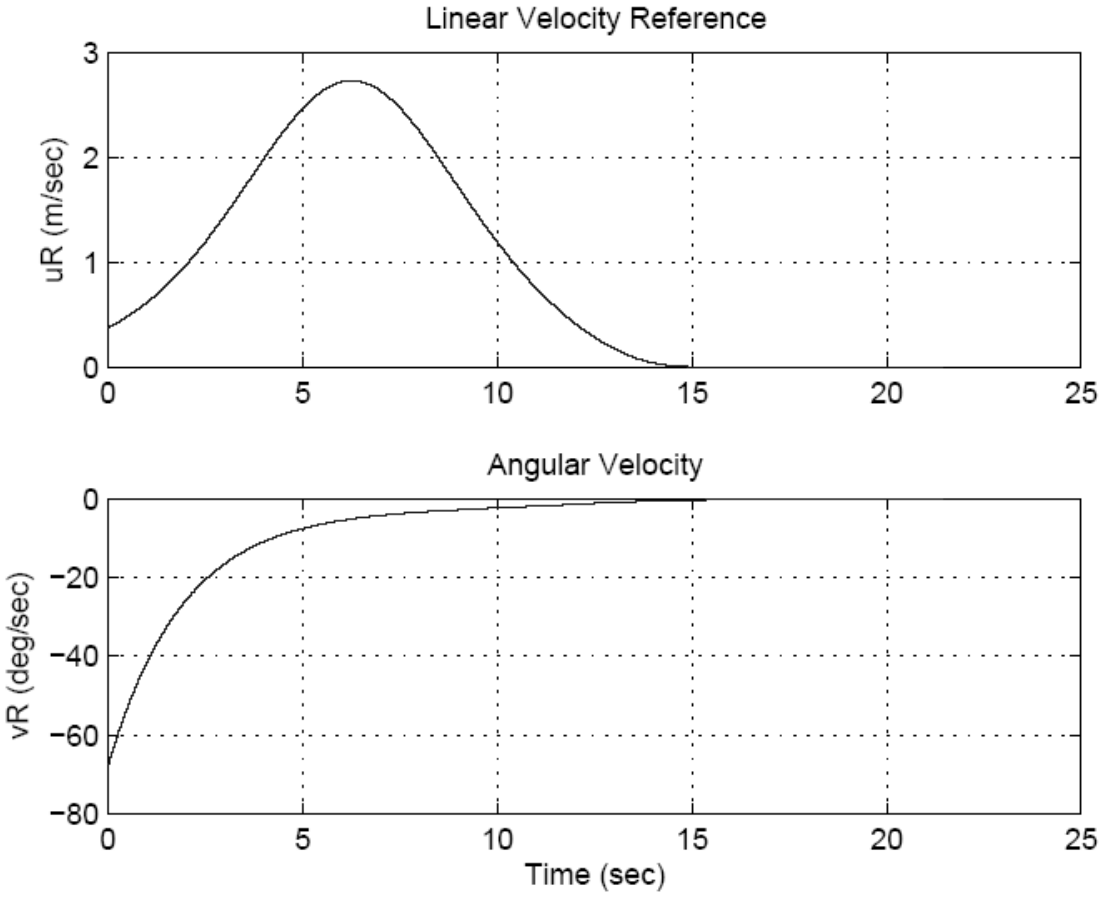


Figure 5.7: Continuous reference velocities with tolerance, $\theta_{tol} = 0.01$

5.2.1 Velocity due to Neighbors

A biologically inspired coordination method where R_i is in interaction only with its closest two neighbors is used as in the *Dynamic Coordination Model*. In this model, the second closest neighbor has no effect on the desired velocity of R_i, as long as the distance between R_i and T is below a predefined value d_{relax}; i.e. R_i is around T.

Coordinated motion of a group of autonomous nonholonomic mobile robots is defined in terms of maintaining certain mutual distances between the robots. In other words, each R_i should maintain a certain distance d_{coord} from its closest neighbors to achieve coordinated motion as a group. The desired velocity vector of R_i due to its closest neighbors, $\mathbf{v}_{coord}$, that forces the maintenance of distance d_{coord} between R_i and its neighbors is given as follows:

$$\begin{aligned} \mathbf{v}_{coord} &= \mathbf{v}_{cl1} + \mathbf{v}_{cl2} \ , \\ \mathbf{v}_{cl1} &= k_{lin}(d_{i2cl1} - d_{coord})\mathbf{n}_{i2cl1} \ , \\ \mathbf{v}_{cl2} &= \begin{cases} k_{lin}(d_{i2cl2} - d_{coord})\mathbf{n}_{i2cl2}, & \text{if} \quad d_{i2T} \geq d_{relax} \\ 0, & \text{if} \quad d_{i2T} < d_{relax} \end{cases} \ , \end{aligned} \tag{5.5}$$

where $k_{lin} > 0$ is the constant of proportionality, d_{coord} is the coordination distance R_i should maintain between itself and its closest neighbors, $\mathbf{v}_{cl1}$ and $\mathbf{v}_{cl2}$ are the velocities due to the closest and second closest neighbors respectively, d_{i2cl1} and d_{i2cl2} are the distances between R_i and its closest and second closest neighbors, d_{i2T} is the distance between R_i and T, $\mathbf{n}_{i2cl1}$ and $\mathbf{n}_{i2cl2}$ are the unit vectors from R_i to its closest neighbors, and d_{relax} is the critical distance of R_i to T below which the second closest neighbor loses effect.

5.2.2 Velocity due to Target

Each robot, R_i, in a group of autonomous nonholonomic mobile robots aimed to accomplish the specified coordinated task should move towards T from any initial pose as explained in the requirements of the task.

For each robot, R_i, a desired velocity vector due to T, $\mathbf{v}_{targ}$ is introduced, so that it maintains the specified distance d_{targ} between itself and T:

$$\mathbf{v}_{targ} = k_{lin}(d_{i2T} - d_{targ})\mathbf{n}_{i2T} \ , \tag{5.6}$$

where $k_{lin} > 0$ is a proportionality constant used in (5.1), d_{i2T} is the distance between R_i and T, and $\mathbf{n}_{i2T}$ is the unit vector in the direction of the vector from R_i to T.

5.2.3 Linear Combination for Reference Velocity

The reference velocity vector for R_i, namely $\mathbf{v}_{ref}$, is obtained as a linear combination of the desired velocities $\mathbf{v}_{coord}$ from (5.5) and $\mathbf{v}_{targ}$ from (5.6) multiplied by appropriate coefficients as follows:

$$\mathbf{v}_{ref} = k_{coord}\mathbf{v}_{coord} + k_{targ}\mathbf{v}_{targ} \ , \tag{5.7}$$

where k_{coord} is the weight of the velocity due to the closest neighbors of R_i, and k_{targ} is the weight of the velocity due to T.

The coefficients, $\mathbf{k}_{coord}$ and $\mathbf{k}_{targ}$, define the dependence of the generated reference velocity, $\mathbf{v}_{ref}$, on the neighbors and T throughout the coordinated motion. The reference velocity so obtained is then integrated to generate an appropriate reference trajectory for R_i.

5.3 Parameter Switching

Coordinated task manipulation by a group of mobile robots is defined as the accomplishment of a specified task together in certain formations. The necessary formation may vary based on the specifications of the coordinated task.

Successful manipulation of coordinated tasks by certain formations might require the parameters of the above equations to be changed dependent on the existence of specific conditions as described below.

In this chapter, the specified scenario of this work is split into two main phases:

- (A) Approaching T starting from an initial setting.
- (B) Achieving a circular formation with radius d_{targ} with T at the center to which all robots head towards.

In Phase (A), i.e. when R_i is approaching T from its initial position, the priority is given to coordination. In other words, $\mathbf{v}_{coord}$ is dominant in this case so that the robots move as a group. However, $\mathbf{v}_{targ}$ still contributes to the desired velocity so the group approaches T. To achieve the dominance of $\mathbf{v}_{coord}$ on $\mathbf{v}_{targ}$, the coefficients of the linear combination given by (5.7) are chosen to satisfy: $k_{coord} > k_{targ}$. In this phase, d_{coord} in (5.5) is set to the initially defined value d_{far} while k_{coord} is set to the predefined value k_{far}. R_i remains in this phase as long as $d_{i2T} \geq d_{relax}$; i.e. R_i is far from T.

If d_{i2T} is below d_{relax}, R_i is in Phase (B). In this phase, the priority is given to maintaining the distance d_{targ} from T. Hence, $\mathbf{v}_{targ}$ is dominant on $\mathbf{v}_{coord}$. To achieve this, $k_{targ} > k_{coord}$ in (5.7) should be satisfied. In this phase, k_{coord} is set to a new value, k_{near}. To achieve a uniform distribution on the formation circle, d_{coord} in (5.5) should also be changed to a new value, d_{near}, possibly different from the initial coordination distance, namely d_{far}. The required value of d_{near} is given by (5.8) as explained in Section 4.2.

$$d_{near} = d_{targ}\sqrt{2(1-\cos(2\pi/n))}\ . \tag{5.8}$$

The last parameter that affects the generation of the reference velocity for R_i is k_{targ} in (5.7), which defines the dependency of $\mathbf{v}_{ref}$ on $\mathbf{v}_{targ}$. Since k_{coord} is switched from k_{far} to k_{near} when R_i enters Phase (B) from Phase (A), a constant k_{targ} might be used as long as the condition $k_{far} > k_{targ} > k_{near}$ is satisfied. However, for the dominance in each phase to be more significant, k_{targ} is also switched to a new value in this model.

The coefficient of the velocity due to the neighbor interactions in (5.7), k_{coord}, is switched as a continuous function of the distance of R_i to T, namely d_{i2T}. This parameter switching is modeled by the following sigmoid function:

$$k_{coord} = k_{near} + \frac{k_{far} - k_{near}}{1 + exp(\mu(d_{relax} - d_{i2T} + \phi))}\ , \tag{5.9}$$

where $\mu > 0$ and $\phi > 0$ are constants defining the characteristics of the curve, k_{far} and k_{near} are the coefficients of $\mathbf{v}_{coord}$ in (5.7) in Phase (A) and Phase (B),

respectively. Fig. 5.8 depicts the obtained switching behavior of k_{coord}.

Instead of computing k_{targ} with a similar sigmoid function, k_{targ} is defined as a function of k_{coord} as follows:

$$k_{targ} = 1 - k_{coord} \ . \tag{5.10}$$

It is important to note that, the choice of k_{near} and k_{far} in (5.9) should satisfy: $0 \leq k_{coord} \leq 1$ for (5.10) to yield proper values. Fig. 5.9 depicts the obtained continuous switching of k_{targ} using (5.10).

Similarly, d_{coord} is switched as a continuous function of d_{i2T}, again modeled by a sigmoid function defined as:

$$d_{coord} = d_{near} + \frac{d_{far} - d_{near}}{1 + exp(\mu(d_{relax} - d_{i2T} + \phi))} \ . \tag{5.11}$$

where $\mu > 0$ and $\phi > 0$ are constants defining the characteristics of the curve, d_{far} and d_{near} are the distances to be maintained between the robots for $d_{i2T} \geq d_{relax}$ and $d_{i2T} < d_{relax}$ in (5.5), respectively. Fig. 5.10 depicts so obtained switching behavior of d_{coord}.

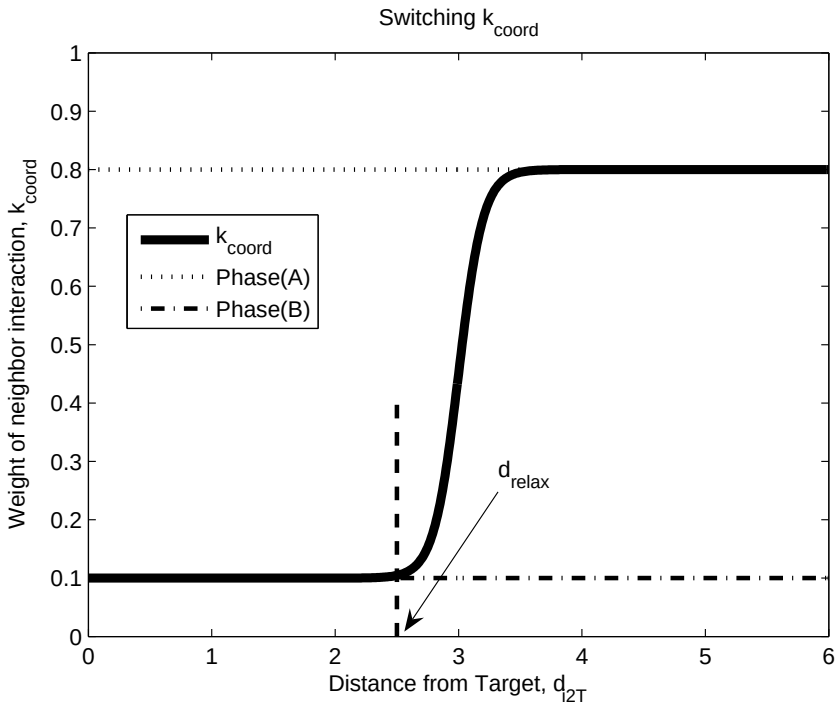


Figure 5.8: Adaptive neighbor interaction coefficient, k_{coord} vs distance from T, d_{i2T}

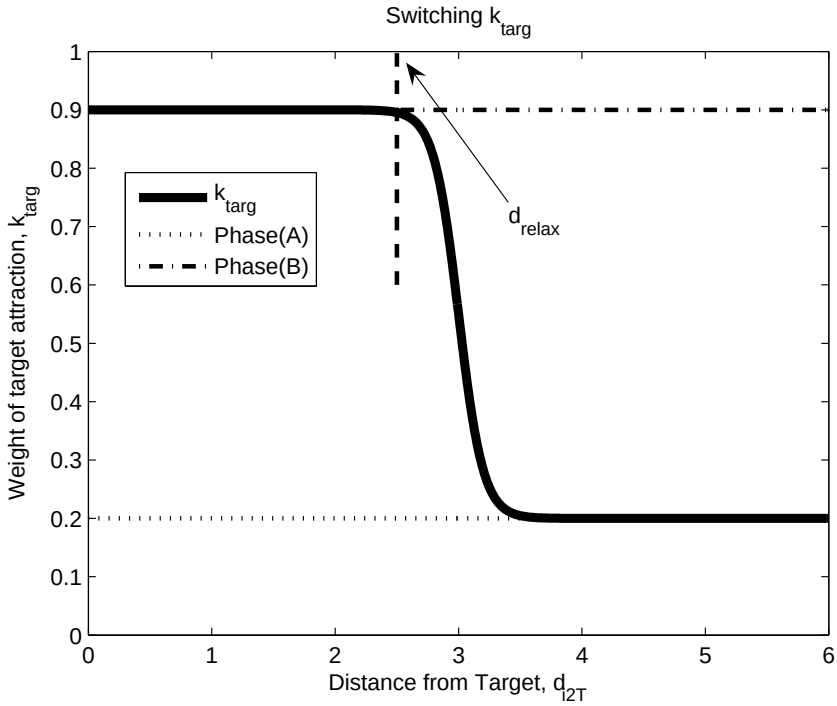


Figure 5.9: Adaptive target attraction coefficient, k_{targ} vs distance from T, d_{i2T}

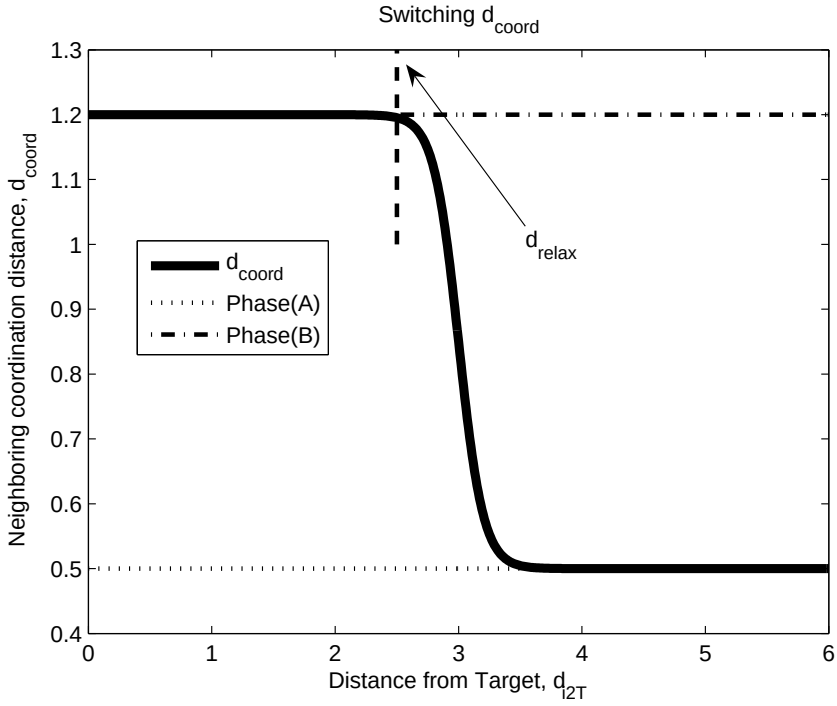


Figure 5.10: Adaptive coordination distance, d_{coord} vs distance from T, d_{i2T}

5.4 Velocity Update to Avoid Collisions

As discussed in Chapter 1, collision avoidance is one of the essential problems in modeling coordinated motion of a group of autonomous mobile robots. The algorithm to avoid the collisions in this method is the same with the algorithm given in Section 4.3. In this work, neither static obstacles are considered nor T is considered as an obstacle since $\mathbf{v}_{targ}$ is dominant in Phase (B), and it keeps the robot at distance d_{targ} from T.

When R_i detects a collision risk, the reference velocity of R_i is updated conveniently as described in the previous chapter and $\mathbf{v}_{ref}$ is disregarded for some specific time delay. During this delay, the priority is to avoid the predicted collision so R_i moves in the direction that will yield a collision-free trajectory. After the avoidance, R_i is oriented back towards T, under the effect of $\mathbf{v}_{ref}$.

