COORDINATION OF SYSTEMS OF MOBILE ROBOTS WITH SOCIAL POTENTIAL FUNCTIONS

by

RICHARD PATRICK SAMPLES

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical
and Computer Engineering
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2009

Abstract

The research in this dissertation is concerned with developing a method to produce flocking behavior in a swarm of mobile robots. The dissertation itself is divided into three articles. The research in the first article is a proof-of-concept that demonstrates that one can create a swarm of mobile robots that engage in flocking behavior by the use of a position tracking controller in concert with a method for defining the desired trajectory for each object. The research in the second article develops a method for implementing both attractive and repulsive artificial potential functions with a position tracking controller. The research in the third article builds on that in the first two articles in order to construct a large swarm of mobile robots that engages in flocking behavior using a social potential function that is contructed from the attractive and repulsive functions of the second article. Each article contains both control-theoretic analysis of robot behavior and demonstrations of the behavior using Matlab simulations. The overall result is a successful, yet relatively simple, method for creating a swarm of mobile robots in which each robot has a degree of freedom of action.

List of Symbols

A_d Convergence region for attractive artificial potential function

A_{free} Free action region

APF Artificial potential function

B_d Convergence region for repulsive artificial potential function

BSRT Behavioral state residency time

$f_3^{i,k}$ Component of repulsive artificial potential function

$F_3^{i,k}$ Component of additive repulsive artificial potential function

$g^i\left(r^{i,k}\right)$ Social potential function

$g_3^{i,k}$ Component of repulsive artificial potential function

g_{att}^i Attractive component of social potential function

g_{free}^i Free action component of social potential function

g_{rep}^i Repulsive component of social potential function

K_1 Controller constant

K_2 Controller constant

K_3 Controller constant

K_{random} Controller constant

l_a Radius of influence for attraction

l_r Radius of influence for repulsion

M_{att}^i Size of attractive subswarm for robot i

M_{rep}^i Size of repulsive subswarm for robot i

M_{sub} Size of a subswarm

$N_{1,sign}$ Random variable

$N_{2,sign}$ Random variable

$N_{1,var}$ Random variable

$N_{2,var}$ Random variable

S^M Swarm with M members

S^i_{att} Attractive subswarm for robot i

S^i_{rep} Repulsive subswarm for robot i

r^i Distance from robot i to the desired position

r^i_d Distance from robot i to the desired position

$r^{i,k}$ Distance from robot i to the reference position k

$\tilde{r}^{i,k}$ Relative distance error

$\dot{r}^{i,k}$ Relative speed between robot i and the reference position k

$\left(\dot{r}^{i,k}\right)_d$ Desired relative speed between robot i and the reference position k

r_{att} Radius of influence for attraction

r_{conv} Radius of convergence

r_{rep} Radius of influence for repulsion

r_{sep} Maximum distance between members of a subswarm

U_{att} Attractive artificial potential function

∇U_{att} Gradient of attractive artificial potential function

U_{rep} Attractive artificial potential function

∇U_{rep} Gradient of attractive artificial potential function

V Lyapunov function

$\dot{V}$ Time derivative of Lyapunov function

x^i_1 x-axis component of position of robot i

x^i_2 y-axis component of position of robot i

x^i Position of robot i

x^i_d Desired position of robot i

$x^i_{d,free}$ Desired position of robot i for free action

$x^i_{d,general}$ Generalized desired position of robot i

$x^i_{d(rep)}$ Desired position of robot i for repulsive behavior

x^k Position of robot k

$\bar{x}$ Center of the swarm

$\bar{x}^i_{att}$ Center of attractive subswarm for robot i

$\bar{x}^i_{rep}$ Center of repulsive subswarm for robot i

$\bar{x}_{sub}$ Center of a subswarm

v^i Translational speed control input for robot i

ε Tolerance for desired relative distance

θ^i Heading of robot i

ξ^i Pose of robot i

ϕ^i Relative angle from robot i to the desired position

$\phi^{i,k}$ Relative angle from robot i to the reference position k

σ_1 Switching variable

σ_2 Switching variable

σ_3 Switching variable

σ_4 Switching variable

ω^i Rotational speed control input for robot i

Ψ^i Direction from desired position to robot i

$\Psi^{i,k}$ Direction from reference position k to robot i

Ω^i_{att} Non-attraction region for robot i

Ω^i_{free} Free action region for robot i

Ω^i_{rep} Non-repulsion region for robot i

Acknowledgments

First and foremost, I must thank Robert P. Leland, Ph.D., for being my Ph.D. advisor at The University of Alabama from August 2002 to May 2009. I am especially grateful to him for remaining my Ph.D. advisor after he moved from The University of Alabama to Oral Roberts University in Tulsa, Oklahoma. Being a Ph.D. advisor is hard work, and being a long-distance Ph.D. advisor is going above and beyond the call of duty.

I would also like to thank the rest of my Ph.D. committee: Tim Haskew, Ph.D., Tan-Yu Lee, Ph.D., Keith Williams, Ph.D., and Kenneth Ricks, Ph.D. Your assistance is greatly appreciated.

Special thanks are due to Zhijian Wu, Ph.D., for helping me prepare for the Real Analysis section of my written qualifying exam. Special thanks are also due to Stan Jones, Ph.D., Joseph Neggers, Ph.D., and Robert Scharstein, Ph.D., for grading sections of my written qualifying exam.

Thanks to John Baker, Ph.D., who was the UA campus director for the Alabama Space Grant Consortium during my time at UA, for helping me get a Alabama Space Grant Consortium Fellowship for 2007-2008 and 2008-2009.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of Research Project

This dissertation is concerned with the problem of designing a method for producing flocking behavior in a swarm of differential-drive mobile robots. In flocking, each robot in the swarm keeps itself within a range of distance from every other robot. That is, it keeps itself inside a maximum range and outside of a minimum distance from the other robots. We use differential-drive robots because they are a kind of mobile robot that is commonly used in robotics research and because they have relatively simple kinematics. Also, there are many control laws for this kind of robot in the control and robotics literature. Our main goal in our research project is to demonstrate that we can achieve swarm cohesiveness by causing all robots stay relatively close to each other, prevent collisions between robots and allow each robot a degree of freedom of action. This dissertation is an interdisciplinary work in the sense that it deals with problems in the fields of robotics, multiagent systems, control theory and switched systems theory.

1.2 Robotics

Robotics can be divided into articulated robotics and mobile robotics. *Articulated robotics* deals with robot arms (or robot manipulators), such as those that are used in automated manufacturing systems. This field uses a large amount of advanced mathematics, physics and control theory [1][2].

Mobile robotics deals with robots that are equipped with wheels, legs, or some other device that gives them the ability to move on their own. These mobile robots can be ground vehicles, aircraft or marine vessels. The field makes use of knowledge from a wide range of disciplines, including mathematics, physics, computer science, aritificial intelligence (AI) and control theory [3][4][5][6].

Robot motion planning is concerned with determining how a robot should move in order to accomplish a given task properly. In articulated robotics, the classic motion planning problem involves how to make the robot arm move properly in order to accomplish a manufacturing task. In mobile robotics, the motion planning problem involves how to make a mobile robot move through a workspace in the proper way. If the robot is operating as part of a robot swarm, then the motion planning problem becomes that