5.5 Reference Trajectory Generation

The reference linear and angular velocities, u_{1r} and u_{2r}, of the robots are derived from the calculated reference velocity, $\mathbf{v}_{ref} = \begin{bmatrix} v_{xref} & v_{yref} \end{bmatrix}^t$, given by (5.7).

The first approach to generate of discontinuous reference linear velocities might be used for the generation of velocities. However, the discontinuous reference signals might be problematic in real-time applications as already discussed in Section 5.1.1.

In the developed method of this chapter, the second approach described in Section 5.1.2 is used. Hence, smooth and continuous references, that can be tracked by the control laws presented in Chapter 3, are generated for the autonomous nonholonomic mobile robots.

The reference velocity for a unicycle type mobile robot can be expressed in terms of a reference speed and a reference orientation due to the nonholonomic constraint. Such an expression of the reference facilitates the generation of linear and angular velocities.

The reference speed, $|\mathbf{v}_{ref}|$, and reference orientation, θ_{ref}, are obtained from the calculated reference velocity as follows:

$$
\begin{aligned}
|\mathbf{v}_{ref}| &= \sqrt{v_{xcoord}^2 + v_{ycoord}^2} \ , \\
\theta_{ref} &= \arctan(v_{yref}/v_{xref}) \ .
\end{aligned}
\tag{5.12}
$$

The angular velocity reference for R_i, u_{2ref}, is designed in terms of the orientation error, e_θ, between the reference orientation, θ_{ref}, and the actual orientation, θ_i, by a proportional gain of $k_{rot} > 0$ as follows:

$$
\begin{aligned}
e_\theta &= \theta_{ref} - \theta_i \ , \\
u_{2ref} &= k_{rot} e_\theta \ .
\end{aligned}
\tag{5.13}
$$

On the other hand, the linear velocity reference for R_i, u_{1ref}, is designed as a piecewise linear function to enable saturation of the linear velocity as in:

$$
u_{1ref} = \begin{cases} (1/|e_\theta|)\,|\mathbf{v}_{ref}|\,, & \text{if} \quad |e_\theta| \geq \theta_{lim} \\ (1/\theta_{lim})\,|\mathbf{v}_{ref}|\,, & \text{if} \quad |e_\theta| < \theta_{lim} \end{cases} \ , \tag{5.14}
$$

where $\theta_{lim} > 0$ is a limiting value to saturate the linear velocity, e_θ is the orientation error, and $\mathbf{v}_{ref}$ is the reference speed.

The reference pose of R_i, $\mathbf{X}_{ref} = \begin{bmatrix} x_{ref} & y_{ref} & \theta_{ref} \end{bmatrix}^t$, is obtained by the integration of the calculated reference linear and angular velocities:

$$
\begin{bmatrix} x_{ref} \\ y_{ref} \\ \theta_{ref} \end{bmatrix} = \begin{bmatrix} \int u_{1ref} cos\theta_{ref} dt \\ \int u_{1ref} sin\theta_{ref} dt \\ \int u_{2ref} dt \end{bmatrix} \ . \tag{5.15}
$$

5.6 Switching Between Controllers

Finally, the generated reference pose is input as reference to the controller of the robot that applies the feedback control laws given in Chapter 3.

When the robots are uniformly distributed around T on a circle of radius d_{targ} based on the framework given throughout the chapter, the last maneuver that robots should do is to orient themselves towards the target.

The reference trajectories generated by the integration of reference velocities are tracked by the control law given by (3.9). After the robots are fully distributed on the circle of radius d_{targ} around T, the control law of (3.12) is used to park the robots on the circle with necessary orientations so that they head towards T. The fixed-point reference position for R_i to park at is used as the current position of R_i on the circle, whereas the reference orientation for R_i to be headed towards T is calculated artificially for that fixed reference position.

Chapter 6

Simulations and Experiments

Computer simulations were carried out to test the performance of the developed models; both for *Dynamic Coordination Model* of Chapter 4 and for *Kinematic Coordination Model* of Chapter 5.

The system was simulated in *Simulink* 6.1 embedded in *MATLAB* 7.0.1, a very convenient tool for modeling and simulation of systems, both continuous and discrete, linear and nonlinear. Plotting and animation properties were very useful in adjustment of parameters, development of models, etc.

The novel collision avoidance algorithm explained in detail in Section 4.3 was also tested experimentally on Boe-Bot unicycle robots along with OpenCV, where a wide variety of image interpretation tools can be implemented, as the software programming basis.

6.1 Dynamic Coordination Model Simulations

The structure of the considered model for a group of three autonomous mobile robots generated for the simulations in *Simulink* is depicted in Fig. 6.1.

The virtual masses used in simulations are set as $m_i = 1kg$. The maximum speed of the masses, hence the robots, are set to $0.5m/sec$, while the maximum angular speed of the robots was set to $(2\pi)rad/sec$. The virtual collision prediction regions of the autonomous robots have radii $r_{coll} = 0.45m$ and angle $\theta_{coll} = (pi/2)rad$. The constants in the control laws were set as: $k_1 = k_2 = 20.0$ for time-variant reference trajectories in (3.9) to be used in Phase (1) and $k_{1p} = 23.0$, $k_{2p} = 16.0$ for fixed-point reference trajectories in (3.12) to be used in Phase (2).

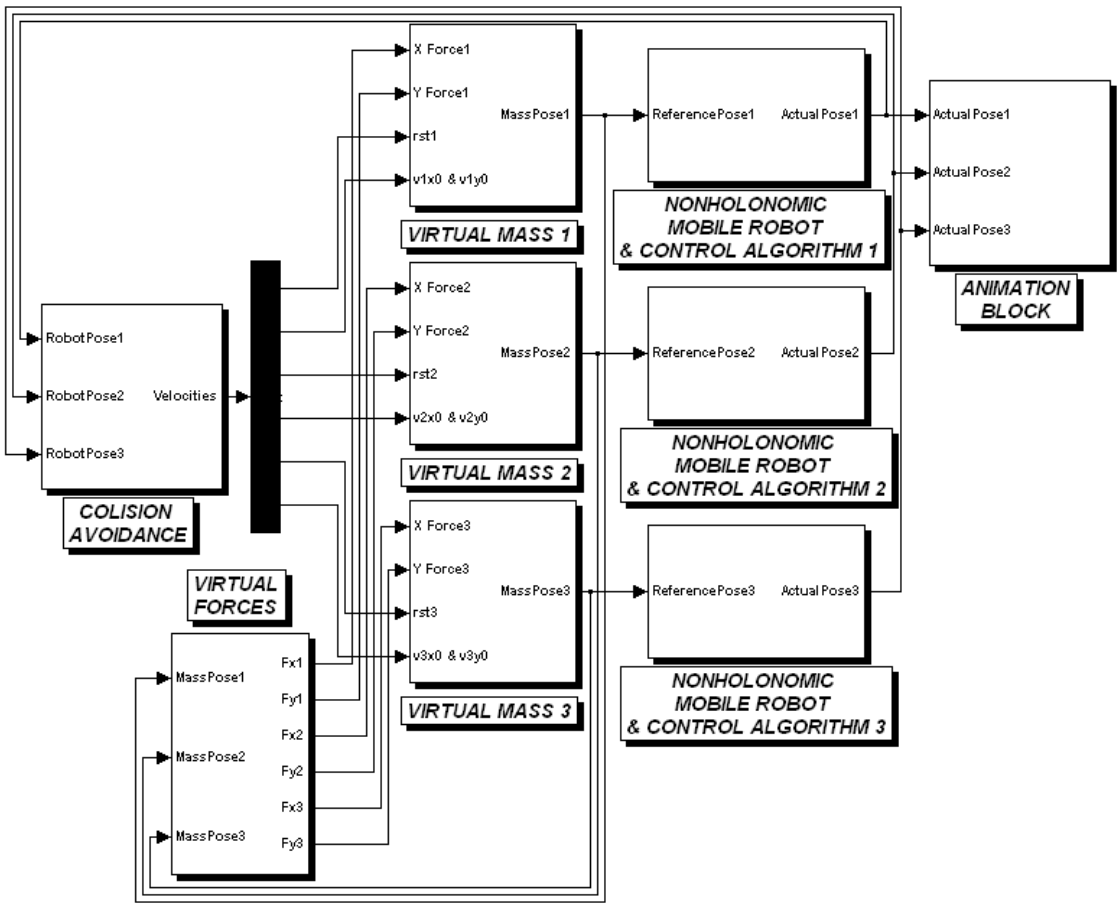


Figure 6.1: *Simulink* block diagram for the simulations of *Dynamic Coordination Model* for 3 robots

6.1.1 Collision Avoidance Simulations

The simulations were run with $\mathbf{F}_{targ} = \mathbf{F}_{coord} = 0$ to see the performance of the collision avoidance algorithm in the absence of any coordinated behavior.

Head-to-Head Collision Avoidance

In this initial setting, the robots are moving towards each other with constant velocities as seen in Fig. 6.2(a). Fig. 6.2(b) shows the moment of collision prediction. Their new situation is given in Fig. 6.2(c) after they avoid the collision by taking the necessary action. Since both robots touch the other's *VCPR*, they both change their orientations applying the velocity update algorithm explained in Section 4.3.2. The result proves the success of the algorithm for head-to-head collisions.

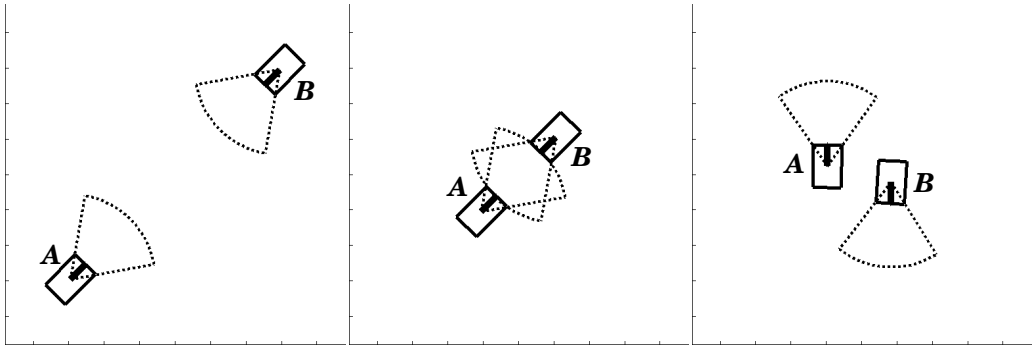


Figure 6.2: Dynamic coordination model, Head-to-Head Collision Avoidance: (a)Before (b)Prediction (c)After

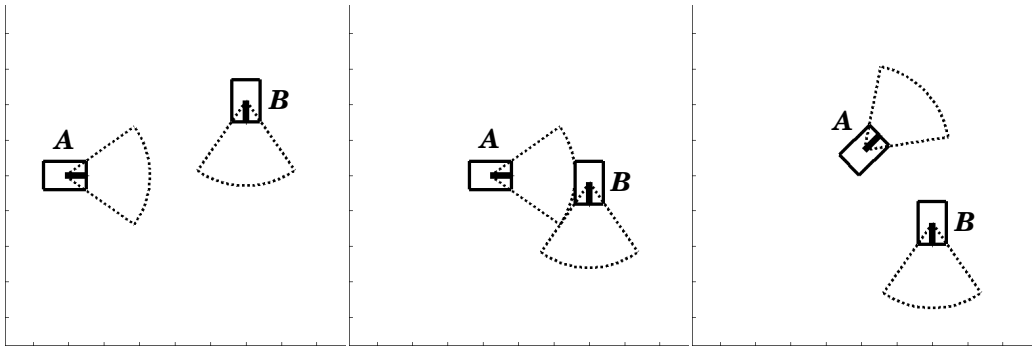


Figure 6.3: Dynamic coordination model, Single-Robot Collision Avoidance: (a)Before (b)Prediction (c)After

Single-Robot Collision Avoidance

Fig. 6.3(a) shows the robots before any collision is predicted for this scenario where one of the robots is approaching the other from its right hand side. Only one of the robots, A, senses that the other is too close, i.e. detects the risk of a collision, as shown in Fig. 6.3(b). The new headings of the robots, after A changes its orientation, are depicted in Fig. 6.3(c). The result proves the success of the collision avoidance algorithm in this situation as well.

6.1.2 Coordinated Motion Simulations

The coordinated motion method was simulated on the basis of the specified coordinated task. The presented method was simulated for two distinct initial configura-

tions for three robots and two different initial configurations for four robots. The tolerance for the robots to start parking was set to be 0.2 percent of d_{near}. The other parameters in the simulations were set as in Table 6.1.

Parameter	Value	Reference Equation
c_{coord}	$10.0Ns/m$	4.3
d_{far}	$2.0m$	4.3
k_{targ}	$15.0N/m$	4.4
c_{targ}	$15.0Ns/m$	4.4
d_{targ}	$2.0m$ for 3 Robots $3.0m$ for 4 Robots	4.4
k_{far}	$50.0N/m$	4.8
k_{near}	$10.0N/m$	4.8
μ	10.0	4.8
ϕ	0.5	4.8
d_{break}	$1.3d_{targ}$	4.8

Table 6.1: Dynamic coordination model parameters for simulations

Scenario-1

This simulation was run for a group of three autonomous nonholonomic mobile robots. Since virtual bonds are constructed for each m_i with its closest two neighbors, mutual coordination forces among each binary combination of the robots exist in this simulation.

The initial configuration is such that the robots and the target are placed on opposite corners of the room as depicted in Fig. 6.4(a). They detect and avoid the collisions and move in a coordinated fashion in the form of an equilateral triangle with sides d_{far}. As the distance between each R_i and T falls below d_{break}, it starts circular motion since the virtual bonds are relaxed and $\mathbf{F}_{targ}$ becomes dominant. Finally, they take the form of an equilateral triangle with sides equal to d_{near}. Snapshots from the animation are given in Fig. 6.4.

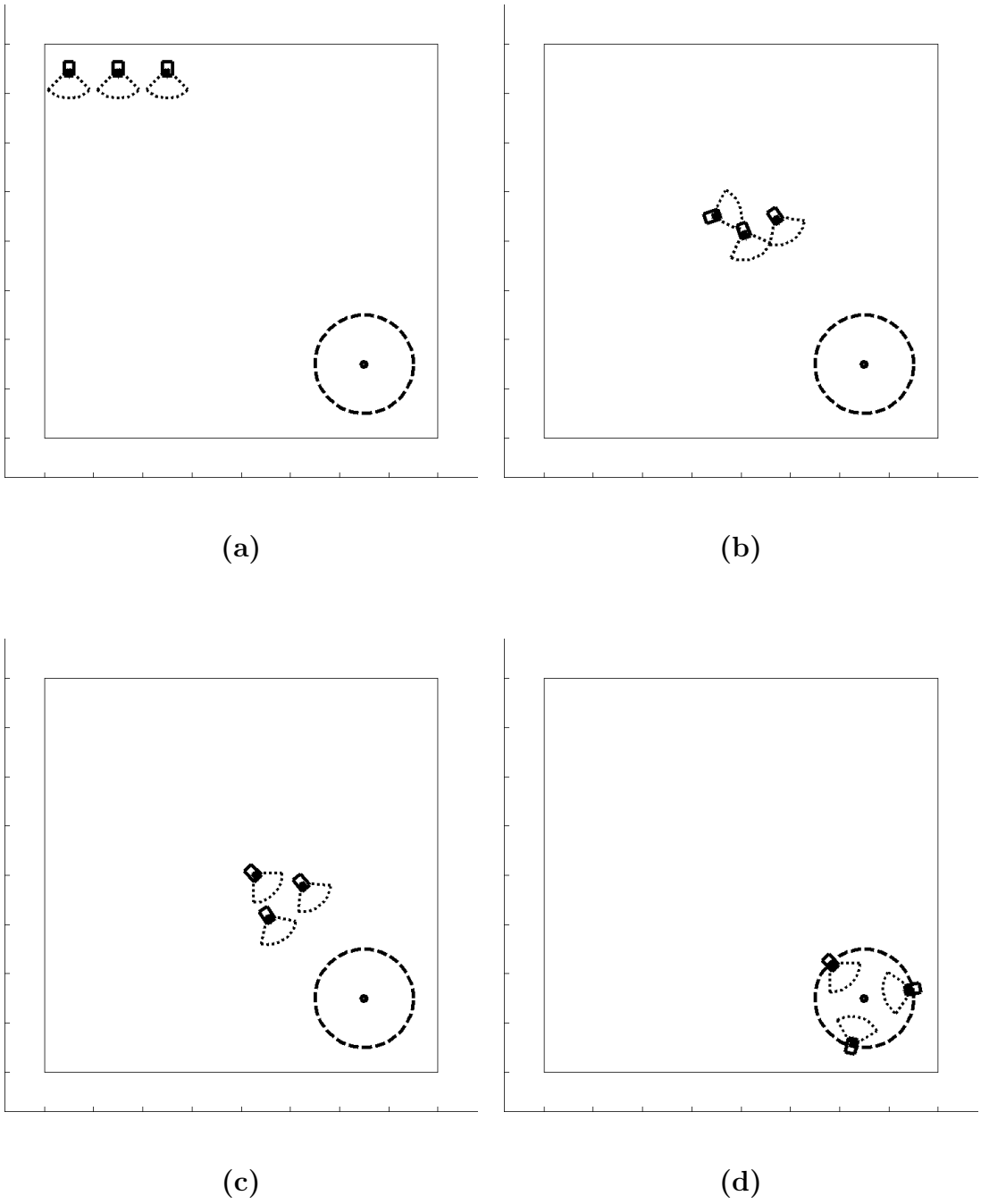


Figure 6.4: Dynamic coordination model, Scenario-1: (a)Initial configuration (b)Collision avoidance (c)Coordinated motion (d)Desired formation achieved

Scenario-2

This simulation was also run for a group of three autonomous nonholonomic mobile robots. Each m_i is in coordination with all the other robots in the group since there are three robots in this simulation.

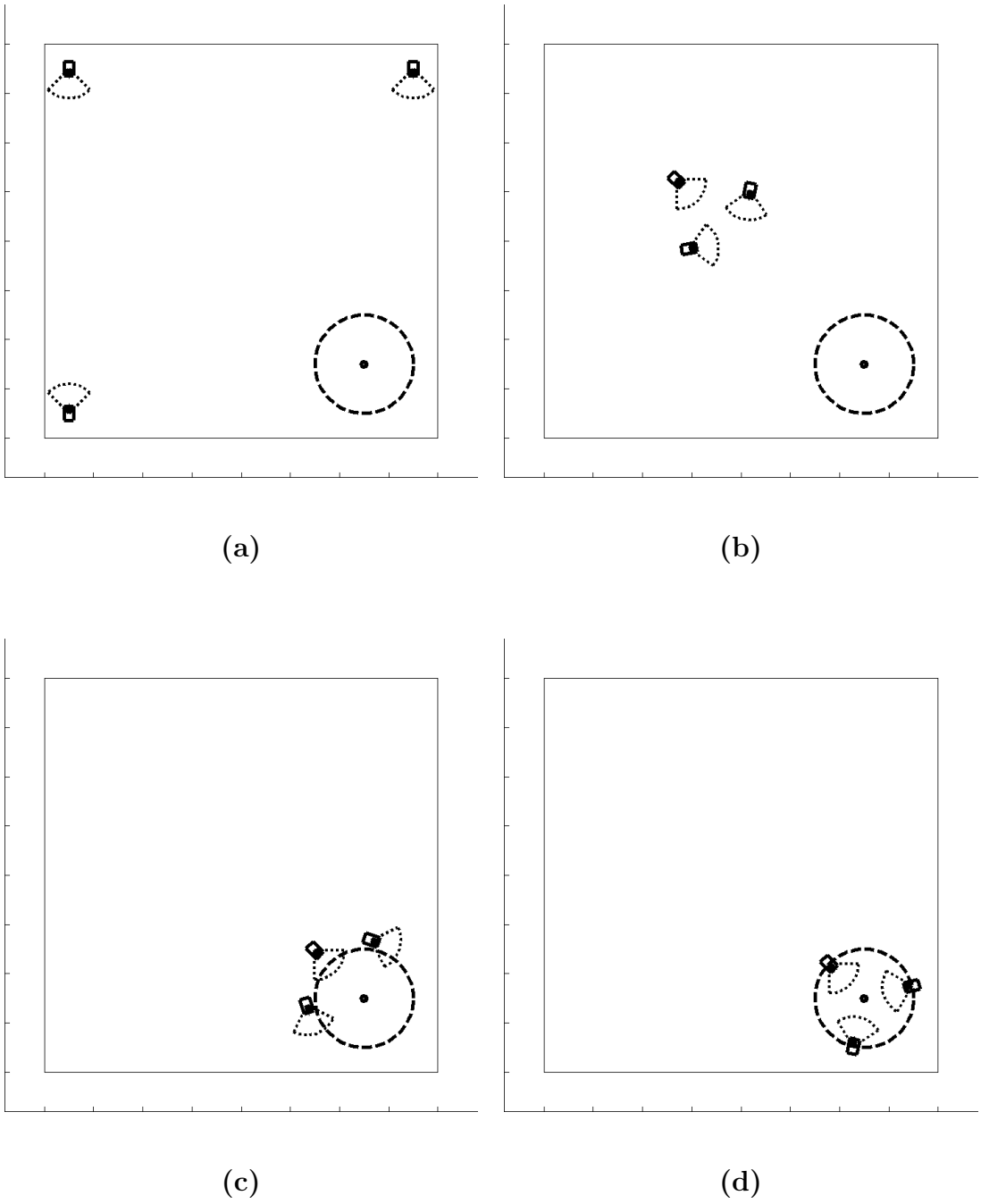


Figure 6.5: Dynamic coordination model, Scenario-2: (a)Initial configuration (b)Coordination dominant (c)Virtual bonds relaxed (d)Desired formation achieved

The initial setting of this scenario is depicted in Fig. 6.5(a). In this case, the robots first move towards each other and then approach T as a group. This is because $k_{coord} = k_{far} > k_{targ}$ initially. After the robots form the triangle seen in Fig. 6.5(b), they move in a coordinated manner and achieve the desired formation.

Scenario-3

This simulation was run for a group of four autonomous nonholonomic mobile robots. Since virtual bonds are constructed for each m_i with its closest two neighbors, the farthest robot's virtual mass with respect to m_i doesn't apply any force on m_i in

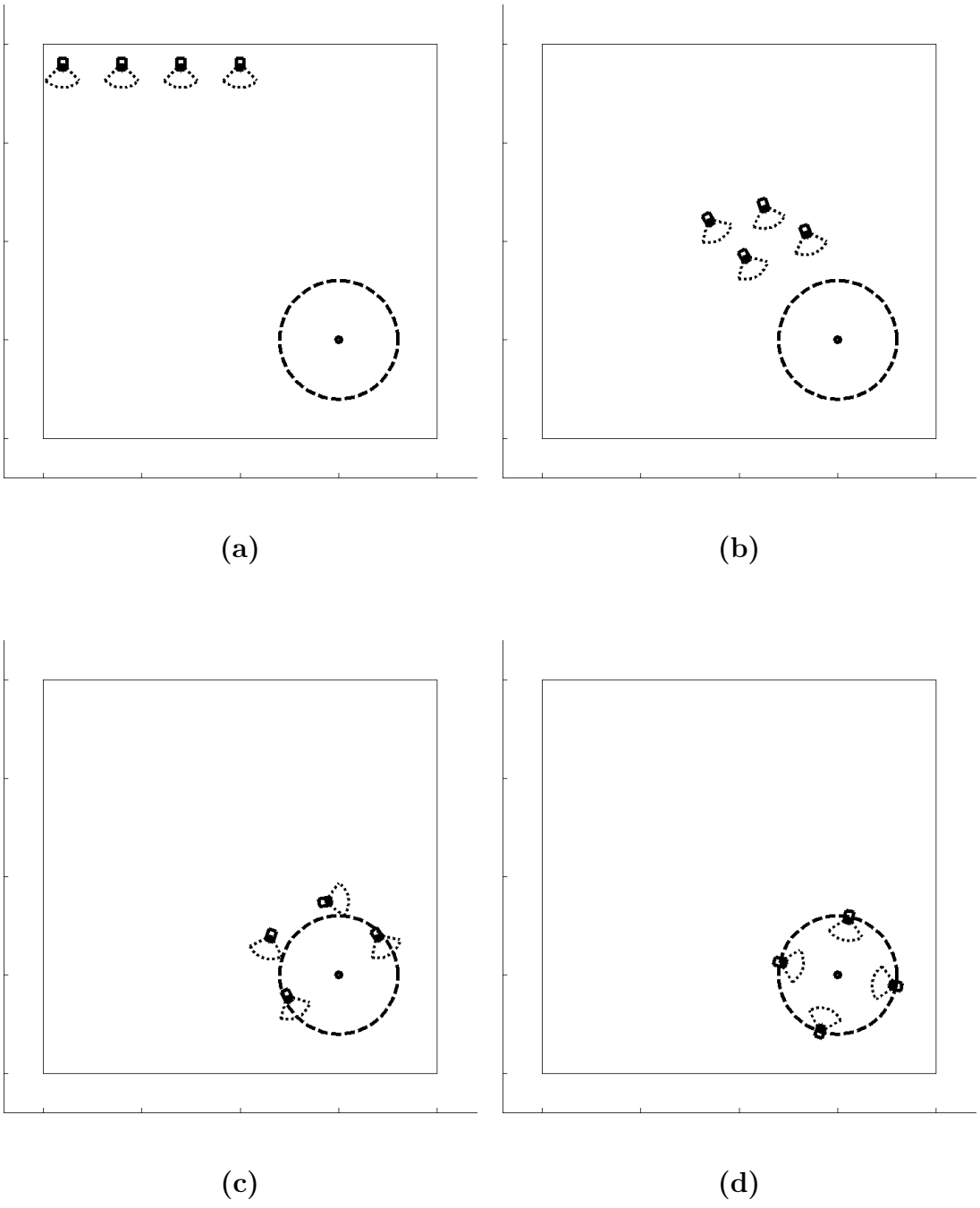


Figure 6.6: Dynamic coordination model, Scenario-3: (a)Initial configuration (b)Coordinated motion (c)Virtual bonds relaxed (d)Desired formation achieved

cases where number of robots is higher than three such as this simulation.

In this scenario, the robots and the target are placed on opposite corners of the room as in Fig. 6.6(a). Fig. 6.6(b) shows their motion in the form of a parallelogram with sides equal to d_{far}. Fig. 6.6(c) is a snapshot depicting the circular motion of the robots around T under the dominant effect of $\mathbf{F}_{targ}$. Finally, they take the form of a square with sides d_{near} as seen in Fig. 6.6(d).

Scenario-4

This simulation was also run for a group of four autonomous nonholonomic mobile robots. The farthest virtual mass doesn't have any effect on m_i in this simulation as well.

In this initial configuration, four robots are placed at the corners of the room while the target is in the middle as shown in Fig. 6.7(a). All robots directly head towards the target, and wait for the others since they need to achieve distances of d_{near} before they can park on the formation circle. The robots oscillate performing circular motion around the target as shown in Fig. 6.7(c). Fig. 6.7 shows some snapshots from this animation.

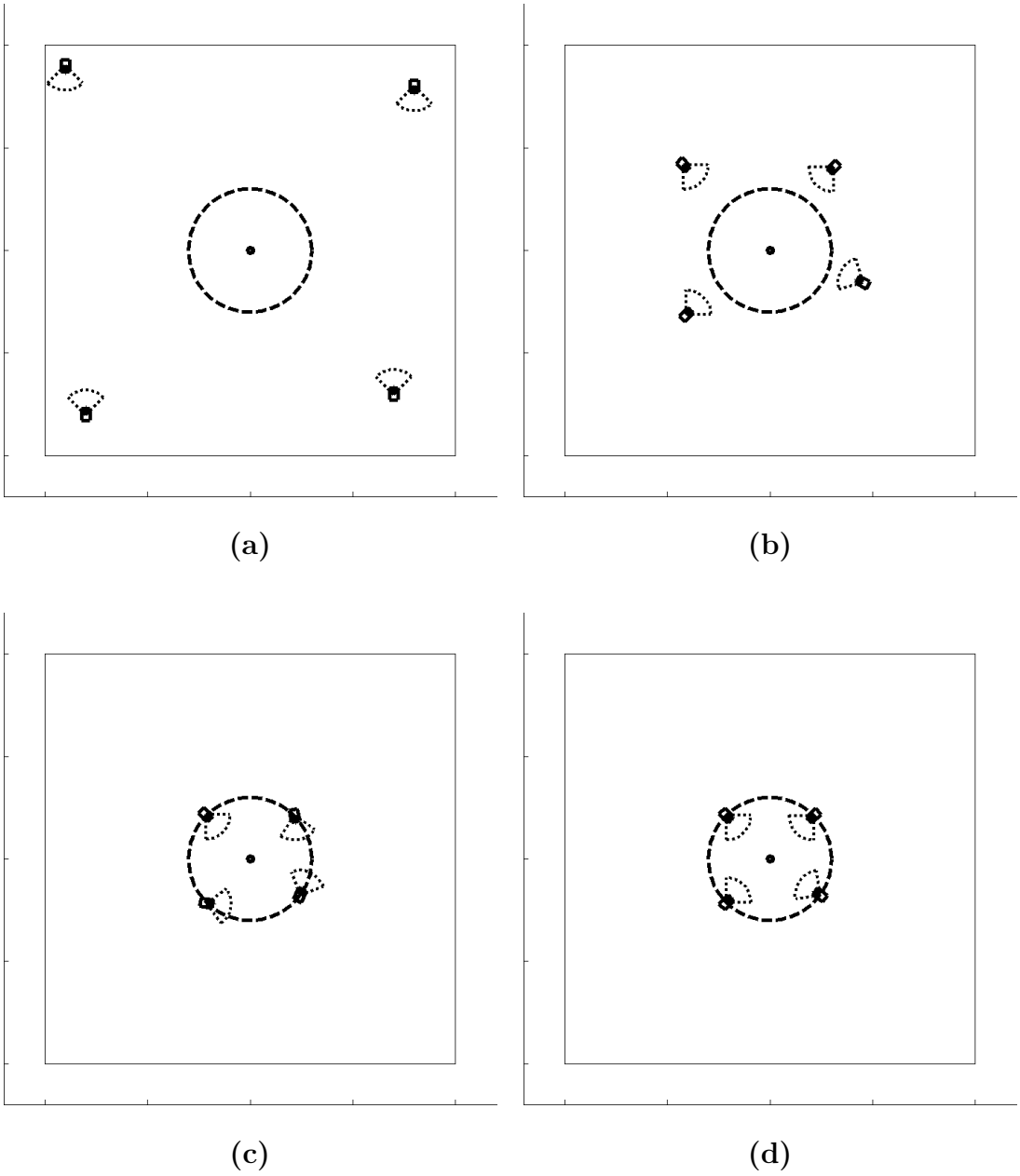


Figure 6.7: Dynamic coordination model, Scenario-4: (a)Initial configuration (b)Approaching T (c)Oscillations around T (d)Desired formation achieved

The results of the simulations for the proposed dynamic coordination model with three and four robots are all satisfactory in the sense that the robots move in a coordinated manner, detect and avoid any possible collisions and form a uniform polygon with sides equal to d_{near}, with each robot heading towards the target, where the center of the peripheral circle of that polygon is the target.

6.2 Kinematic Coordination Model Simulations

The structure of the considered model for a group of three autonomous mobile robots generated for the simulations in *Simulink* is depicted in Fig. 6.8.

In the simulations, the maximum linear speed of the robots is set to $0.5m/sec$, while the maximum angular speed is set to $(\pi/3)rad/sec$. The virtual collision prediction regions of the autonomous robots have radii $r_{coll} = 0.9m$ and angle $\theta_{coll} = (pi/2)rad$. The constants in the control laws were set as: $k_1 = k_2 = 20.0$ for time-variant reference trajectories in (3.9) to be used in Phase (A) and $k_{1p} = 23.0$, $k_{2p} = 16.0$ for fixed-point reference trajectories in (3.12) to be used in Phase (B).

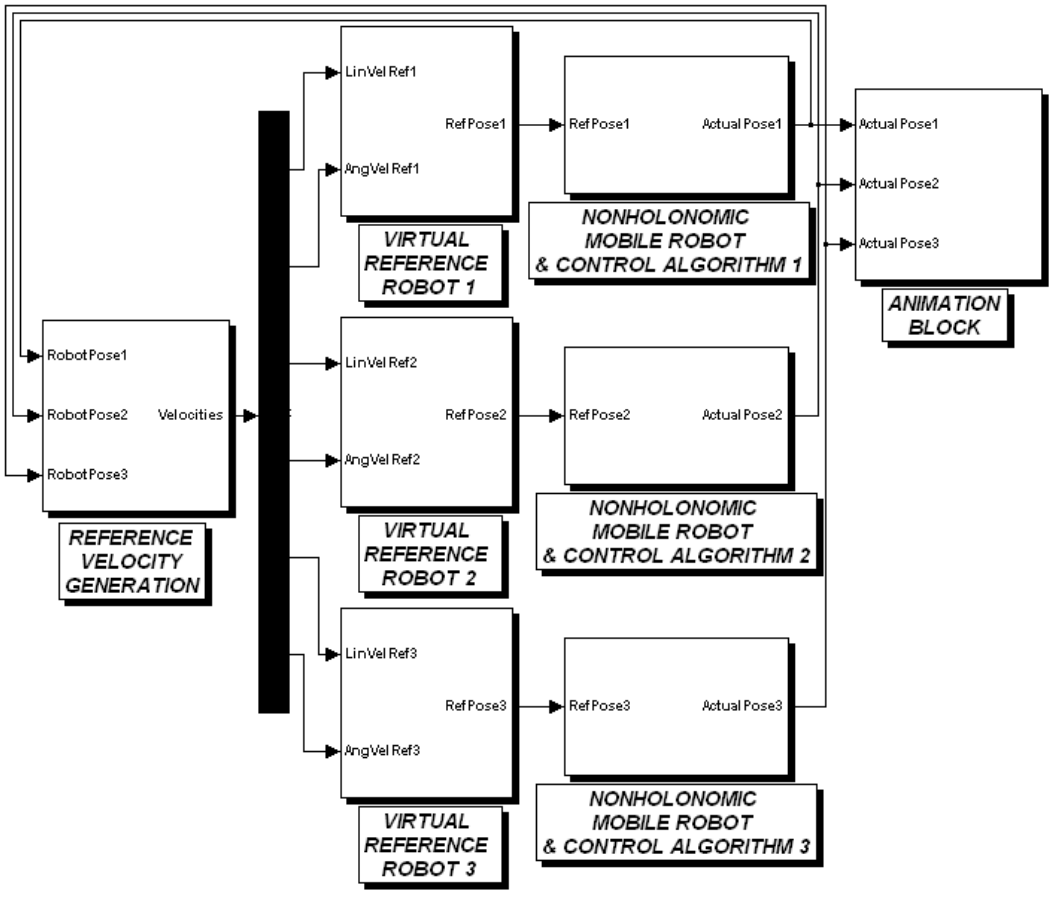


Figure 6.8: *Simulink* block diagram for the simulations of *Kinematic Coordination Model* for 3 robots

6.2.1 Collision Avoidance Simulations

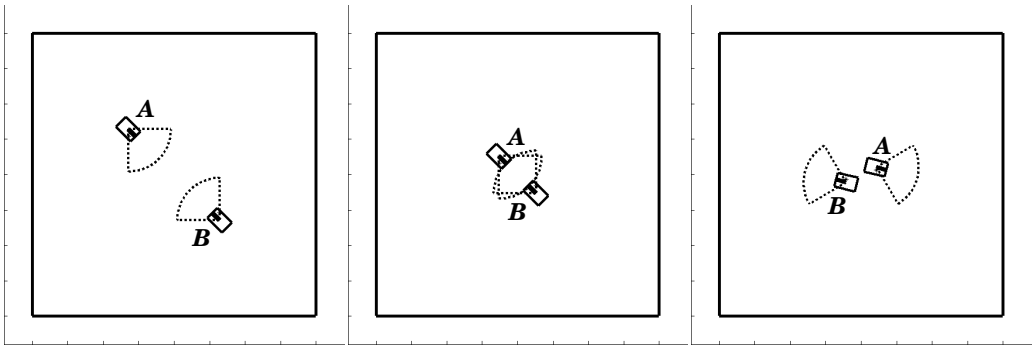
The simulations were run with $\mathbf{v}_{targ} = \mathbf{v}_{coord} = 0$ to see the performance of the collision avoidance algorithm in the absence of any coordinated behavior.

Head-to-Head Collision Avoidance

In this collision avoidance test, the robots move towards each other with constant velocities as seen in Fig. 6.9(a). The moment of collision prediction is depicted in Fig. 6.9(b). The new situation is given in Fig. 6.9(c) after they avoid the collision by taking the necessary actions. Since both robots touch the other's *VCPR*, they both change their orientations by the application of the velocity update algorithm described in Section 4.3.2. The result proves the success of the algorithm for predicting and avoiding head-to-head collisions.

Single-Robot Collision Avoidance

The initial states of the robots before any collision is predicted for this scenario where one of the robots is approaching the other from its side is depicted in Fig. 6.10(a). Only one of the robots, B, senses that the other robot is too close, i.e. there's a collision risk, as shown in Fig. 6.10(b). The new headings of the robots, after B changes its orientation, are given in Fig. 6.10(c). The result proves the success of the collision avoidance algorithm in the case of only one of the robots predicting the collision as well.



(a) **(b)** **(c)**

Figure 6.9: Kinematic coordination model, Head-to-Head Collision Avoidance: (a)Before (b)Prediction (c)After

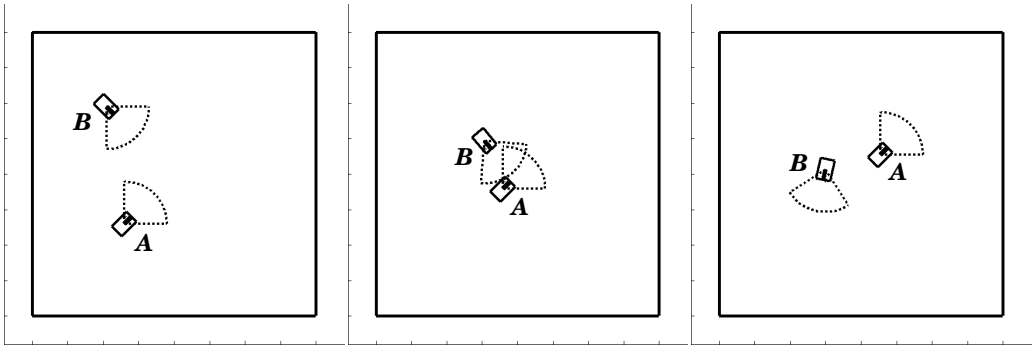


Figure 6.10: Kinematic coordination model, Single-Robot Collision Avoidance: (a)Before (b)Prediction (c)After

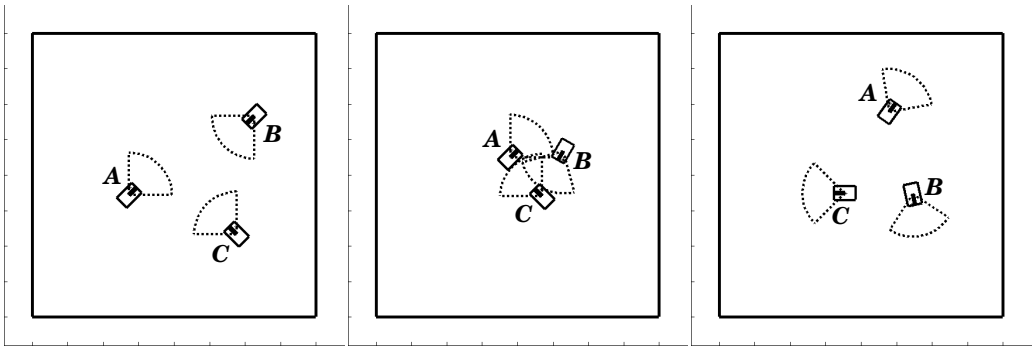


Figure 6.11: Kinematic coordination model, Three-Robots Simultaneous Collision Avoidance: (a)Before (b)Prediction (c)After

Three-Robots Collision Avoidance

The performance of the collision avoidance algorithm using kinematic was also tested for three autonomous nonholonomic mobile robots. In this scenario, three robots are headed towards each other as given in Fig. 6.11(a). At the moment shown in Fig. 6.11(b), all robots predict collisions simultaneously. Although small oscillations were observed for a short time, they were seen to avoid collisions and the resulting safe configuration is given in Fig. 6.11(c). This result reveals the satisfactory performance of the proposed algorithm for simultaneous collision risks among three robots. This is expected due to the autonomous and modular nature of the generated system.

6.2.2 Coordinated Motion Simulations

The proposed coordinated motion model was simulated on the basis of the specified coordinated task. The presented method was simulated for two distinct initial configurations for three robots, three different initial configurations for four robots and a single initial configuration for five robots. The tolerance for the robots to start parking was set to be 0.17 percent of d_{near}. The other parameters in the simulations were set as in Table 6.2.

Parameter	Value	Reference Equation
k_{lin}	0.5	5.5 and 5.6
d_{relax}	$1.3d_{targ}$	5.5, 5.9 and 5.11
d_{targ}	$2.0m$	5.6
k_{far}	0.8	5.9
k_{near}	0.1	5.9
μ	10.0	5.9 and 5.11
ϕ	0.5	5.9 and 5.11
d_{far}	$2.0m$	5.11
k_{rot}	1.0	5.13
θ_{lim}	1°	5.14

Table 6.2: Kinematic coordination model parameters for simulations

Scenario-1

This simulation was run for a group of three autonomous nonholonomic mobile robots. Each R_i is in coordination with all the other robots in the group as long as $d_{i2T} \geq d_{relax}$ as stated in (5.5).

In this scenario, the robots are placed on one corner of the room while T is at the opposite corner initially. It was observed that they approach each other and move towards T in a coordinated manner. Once they are close enough to T, to achieve mutual distances of d_{near}, they spread around on the circle since they should stay on the circle due to the dominant effect of $\mathbf{v}_{targ}$. Finally, the group achieves the desired formation. Snapshots from this animation are given in Fig. 6.12.

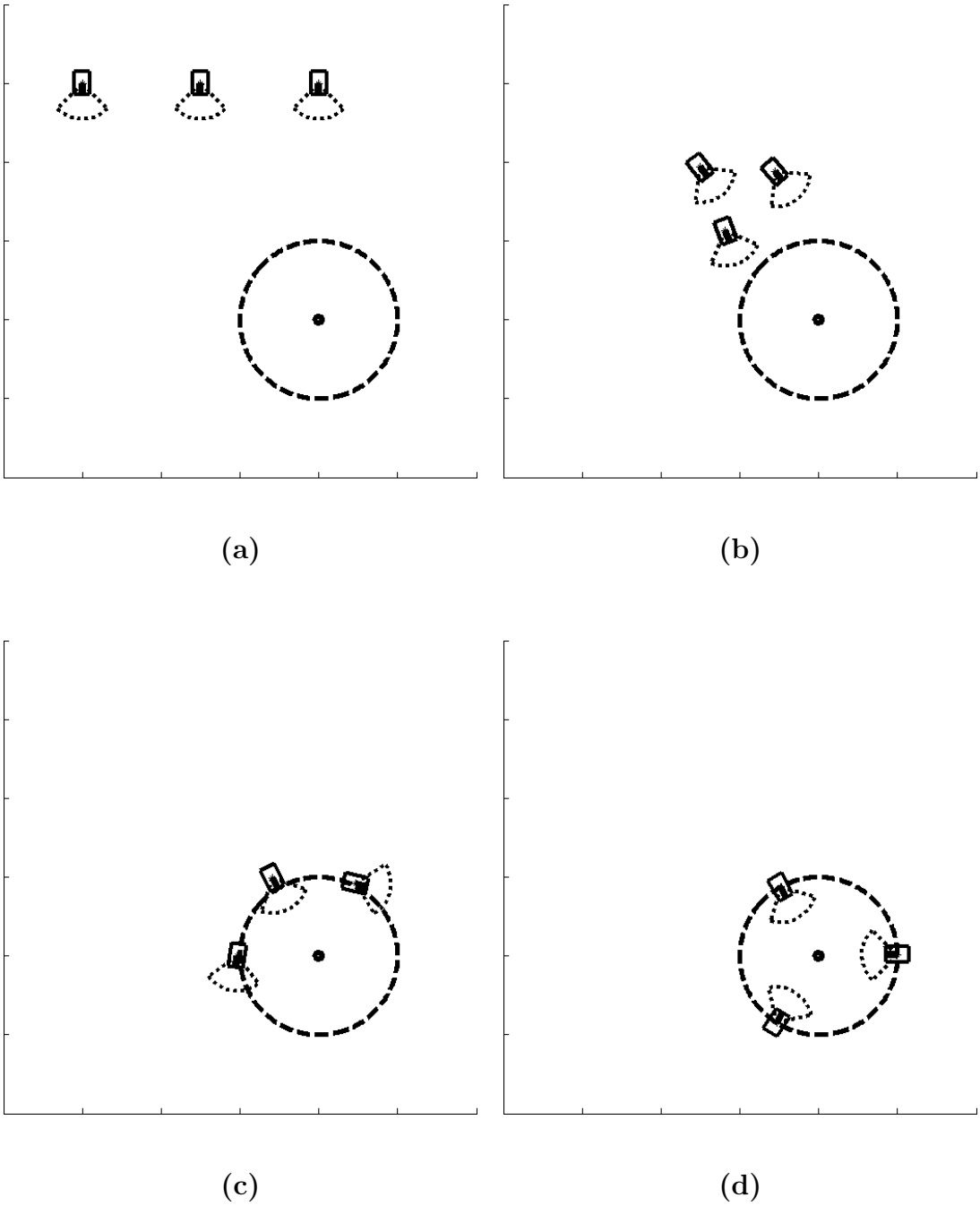


Figure 6.12: Kinematic coordination model, Scenario-1: (a)Initial configuration (b)Coordinated motion (c)Distribution on circle (d)Desired formation achieved

Scenario-2

This simulation was also run for a group of three autonomous nonholonomic mobile robots. Each R_i is in coordination with all the other robots in the group unless $d_{i2T} < d_{relax}$.

The initial setting in this case is depicted in Fig. 6.13(a). The robots move towards T and towards each other but coordination effects are dominant as seen in Fig. 6.13(b). Once they reach the circle, they spread around the circle due to $\mathbf{v}_{coord}$ as shown in Fig. 6.13(c). Snapshots from this animation are shown in Fig. 6.13.

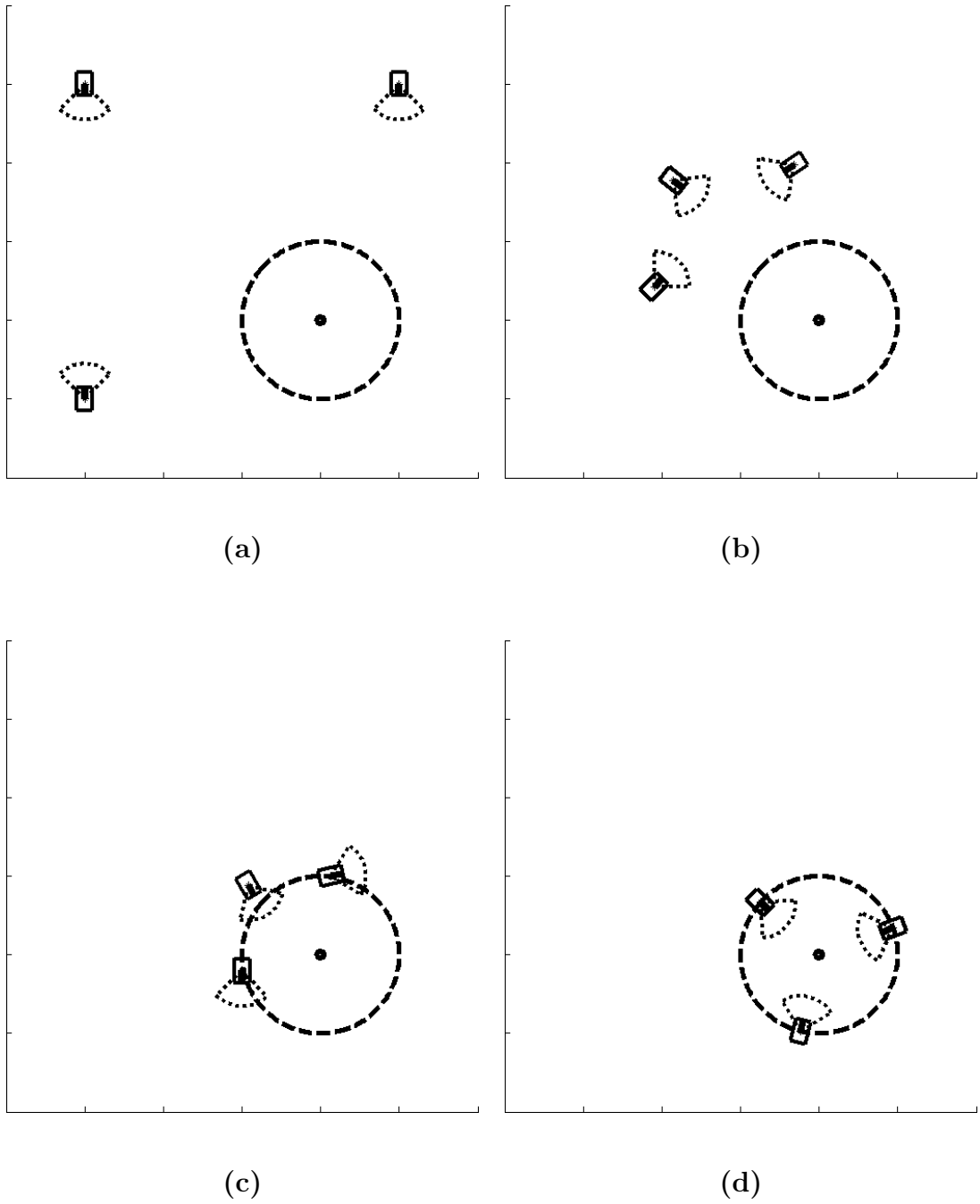


Figure 6.13: Kinematic coordination model, Scenario-2: (a)Initial configuration (b)Coordination dominant (c)Distribution on circle (d)Desired formation achieved

Scenario-3

This simulation was run for a group of four autonomous nonholonomic mobile robots. Since R_i is effected only by its closest two neighbors, the farthest robot with respect to R_i doesn't affect R_i in cases where number of robots is higher than three.

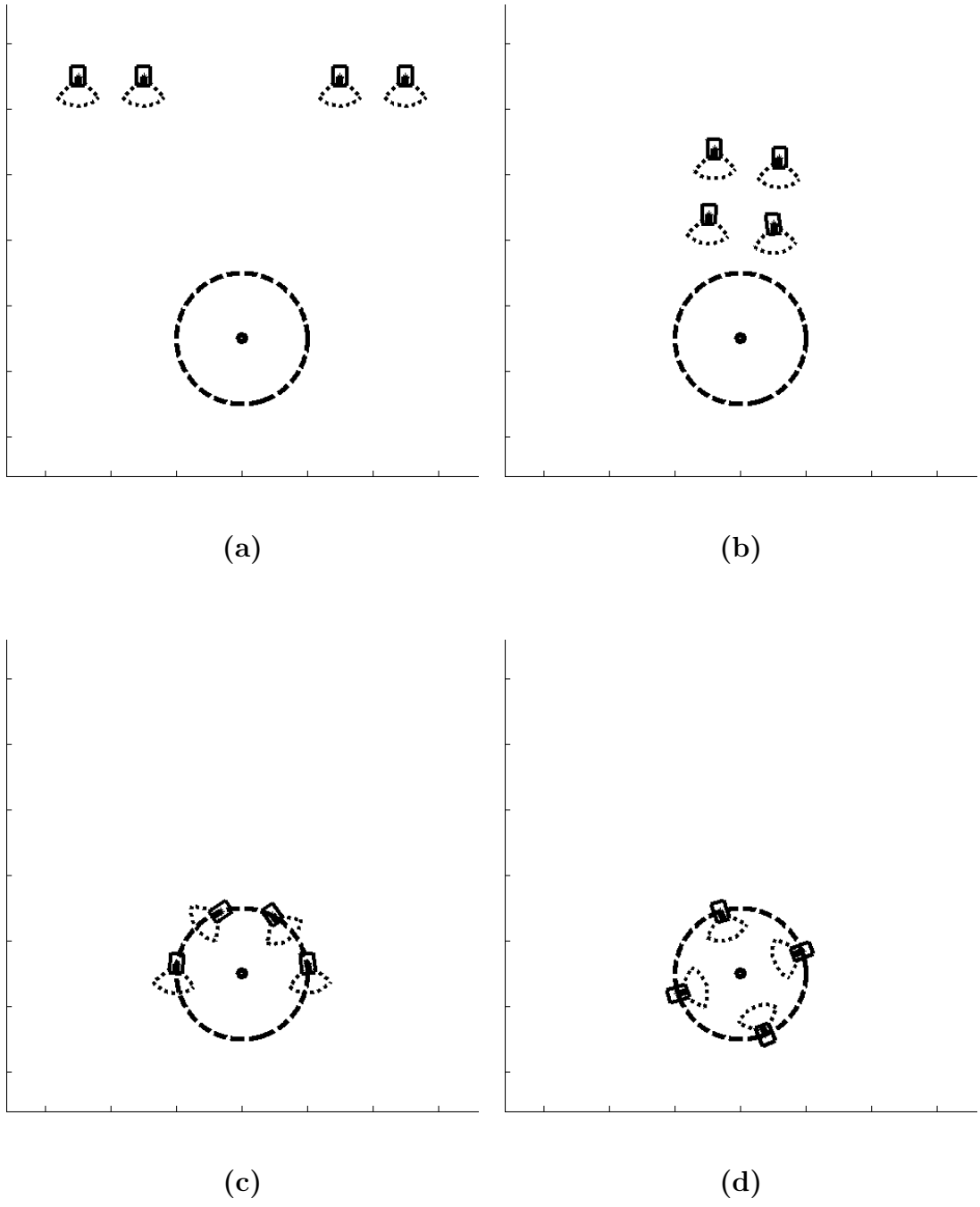


Figure 6.14: Kinematic coordination model, Scenario-3: (a)Initial configuration (b)Coordinated motion (c)Distribution on circle (d)Desired formation achieved

This scenario has the initial setting given in Fig. 6.14(a). Since coordination is dominant in phase (A), the robots approach each other, instead of going directly towards T, and form a square with sides d_{far} as shown in Fig. 6.14(b). The result is a circular formation with the robots on the corners of a square of sides d_{near} as depicted in Fig. 6.14(d).

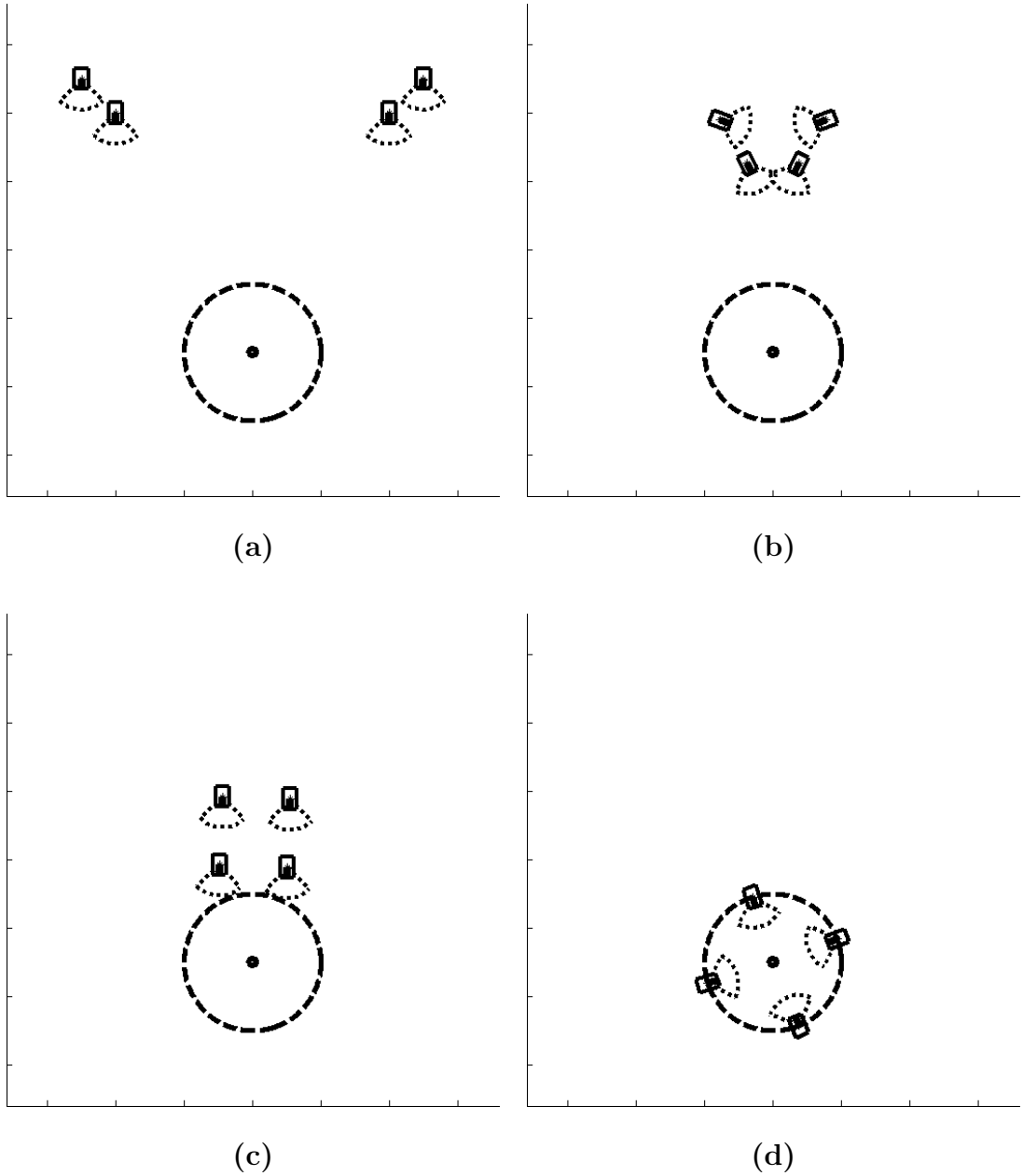


Figure 6.15: Kinematic coordination model, Scenario-4: (a)Initial configuration (b)Coordination dominant (c)Coordinated motion (d)Desired formation achieved

Scenario-4

This simulation was also run for a group of four autonomous nonholonomic mobile robots.

The initial configuration of this scenario is almost the same as the initial configuration of Scenario-3, except that the symmetry of the robots is perturbed as seen in Fig. 6.15(a). The results were similar to results of Scenario-3. This reveals the

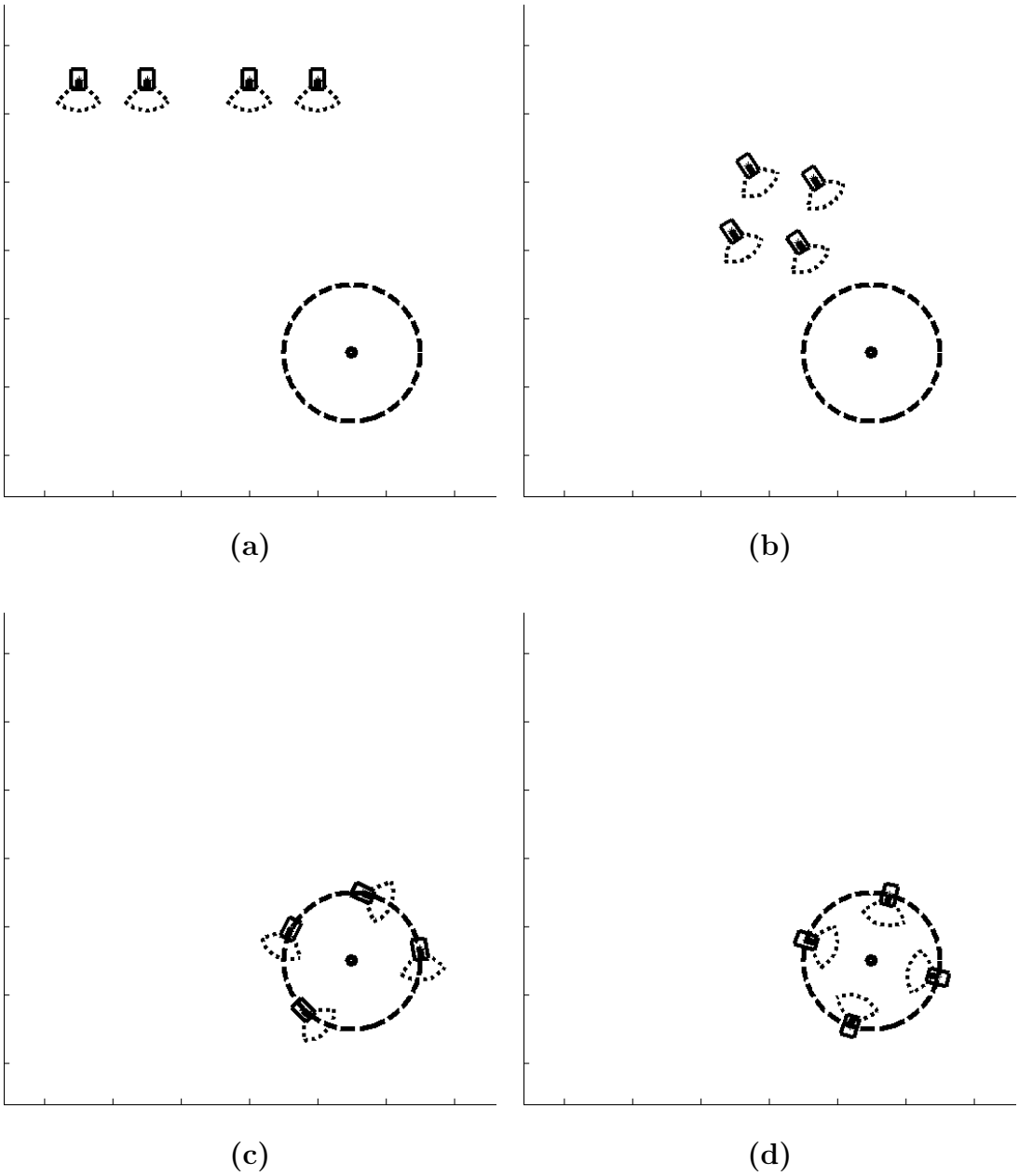


Figure 6.16: Kinematic coordination model, Scenario-5: (a)Initial configuration (b)Coordinated motion (c)Distribution on circle (d)Desired formation achieved

fact that, there's no constraint on the initial configuration for the proposed model to yield satisfactory results. Snapshots from this animation are given in Fig. 6.15.

Scenario-5

The initial setting of this scenario is very similar to the initial configuration of Scenario-1, except that the group consists of four robots. The only difference is that the robots move as a parallelogram instead of a triangle when approaching T in coordination. This proves the success of coordination with the closest two neighbors in achieving the coordinated motion as a group even when there are more than three robots. Fig. 6.16 shows some snapshots of this animation.

Scenario-6

This simulation was run for a group of five autonomous nonholonomic mobile robots. Since R_i is effected only by its closest two neighbors, the farthest two robots with respect to R_i don't affect R_i in such a group of five robots.

The initial setting is depicted in Fig. 6.17(a). As there are more robots, the risk of collision increases for the same value of d_{targ}. As depicted in Fig. 6.17(b), some collisions were predicted around the formation circle, but they were successfully avoided and the robots started circular motion on the formation circle as shown in Fig. 6.17(c). The final formation is satisfactory with a uniform distribution of robots as shown in Fig. 6.17(d).

The videos of the animations, snapshots from which are presented above, can be found in the CD enclosed at the back of this thesis in the folder named `SimVideos`. The nomenclature is such that each video has file name extension `.avi` and file name same as the number of the figure presented above. For example, the video file of the animation from which snapshots are given in Fig. 6.5 is named `65.avi`. The videos are also currently available on the following web site:

`http://students.sabanciuniv.edu/ nusrettin/cd/SimVideos`

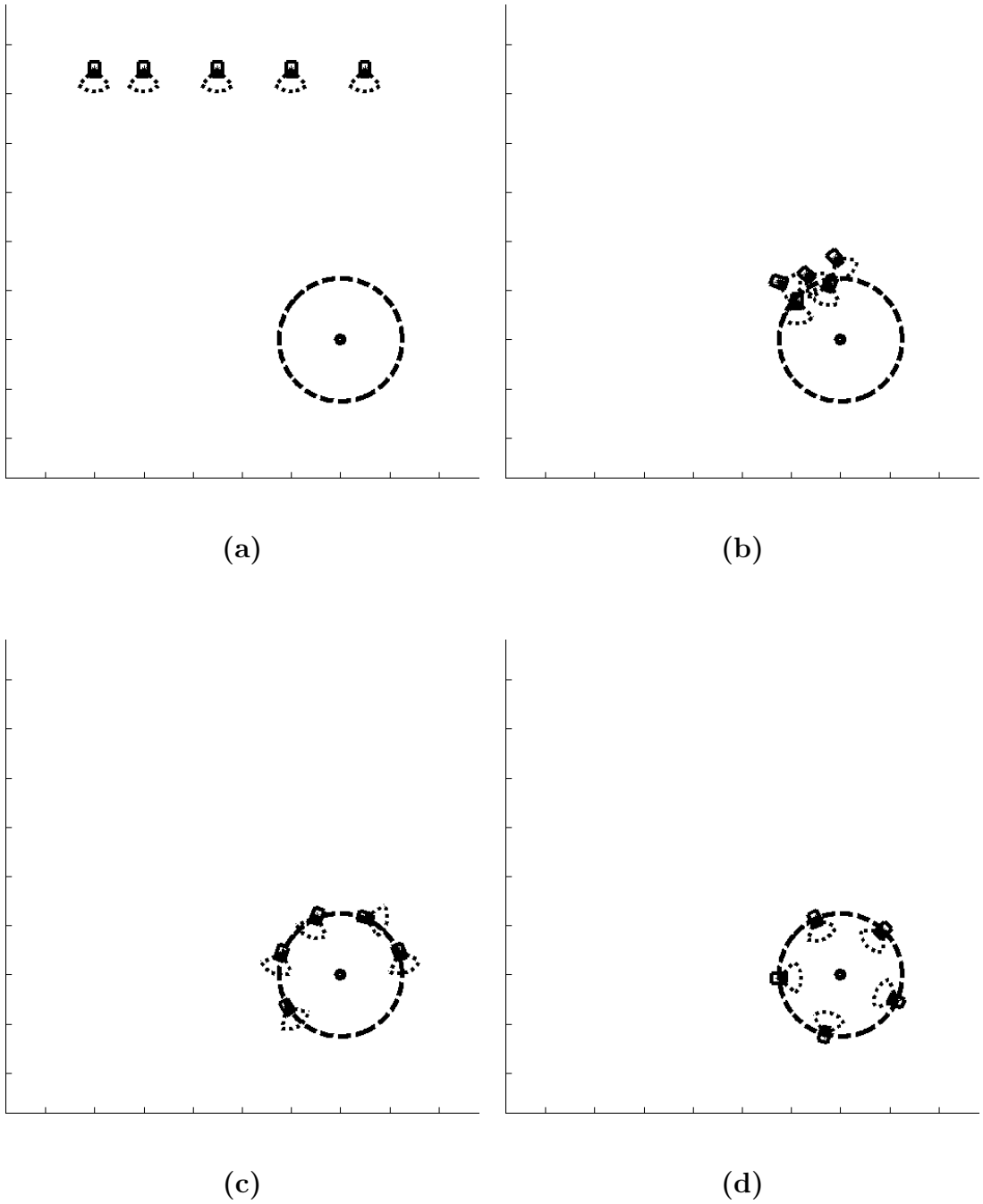


Figure 6.17: Kinematic coordination model, Scenario-6: (a)Initial configuration (b)Collision avoidance (c)Distribution on circle (d)Desired formation achieved

The results of the simulations for the proposed kinematic coordination model with three, four and five robots are all satisfactory. Moreover, the consistent success of the algorithm across the increase in the number of robots is promising for the modular and autonomous nature of the system.

6.3 Experiments

The collision avoidance algorithm proposed in this thesis and explained in detail in Section 4.3 was tested experimentally using two of the autonomous robots shown in Fig. 6.18.

Each autonomous robot consists of four main parts:

- A unicycle robot equipped with Board of Education(BOE), namely Boe-Bot.
- A Philips PCVC 830 webcam.
- A cylinder on top for identification in image processing.
- A PC where image processing and collision avoidance algorithms are run.

The Boe-Bot, seen in Fig. 6.19(a), is an off-the-shelf and cheap unicyle robot designed for educational purposes. The robot consists of a base to which 2 servomotors are attached for the right and left wheels, and a Boe-Board is mounted. The Boe-Board consists of digital input-output pins, the power circuitry for the servos,

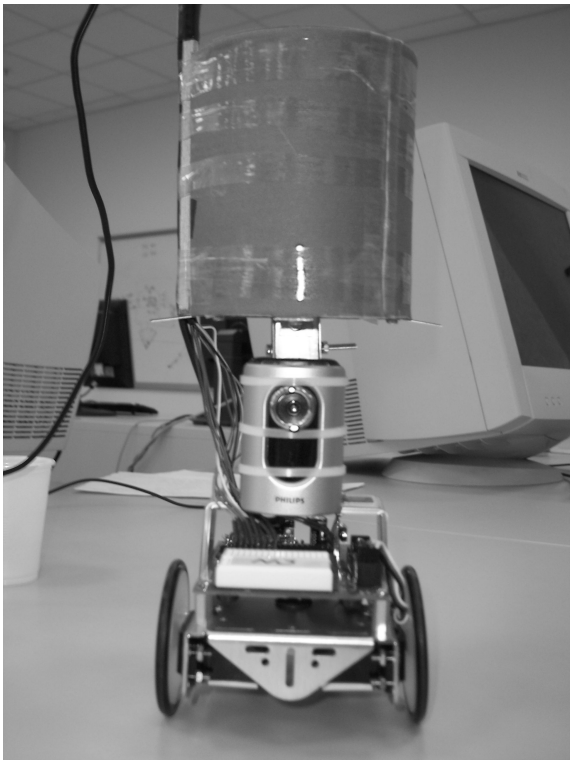


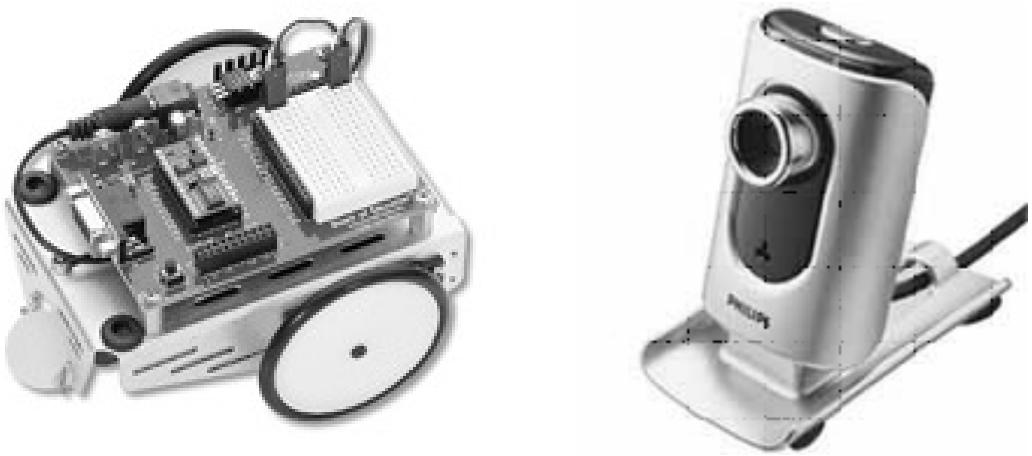
Figure 6.18: Autonomous robot prepared for experiment

a serial programming port, and a PIC that can process digital signals. The PIC is programmed via Basic Stamp II, a programming language very similar to classical BASIC. More information on Boe-Bot and Basic Stamp II can be found in Appendix A.

The Philips PCVC 830 camera, seen in Fig. 6.19(b), is mounted on the robot to acquire images to gather information about the environment along with image processing. The camera is capable of capturing 30 frames per second with resolution 320×240 pixels.

The cylinder mounted above the robot is a simple cylindrical can wrapped with colored paper. The cylinder is used to indicate that there's a robot in the image. Specifically, a cylindrical structure is chosen due to the ease its symmetrical structure provides in image processing; i.e. a cylinder's diameter is constant looking from any viewpoint.

The camera is connected to a PC, where image processing and control algorithms are carried out, by a USB connection. On the other hand, the result of the algorithm, i.e. the generated reference linear and angular velocities, are sent back to the robot via the parallel port. Information about the parallel port and accessing its output pins in Windows XP environment can be found in Appendix B. Since the parallel port has 8 output pins, the velocity information is sent to each wheel via 4-bits.



(a) **(b)**

Figure 6.19: Components of experimental setup: (a)Boe-Bot unicycle robot (b)Philips PCVC 830 camera

With these bits, the robot could have 14 speed levels, front and back, since at least one of the 16 values that can be coded by 4-bits should stop the motors. However, Basic Stamp II can compile codes if and only if they are shorter than 675 lines. The nonlinear nature of the velocities of the servomotors with respect to the applied pulse widths requires `if-then-else` loops for every distinct situation. In those 675 lines, only 6 gears could be coded, 4 forward and 2 backward.

The image processing is done using OpenCV library under C++. OpenCV implements a wide variety of tools for image interpretation. It is mostly a high-level library implementing algorithms for calibration techniques, feature detection and tracking, shape analysis, object segmentation and recognition. The essential feature of the library is its high performance. More information about OpenCV along with a template code for beginners can be found in Appendix C.

To conduct the experiments, a room of $2m \times 2m$ was constructed for the robots to navigate inside. To ensure proper detection of the cylinder in the image processing algorithm, the walls of the room were built using white paper. The features to track were chosen as the vertical lines implying the sides of the cylinder. However, vertical sides of the cylinder were tilted on the image due to perspective projection, since the lens of the camera was below the center of the cylinder by $5cm$. To get rid of this tilt effect, a tolerance was introduced on the slope of the detected lines to interpret them as vertical lines. The distance between midpoints of the two lines was used to extract information about how far the sensed robot is. More information on perspective projection can be found in Appendix D.

6.3.1 PseudoCode

Each autonomous mobile robot runs the following algorithm during the experiments:

1. Image capture and enhancement.
    - Capture the image, convert it to grayscale for faster processing.
2. Extraction of features.
    - Carry out line detection by the built-in function, `cvHoughlines2`.
    - If there are at least 2 vertical lines:

- Find the highest and lowest values among the x pixel coordinates of the midpoints of the extracted lines.
- Find the difference between those values to obtain the diameter of the cylinder; hence the approximate depth information.
- Find the midpoint of the above extracted points and interpret as the center of the cylinder; i.e. the sensed robot's position.
- By perspective projection, calculate the position of the sensed robot.

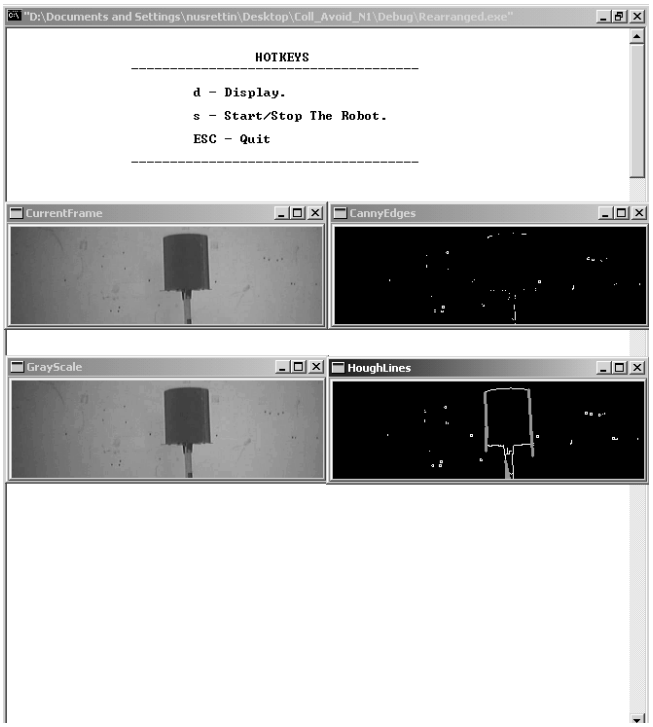
3. Calculation of the reference orientation.

   - Run the collision avoidance algorithm to decide on the reference orientation. Keep the orientation constant if a collision is not predicted.

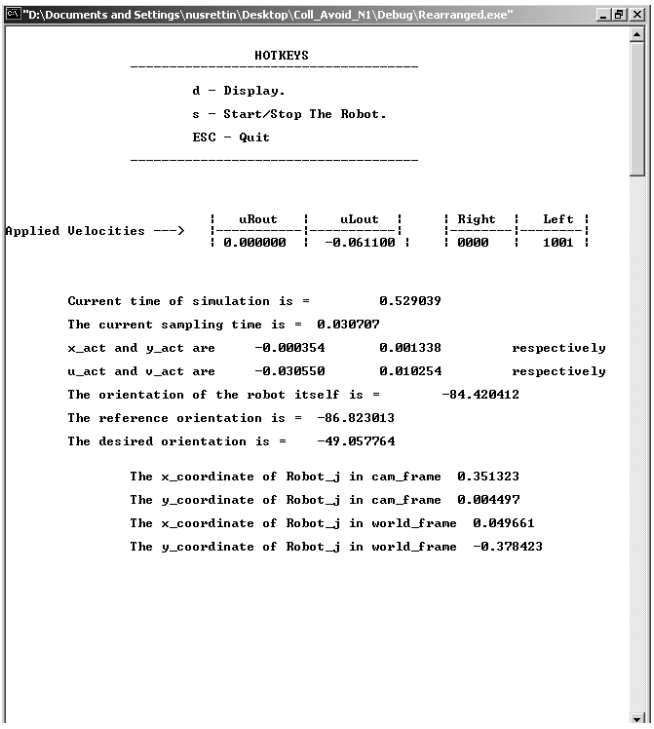
6.3.2 Results

Experiments for the case of a static obstacle and a moving obstacle were conducted using the given algorithm above for each autonomous mobile robot. The `C++` code generated to be run on each robot can be found in the CD included at the back of this thesis with the name "`CollisionAvoidance.cpp`".

A sample run of the generated program is shown in Fig. 6.20(a). In this sample, the sides of the cylinder are detected as vertical lines. The distance between the midpoints of these lines is extracted by image processing tools offered by OpenCV. The so obtained diameter of the cylinder is used to extract depth information for the sensed robot; hence the pose of the detected robot in world coordinate system is reconstructed at each computational step. Essential information such as pose of the sensed robot, the sampling time, etc is displayed on the screen when '`d`' is stroke on the keyboard. A sample display of the information is shown in Fig. 6.20(b). The current step time for the computations is $30.707ms$, which yields an approximate frequency of $30Hz$ for the generated program to be cycled.



(a)



(b)

Figure 6.20: Screen shots of the generated `C++` code: (a)Detection of the lines on the image (b)Essential information displayed when '`d`' is pressed

6.3.3 Static Obstacle Avoidance

To enable static obstacle detection, a stationary robot was used as a static obstacle. Snapshots from this experiment are given in Fig. 6.21. In this scenario, A serves as a static obstacle. B approaches A from the depicted initial position of Fig. 6.21(a). B predicts a collision at the instant given in Fig. 6.21(b). After adjusting its orientation as seen in Fig. 6.21(c), B moves with a constant velocity on the collision-free path as seen in Fig. 6.21(d).

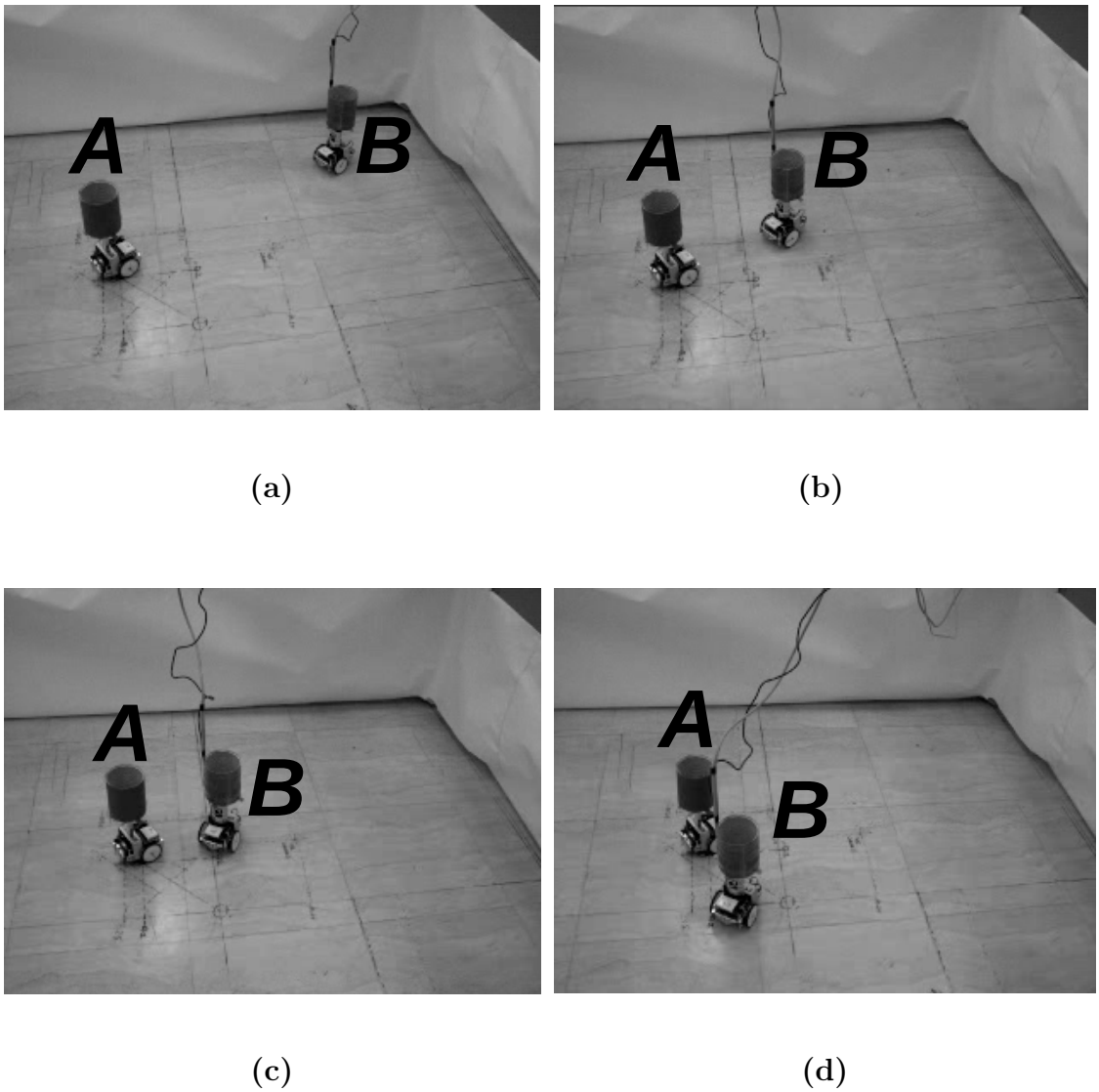
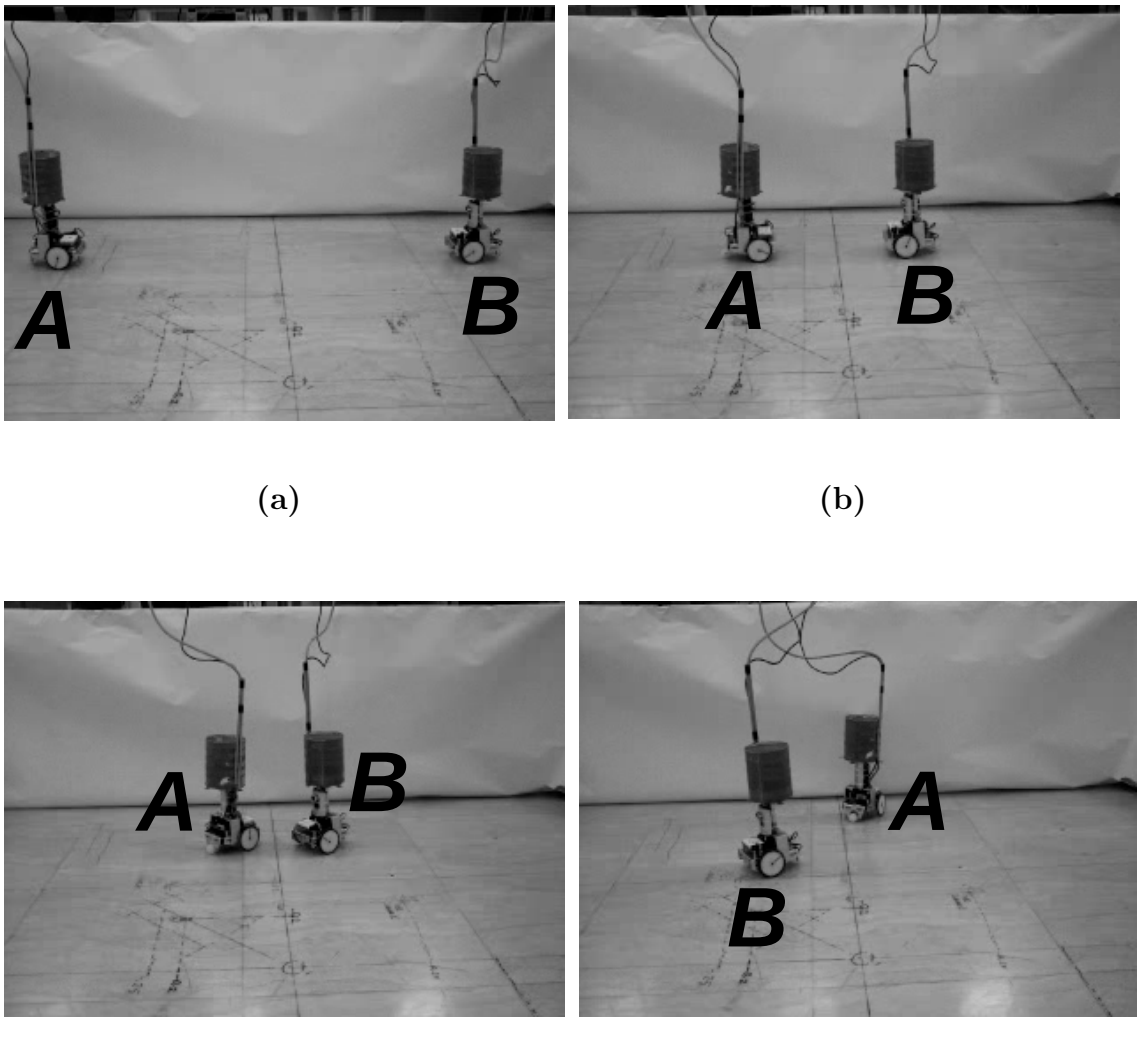


Figure 6.21: Static obstacle avoidance experiment: (a)Initial configuration (b)Collision prediction (c)Adjusting orientation (d)Collision avoided

6.3.4 Head-to-Head Collision Avoidance

In this scenario, the robots are headed towards each other as depicted in Fig. 6.22(a). As they approach each other, they sense the other and calculate its position and velocity. When the distance between them becomes equal to r_{coll}, they both change their orientations and move safely away, without any collisions. The snapshots of this experiment can be seen in Fig. 6.22.



(a) **(b)**

(c) **(d)**

Figure 6.22: Head to head collision avoidance experiment: (a)Initial configuration (b)Collision predictions (c)Adjusting orientations (d)Collision avoided

The videos of the experiments, snapshots from which are presented above and some others, can be found in the CD enclosed at the back of this thesis in the folder named `ExpVideos`. The nomenclature is the same as animation videos. For example, the video file of the experiment from which snapshots are given in Fig. 6.21 is named `621.avi`. The videos are also currently available on the following web site:

`http://students.sabanciuniv.edu/ nusrettin/cd/ExpVideos`

Chapter 7

Conclusions

In this thesis, two novel approaches for the coordinated motion of a group of autonomous mobile robots, aimed for the achievement of a coordinated task, have been proposed. A novel online collision avoidance algorithm is inherent in both of the proposed models. The test beds considered are unicycle type autonomous nonholonomic mobile robots. For simplicity in analysis, reliable communication protocols between these robots were assumed, to demonstrate the satisfactory performance of the proposed models.

A virtual reference system that consists of virtual mass-spring-damper units is introduced in the first model. The reference trajectories for the autonomous robots are obtained from the dynamics of this computer-generated system. A feedback path between the actual robots and the virtual reference system was constructed enabling online generation of references. Together with the proposed collision avoidance algorithm, this system generates online collision-free reference trajectories, required to manipulate a coordinated task. The performance of this approach was tested in computer simulations and the results are promising.

In the second approach, online generation of references for the autonomous robots in terms of their linear and angular velocities is enabled. This model consists of virtual reference robots, the linear and angular velocities of which are designed for the coordinated motion and coordinated task manipulation of a group of robots. The reference trajectories to be used in the control laws are then generated by the integration of so-obtained reference velocities. The essential advantage of this approach is that possible nonholonomy of the robots is dealt with in the phase of reference generation, so that the regulation of tracking errors to zero is guaranteed.

The results of the simulations run for the performance test of this algorithm are also satisfactory.

In addition to the proposed approaches to the problem of modeling the coordinated motion of a group of autonomous mobile robots, some other problems were attacked. A novel algorithm for collision avoidance - that is inherent to any coordinated motion problem - has been proposed. In the proposed algorithm, the autonomous robots use sensory information to predict collisions ahead of time by the introduction of a virtual circular arc. The algorithm updates the velocities of the robots in case of such a collision prediction to avoid any possible collisions with the other members of the group or any static obstacles. This algorithm is used in both of the proposed coordinated motion models. The problem of the group's achievement of certain formations to accomplish certain coordinated tasks has also been addressed. Some parameters of the developed models were switched throughout the manipulation of the coordinated task to enable achievement of required formations. For example, the equilibrium length of the virtual springs in the first approach were switched. For this parameter switching problem, a sigmoid function was used to facilitate smooth response of the robots.

Experiments were conducted to test the performance of the collision avoidance algorithm using BoeBot type robots, a brand of unicycle robots, and OpenCV. The performance of the proposed collision algorithm was promising both in the simulations of coordinated motion and in the experiments where any coordinated behavior was absent for simplicity. However, the experiments could have been extended to more complicated scenarios of collision avoidance and tests of coordinated motion approaches could have been carried out if the robots that are used could work with higher precision.

The work could be extended to the construction of appropriate communication protocols between the robots. Once this is done, the coordinated motion of a group of autonomous mobile robots for various coordinated tasks such as search and rescue could be realized by the aid of the proposed models in this thesis.

Appendix A

Boe-Bot and Basic Stamp

A.1 Boe-Bot

[1] The Boe-Bot is a unicycle type mobile robot that mechanically consists of the parts listed below.

- (1) Boe-Bot chassis.
- (1) Battery pack.
- (2) Parallax pre-modified servos.
- (2) Plastic wheels and tires.
- (1) Polyethylene ball.
- (1) Board of Education and BASIC Stamp II.

The servos, to which the wheels are connected are mounted to the chassis as well as the other components. The polyethylene ball is used as the back wheel of the robot that provides the equilibrium of the chassis. The robot works with 4 AA size batteries that are used inside the battery pack that is mounted below the chassis.

A.1.1 Parallax Servo Motors

Parallax servomotors shown in Fig. A.1 are used to drive the wheels of the robot.

The RC type servo motors are originally fabricated for a limited rotational motion range. For a complete rotation of $(2\pi)rad$, they are modified. The mechanical stop is removed and the potentiometer originally placed for encoding rotational position is replaced by a resistor pair. Parallax servos with the label `PM` are pre-modified for such purposes.

[1] Parallax Inc. Educational Materials, Robotics Student Workbook Version 1.5

Figure A.1: Parallax pre-modified servomotor

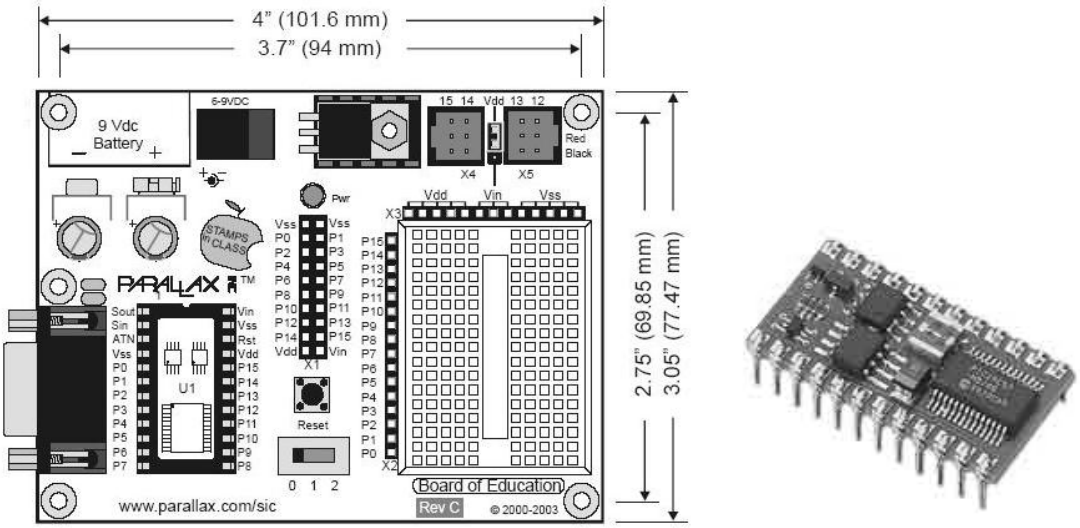
The servo's control line is used to send the motor a "pulse train" that encodes the velocity reference. Pulse width is what controls the servo's motion. The low time between pulses can range between $10ms$ and $40ms$ without adversely affecting the servo's performance.

A pre-modified servo can be pulsed to make its output shaft turn continuously. The pulse widths for pre-modified servos range between $1.0ms$ and $2.0ms$ for full speed clockwise and counterclockwise, respectively. For example, pulses of width $1.25ms$ will make the shaft turn clockwise at almost half of the full speed. The so-called "center pulse width" is around $1.5ms$, and makes the servo stay still. Due to errors in fabrication and resistive tolerances, the center pulse width might deflect from this value.

A.1.2 Board of Education and Basic Stamp II

The Board of Education, BOE, is designed to teach students the basics of robotics. It is compliant with Basic Stamp II, a micro-controller that can process digital signals. The necessary power and digital signals necessary for running the servos in Boe-Bot are processed by the Board of Education.

The input pulse trains applied to the motors are generated by Basic Stamp II micro-controller and processed by the Boe-Board. The power to the servos is regulated and supplied by the circuitry on the board. The Boe-Board also contains additional circuitry for special applications of Boe-Bot such as battery-low alarm. There's a spare breadboard for possible modifications to the Boe-Board. Students can experiment using circuits built on that and the Basic Stamp programming.



(a) **(b)**

Figure A.2: Board of Education, BOE, and Basic Stamp II: (a)Board of Education, Rev. B (b)Basic Stamp II micro-controller

A.2 Basic Stamp

Basic Stamp is a preliminary programming language that is used to program the micro-controller Basic Stamp II. For applications of Boe-Bot, the language is generally used to control the velocity of wheels the robot. The pulse widths of the input pulse trains are controlled by certain code generated in this language.

The generated code to program the robots for the experiments of this thesis can be found in the CD enclosed at the back of this thesis in the folder named `BasicStamp`.

Appendix B

Parallel Port

For the experiments in this thesis, the explained algorithm is run in `C++` along with OpenCV library. The result of the algorithm, i.e. the generated reference linear and angular velocities, are sent back to the robot via the parallel port.

The parallel port has 25 pins 8 of which can be used as digital outputs. The pin configuration of the parallel port is depicted in Fig. B.1. 8 pins of the data register is used to send the robot velocity information.

The parallel port in Windows XP cannot be controlled directly using any programming language. Activating the parallel port outputs in Windows XP is explained in the document with the name "`Activation.doc`". This document can be found in the CD included at the back of this thesis under the folder named `ParallelPortXP`. This folder also includes the program "userport" that is required to activate direct access to the ports in Windows XP.

Once the parallel port output pins are activated, the signal at each pin can be pulled up to high value around $5V$ or pulled down to a low value around $0V$. In `C++`, the following code declares and writes a specified value to the parallel port's

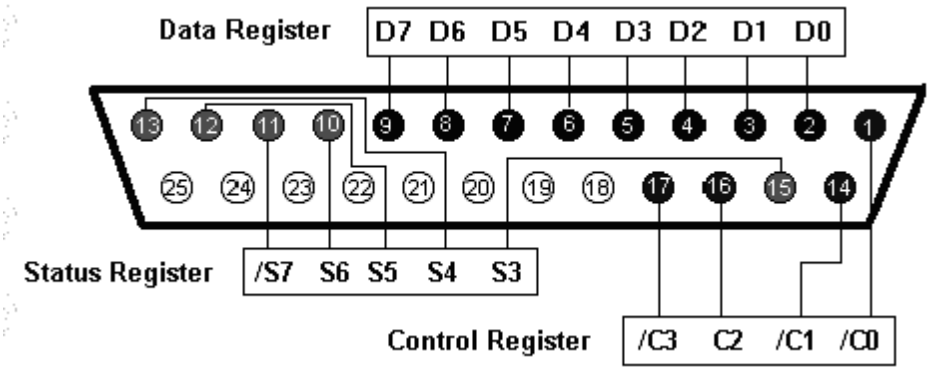


Figure B.1: Parallel port pin configurations

data register. The value of the variable "parout" can be changed by any control algorithm between lines 3 and 4.

```
int _outp(unsigned short port, int databyte );

int IN0=0, IN1=0,IN2=0, IN3=0, //Parallel port output pins.
    IN4=0, IN5=0, IN6=0, IN7=0;

int parout=0;    // Byte to be output from parallel port.

_outp(0x378,parout);   // Output the data to parallel port.
```

Appendix C

OpenCV

[1] OpenCV means Intel® Open Source Computer Vision Library. It is a collection of C functions and few C++ classes that implement some popular algorithms of Image Processing and Computer Vision.

OpenCV is cross-platform middle-to-high level API that consists of a few hundreds C functions. It does not rely on external numerical libraries, though it can make use of some of them (see below) at runtime, if they are available.

More references on the functions and libraries of OpenCV can be found at: `http://www.cs.bham.ac.uk/resources/courses/robotics/doc/opencvdocs/`

C.1 Installation

The installation of OpenCV in Windows XP requires certain environments to be set and certain libraries to be copied to some specific directories. Please follow the instructions in "`Installation.doc`" to get rid of problems. This instruction manual is located under the folder named `OpenCV` in the included CD of this thesis.

C.2 Template Code for Beginners

The following is a sample code for beginners of OpenCV. This code can also be found in the folder named `OpenCV` with the folder name `WorkingTemplate OpenCV` in the enclosed CD.

[1] `http://www.cs.bham.ac.uk/resources/courses/robotics/doc/opencvdocs/`

```
//--------------------------------------------------------------//
//------------------- OPENCV TEMPLATE SOURCE -------------------//
//---------- Nusrettin GULEC, 5212, Sabanci University ----------//
//------------------- CopyRight (c), 2005, SU -------------------//
//--------------------------------------------------------------//
//**************************************************************//
////////////////////////////////////////////////////////////////
//**************************************************************//
//--------------------------------------------------------------//
//---------------- START YOUR LIBRARIES TO INCLUDE --------------//
//**************************************************************//
// Declare that this is a code written for OpenCV library.
#ifdef _CH_ #pragma package <opencv> #endif
//--------------------------------------------------------------//
// For an OpenCV source code, you should include the following.
#ifndef _EiC #include "cv.h"         // Main OpenCV Library.

#include "highgui.h"    // For high-level GUI functions;

#include <windows.h>    // To open windows and display images.
//--------------------------------------------------------------//
// Any other type of regular libraries that you want to include.
// #include <Library or header file>
#include <stdio.h>      // Standard I/O Library.

#include <ctype.h> // Alpha-numeric settings for character strings
#endif
//**************************************************************//
//---------------- END YOUR LIBRARIES TO INCLUDE ----------------//
//--------------------------------------------------------------//
//**************************************************************//
////////////////////////////////////////////////////////////////
//**************************************************************//
//--------------------------------------------------------------//
//--------------- START YOUR FUNCTION DEFINITIONS --------------//
//**************************************************************//
// void, double, int, etc... any type of function def.s here.
// <ReturnType> <FunctionName> (<Type> <Name>)
// {
//      internal variable declarations.
//      whatever the function does.
//      <return> statement if not void type.
// }
//--------------------------------------------------------------//
/* Examples: double SgnForDoubles(double param)
{
    double SgnParam;
    if (param == 0)
```

```
    {
        SgnParam = 0;
    }
    else
    {
        SgnParam = param/fabs(param);
    }
    return SgnParam;
}
void IncrementPositive(double incremented) {
    if (incremented > 0)
    {
        incremented ++;
    }
}*/
//*****************************************************************//
//---------------- END YOUR FUNCTION DEFINITIONS -----------------//
//-----------------------------------------------------------------//
//*****************************************************************//
///////////////////////////////////////////////////////////////////
//*****************************************************************//
//-----------------------------------------------------------------//
//--------------- START YOUR VARIABLE DECLARATIONS ---------------//
//*****************************************************************//
int main( int argc, char** argv )
{
//-----------------------------------------------------------------//
// Main image processing variables.
    CvCapture* captured = 0;  // BlackBox Structure For Capturing.

    IplImage* frame = 0;      // Storage for captured frame.
    IplImage* gray = 0;       // Storage for grayscale image.

    CvRect RegOfInt;          // Region of Interest to be used.
    int RoiSideX=200;   // Desired size of a ROI to be taken in x.
    int RoiSideY=200;   // Desired size of a ROI to be taken in y.
//-----------------------------------------------------------------//
// Any variables or constants you will define for your algorithm.
// <VariableType> <VariableName> = <Value>;
//-----------------------------------------------------------------//
/* Examples:
// const double pino = 3.141592653589793;     // The pi number.
// bool Displayer = false;
// double ParameterForControl = 0;  */
//*****************************************************************//
//---------------- END YOUR VARIABLE DECLARATIONS ---------------//
//-----------------------------------------------------------------//
//*****************************************************************//
```

```
////////////////////////////////////////////////////////////////////
//****************************************************************//
//----------------------------------------------------------------//
//----------------- START YOUR IMAGE PROCESSING ------------------//
//****************************************************************//
// Capture an image.
    captured = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
//----------------------------------------------------------------//
// Give Error Message if cannot capture.
    if( !captured )
    {
        fprintf(stderr, "\n\n\tERROR!!!\n"
                        "\tCapturing Couldn't Be Started;\n"
                        "\tCheck Your Connections!\n\n\n");
        return -1;          // Exit!
    }
//----------------------------------------------------------------//
// Windows For Displaying the frames.
    cvNamedWindow( "CurrentFrame", 1 );
    cvNamedWindow( "GrayScale", 1 );
//----------------------------------------------------------------//
// Print the user's manual on screen.
printf(
"\n\n\t\t\t\tHOTKEYS\n\t\t-------------------------------------\n\n"
"\t\t\tESC - Quit \n\n"
"\t\t-------------------------------------\n\n\n");
//----------------------------------------------------------------//
// Here is the repeated part of the program
// unless a 'break' is encountered.
    while(1)
    {
        frame = cvQueryFrame( captured );  // Get current frame.

        CvRect CurrRoi = cvGetImageROI(frame);      // Get the current
        RegOfInt.x = (CurrRoi.width-RoiSideX)/2;    // frame's size and
        RegOfInt.y = (CurrRoi.height-RoiSideY)/2;   // change the ROI
        RegOfInt.width  = RoiSideX;                 // according to the
        RegOfInt.height = RoiSideY;                 // desired
        cvSetImageROI( frame, RegOfInt);            // ROI dimensions.
//----------------------------------------------------------------//
// Convert frame to grayscale.
        gray = cvCreateImage( cvGetSize(frame), IPL_DEPTH_8U, 1);
        cvConvertImage(frame, gray, CV_CVTIMG_FLIP);
//----------------------------------------------------------------//
// Apply any kind of image processing algorithm here.
/* Example:
// Form the new image and apply Canny's edge detection algorithm.
    edges = cvCreateImage( cvGetSize(gray), IPL_DEPTH_8U, 1 );
```

```
    cvCanny(gray,edges,EdgeThreshLow,EdgeThreshHigh,3); */
//*****************************************************************//
//-------------------- END YOUR IMAGE PROCESSING -----------------//
//-----------------------------------------------------------------//
//*****************************************************************//
///////////////////////////////////////////////////////////////////
//*****************************************************************//
//-----------------------------------------------------------------//
//------------------ START YOUR CONTROL ALGORITHM ----------------//
//*****************************************************************//
// Apply any kind of controls you might use in the implementation.
//-----------------------------------------------------------------//
/* Example:
while (ParameterForControl)
{
    ParameterForControl = SgnForDoubles(ParameterForControl);
    IncrementPositive(ParameterForControl);
}*/
//*****************************************************************//
//------------------- END YOUR CONTROL ALGORITHM -----------------//
//-----------------------------------------------------------------//
//*****************************************************************//
///////////////////////////////////////////////////////////////////
//*****************************************************************//
//-----------------------------------------------------------------//
//-------------- START YOUR OUTPUTS AND KEY CONTROLS -------------//
//*****************************************************************//
// Apply any image processing and keyboard controls.
        if( !frame )        // Break if frame can't be captured.
            break;
//-----------------------------------------------------------------//
// Show the processed images.
        cvShowImage( "CurrentFrame", frame );
        cvShowImage( "GrayScale", gray );
//-----------------------------------------------------------------//
// Wait for keystroke on keyboard.
        int PressedKey = cvWaitKey(10);
//-----------------------------------------------------------------//
        if( PressedKey == 27 )      // Break if 'ESC' is stroke.
            break;
//-----------------------------------------------------------------//
/* Example: Other possible key controls.
switch( PressedKey )
{
    case 'd':
    Displayer = true;
    break;
```

```
    default:;
}*/ }
//*****************************************************************//
//-------------- START YOUR OUTPUTS AND KEY CONTROLS -------------//
//-----------------------------------------------------------------//
//*****************************************************************//
///////////////////////////////////////////////////////////////////
//*****************************************************************//
//-----------------------------------------------------------------//
//----------------- DESTROY AND EXIT THE PROGRAM ----------------//
//*****************************************************************//
// Destroy the current windows and current data before
// exiting in reverse order you opened them.
    cvDestroyWindow("GrayScale");
    cvReleaseImage( &gray);
    cvDestroyWindow("CurrentFrame");
    cvReleaseCapture( &captured );
//-----------------------------------------------------------------//
// Exit message when 'break' is encountered from while.
printf( "\n\n\t\tThe Program Will Now Exit!\n\n\t\t");
//-----------------------------------------------------------------//
    return 0;       // Exit.
}
//*****************************************************************//
//-------------- END OF OPENCV TEMPLATE SOURCE CODE --------------//
//-----------------------------------------------------------------//
//*****************************************************************//
///////////////////////////////////////////////////////////////////
//*****************************************************************//
//-----------------------------------------------------------------//
//------------------- OPENCV TEMPLATE SOURCE ---------------------//
//-------- Nusrettin GULEC, 5212, Sabanci University ------------//
//------------------ CopyRight (c), 2005, SU --------------------//
//-----------------------------------------------------------------//
//*****************************************************************//
```

Appendix D

Perspective Projection and Camera Model

[1] A simplified model of image formation is shown in Fig. D.1. This is called pinhole model. In this model, the lens is assumed to be a single point. The world coordinates (X,Y,Z) of a point are transformed to the image coordinates (x,y) under perspective projection. If the origin is located at the lens center as in Fig. D.1(a) and f is the focal length of the camera, the distance from the lens to the image plane. Then using the similarity in triangles, the following can be written:

$$\frac{-y}{Y} = \frac{f}{Z} \, . \tag{D.0.1}$$

Since the image is formed upside down, the negative sign in often used in the above equation. The x and y image coordinates can be extracted from the above equation as:

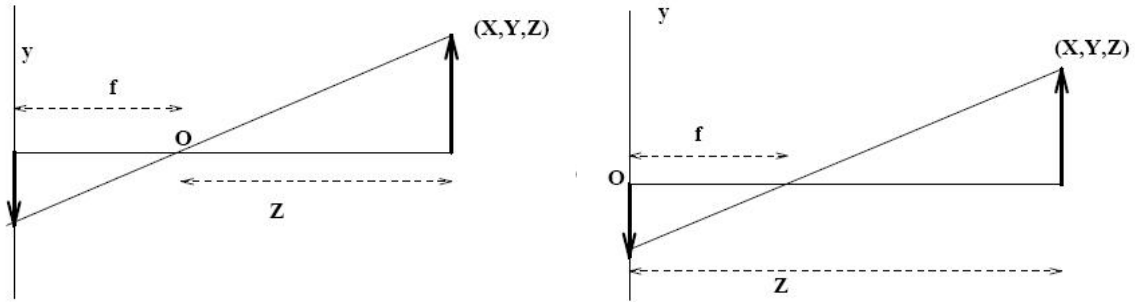
$$x = -\frac{fX}{Z} \, , \qquad\qquad y = -\frac{fY}{Z} \, . \tag{D.0.2}$$

The above equations represent the perspective projection with the origin at the lens. If the origin is moved to the center of image as in Fig. D.1(b), these equations are changed to:

$$x = -\frac{fX}{f-Z} \, , \qquad\qquad y = -\frac{fY}{f-Z} \, . \tag{D.0.3}$$

It is mathematically not possible to derive a perspective matrix such as common translation, rotation or scaling matrices in robotics. Hence, *homogenous transfor-*

[1] Fundamentals of Computer Vision, Mubarak Shah, CS Department, University of Central Florida, Orlando, 1997



(a) **(b)**

Figure D.1: Pinhole camera model: (a)Origin at the lens (b)Origin at the image plane

mations are introduced. The *homogenous transformation* converts the *Cartesian world coordinates* (X,Y,Z) into the *homogenous world coordinates* (kX,kY,kZ,k). Similarly the inverse homogenous transform converts the homogenous image coordinates $(C_{h1},C_{h2},C_{h3},C_{h4})$ into the Cartesian image coordinates $(\frac{C_{h1}}{C_{h4}},\frac{C_{h3}}{C_{h4}},\frac{C_{h3}}{C_{h4}})$. The perspective matrix is then defined as:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix} . \tag{D.0.4}$$

The perspective transformation, which relates the homogenous world coordinates with the homogenous image coordinates is defined as:

$$\begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{f} + k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} . \tag{D.0.5}$$

The Cartesian image coordinates can easily be derived from the above equation as:

$$\begin{aligned} x &= \tfrac{fX}{f-Z} , \\ y &= \tfrac{fY}{f-Z} , \end{aligned} \tag{D.0.6}$$

which is exactly same as equation D.0.3. The inverse perspective matrix is given by:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix} . \tag{D.0.7}$$

Bibliography

[1] T. Suzuki, T. Sekine, T. Fujii, H. Asama, I. Endo, "Cooperative Formation among Multiple Mobile Robot Teleoperation in Inspection Task", *Proceedings of the 39th IEEE International Conference on Decision and Control*, Vol. 1, December 2000, 358-363.

[2] A.K. Das, R. Fierro, V. Kumar, J.P. Ostrowski, J. Spletzer, C.J. Taylor, "A Vision-Based Formation Control Framework", *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, October 2002, 813-825.

[3] J. Kennedy, R.C. Eberhart, Y. Shi, "Swarm Intelligence", *Morgan Kaufmann Publishers*, San Francisco, 2001.

[4] R. Salomon, "The Force Model: Concept, Behavior, Interpretation", *Congress on Evolutionary Computation, CEC2004*, Vol. 1, June 2004, 1119-1126.

[5] J. Lin; A.S. Morse, B.D.O. Anderson, "The Multi-Agent Rendezvous Problem", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Vol. 2, December 2003, 1508-1513.

[6] T.B. Gold, J.K. Archibald, R.L. Frost, "A Utility Approach to Multi-Agent Coordination", *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA '00*, Vol. 3, April 2000, 2052-2057.

[7] S.G. Loizou, H.G. Tanner, V. Kumar, K.J. Kyriakopoulos, "Closed Loop Motion Planning and Control for Mobile Robots in Uncertain Environments", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Vol. 3, December 2003, 2926-2931.

[8] R. Li, Q. Shi, "Study on Integration of Urban Traffic Control and Route Guidance based on Multi-agent Technology", *Proceedings of the 2003 IEEE Interna-*

tional Conference on Intelligent Transportation Systems, Vol. 2, October 2003, 1740-1744.

[9] J.B. de Sousa, F.L. Pereira, "Specification and design of coordinated motions for autonomous vehicles", *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 1, December 2002, 101-106.

[10] Y. Hur, R. Fierro, I. Lee, "Modeling Distributed Autonomous Robots using CHARON: Formation Control Case Study", *Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC '03*, May 2003, 93-96.

[11] P. Seiler, A. Pant, K. Hedrick, "Analysis of Bird Formations", *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 1, December 2002, 118-123.

[12] H. Yamaguchi, "A Cooperative Hunting Behavior by Mobile-Robot Troops", *The International Journal of Robotics Research*, Vol. 18, No. 8, September 1999, 931-940.

[13] H. Yamaguchi, "A Cooperative Hunting Behavior by Multiple Nonholonomic Mobile Robots", *1998 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, October 1998, 3347-3352.

[14] A. Jadbabaie, J. Lin, A.S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules", *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 3, December 2002, 2953-2958.

[15] A. Borkowski, M. Gnatowski, J. Malec, "Mobile Robot Cooperation in Simple Environments", *Proceedings of the Second International Workshop on Robot Motion and Control*, October 2001, 109-114.

[16] S. Yannier, A. Sabanovic, A. Onat, "Basic Configuration for Mobile Robots", *Proceedings of the IEEE International Conference on Industrial Technology*, Vol. 1, December 2003, 256-261.

[17] M.J. Mataric, M. Nillson, K.T. Simsarin, "Cooperative Multi-Robot Box-Pushing", *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, 'Human Robot Interaction and Cooperative Robots'*, Vol. 3, August 1995, 556-561.

[18] A.V. Savkin,"Coordinated Collective Motion of Groups of Autonomous Mobile Robots: Analysis of Vicsek's Model", *IEEE Transactions on Automatic Control*, Vol. 49, No. 6, June 2004, 981-982.

[19] T. Rabie, A. Shalaby, B. Abdulhai, A. El-Rabbany, "Mobile Vision-based Vehicle Tracking and Traffic Control", *Proceedings of the Fifth IEEE International Conference on Intelligent Transportation Systems*, September 2002, Singapore, 13-18.

[20] C. Marques, P. Lima, "Avoiding Obstacles - Multisensor Navigation for Nonholonomic Robots in Cluttered Environments", *IEEE Robotics and Automation Magazine*, Vol. 11, No. 3, September 2004, 70-82.

[21] T. Weigel, J.S. Gutmann, M. Dietl, A. Kleiner, B. Nebel, "CS Freiburg: Coordinating Robots for Successful Soccer Playing", *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, October 2002, 685-699.

[22] M. Perron, H. Zhang, "Coaching a Robot Collective ", *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA '04*, Vol. 4, April 2004, 3443-3448.

[23] W. Sheng, Q. Yang, J. Tan, N. Xi, "Risk and Efficiency: A Distributed Bidding Algorithm for Multi-robot Coordination", *Proceedings of the Fifth World Congress on Intelligent Control and Automation, WCICA 2004*, Vol. 5, June 2004, 4671-4675.

[24] F. Marc, I. Degirmenciyan-Cartault, "Multi-Agent Planning as a Coordination Model for Self-Organized Systems", *IEEE/WIC 2003 International Conference on Intelligent Agent Technology, IAT '03*, October 2003, 218-224.

[25] N. Sarkar, T.K. Podder, "Coordinated Motion Planning and Control of Autonomous Underwater Vehicle-Manipulator Systems Subject to Drag Optimiza-

tion", *IEEE Journal of Oceanic Engineering*, Vol. 26, No. 2, April 2001, 228-239.

[26] Y. Liu, K.M. Passino, M. Polycarpou, "Stability Analysis of One-Dimensional Asynchronous Swarms", *Proceedings of the American Control Conference*, Vol. 2, June 2001, 716-721.

[27] J. Spletzer, A.K. Das, R. Fierro, C.J. Taylor, V. Kumar, J.P. Ostrowski, "Cooperative Localization and Control for Multi-Robot Manipulation", *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, October 2001, 631-636.

[28] J.D. Sweeney, H. Li, R.A. Grupen, K. Ramamritham, "Scalability and Schedulability in Large, Coordinated, Distributed Robot Systems", *Proceedings of the 2003 International Conference on Robotics and Automation, ICRA '03*, Vol. 3, September 2003, 4074-4079.

[29] J.S. Baras, X. Tan, P. Hovareshti, "Decentralized Control of Autonomous Vehicles", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Vol. 2, December 2003, 1532-1537.

[30] S. Souissi, X. Defago, T. Katayama, "Decomposition of Fundamental Problems for Cooperative Autonomous Mobile Systems", *Proceedings of the 24th International Conference on Distributed Computing Systems*, March 2004, 554-560.

[31] K.N. Kutulakos, C.R. Dyer, V.J. Lumelsky, "Provable Strategies for Vision-Guided Exploration in Three Dimensions", *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Vol. 2, May 1994, 1365-1372.

[32] H.S. Oh, C.W. Lee, I. Mitsuru, "Navigation Control of a Mobile Robot based on Active Vision", *1991 International Conference on Industrial Electronics, Control and Instrumentation, IECON '91*, Vol. 2, October 1991, 1122-1126.

[33] S. Vitabile, G. Pilato, F. Pullara, F. Sorbello, "A Navigation System For Vision-Guided Mobile Robots", *Proceedings of the 1999 International Conference on Image Analysis and Processing*, September 1999, 566-571.

[34] O. Shakernia, Y. Ma, T.J. Koo, J. Hespanha, S.S. Sastry, "Vision Guided Landing of an Unmanned Air Vehicle", *Proceedings of the 38th IEEE Conference on Decision and Control*, Vol. 4, December 1999, 4143-4148.

[35] H. Ishiguro, T. Maeda, T. Miyashita, S. Tsuji, "A Strategy for Acquiring an Environmental Model with Panoramic Sensing by a Mobile Robot", *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Vol. 1, May 1994, 724-729.

[36] E.T. Baumgartner, S.B. Skaar, "An Autonomous Vision-Based Mobile Robot" *IEEE Transactions on Automatic Control*, Vol. 39, No. 3, March 1994, 493-502.

[37] Z. Zhang, R. Weiss, A.R. Hanson, "Obstacle Detection Based on Qualitative and Quantitative 3D Reconstruction", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 1, January 1997, 15-26.

[38] B. Kwolek, T. Kapuscinski, M. Wysocki, "Vision-based implementation of feedback control of unicycle robots", *Proceedings of the First Workshop on Robot Motion and Control, RoMoCo '99*, June 1999, 101-106.

[39] P. Steinhaus, M. Walther, B. Giesler, R. Dillmann, "3D Global and Mobile Sensor Data Fusion for Mobile Platform Navigation", *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA '04*, Vol. 4, April 2004, 3325-3330.

[40] H. Ishiguro, K. Kato, S. Tsuji, "Multiple Vision Agents Navigating a Mobile Robot in a Real World", *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, Vol. 1, May 1993, 772-777.

[41] W.M. Wells, III, "Visual Estimation of 3-D Line Segments from Motion-A Mobile Robot Vision System", *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 6, December 1989, 820-825.

[42] R.A. Brooks, "Vision and Spatial Modeling For Mobile Robots", *1989 International Workshop on Industrial Applications of Machine Intelligence and Vision*, April 1989, 9-11.

[43] A.Y. Yang, W. Hong, Y. Ma, "Structure and Pose from Single Images of Symmetric Objects with Applications to Robot Navigation", *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA '03*, Vol. 1, September 2003, 1013-1020.

[44] A.K. Das, R. Fierro, V. Kumar, B. Southhall, J. Spletzer, C.J. Taylor, "Real-Time Vision-Based Control of a Nonholonomic Mobile Robot", *Proceedings of the 2001 IEEE International Conference on Robotics and Automation, ICRA '01*, Vol. 2, May 2001, 1714-1719.

[45] Z. Hai-bo, Y. Kui, L. Jin-dong, "A Fast and Robust Vision System for Autonomous Mobile Robots", *Proceedings of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, Vol. 1, October 2003, 60-65.

[46] Y. Ma, J. Kosecka, S.S. Sastry, "Vision Guided Navigation for a Nonholonomic Mobile Robot", *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 3, June 1999, 521-536.

[47] S.Y. Chen, Y.F. Li, "Automatic Sensor Placement for Model-Based Robot Vision", *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, Vol. 34, No. 1, February 2004, 393-408.

[48] J.R. Spletzer, C.J. Taylor, "Sensor Planning and Control in a Dynamic Environment", *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA '02*, Vol. 1, May 2002, 676-681.

[49] K. Han, M. Veloso, "Reactive Visual Control of Multiple Non-Holonomic Robotic Agents", *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Vol. 4, May 1998, 3510-3515.

[50] J. Spletzer, C.J. Taylor, "A Framework for Sensor Planning and Control with Applications to Vision Guided Multi-robot Systems", *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001*, Vol. 1, December 2001, I-378-I-383.

[51] T.C. Lee; C.Y. Tsai; K.T. Song, "Fast parking control of mobile robots: a motion planning approach with experimental validation", *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 5, September 2004, 661-676.

[52] C. Samson, K. Ait-Abderrahim, "Feedback Stabilization of a Nonholonomic Wheeled Mobile Robot", *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems, IROS '91*, Vol. 3, November 1991, 1242-1247.

[53] C. Samson, K. Ait-Abderrahim, "Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, April 1991, 1136-1141.

[54] P. Morin, C. Samson, "Practical stabilization of a class of nonlinear systems. Application to chain systems and mobile robots." , *Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 3, December 2000, 2989-2994.

[55] J. Shen, D.A. Schneider, A.M. Bloch, "Controllability and Motion Planning of Multibody Systems with Nonholonomic Constraints", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Vol. 5, December 2003, 4369-4374.

[56] C. Samson, "Control of Chained Systems Application to Path Following and Time-Varying Point-Stabilization of Mobile Robots" *IEEE Transactions on Automatic Control*, Vol. 40, No. 1, January 1995, 64-77.

[57] G. Walsh, D. Tilbury, S.S. Sastry, R. Murray, J.P. Laumond, "Stabilization of Trajectories for Systems with Nonholonomic Constraints" *IEEE Transactions on Automatic Control*, Vol. 39, No. 1, January 1994, 216-222.

[58] C. Samson, "Trajectory tracking for non-holonomic vehicles: overview and case study", *Proceedings of the Fourth International Workshop on Robot Motion and Control, RoMoCo'04*, June 2004, 139-153.

[59] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, O. Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles" *Phys. Rev. Lett.*, Vol. 75, No. 6, August 1995, 1226-1229.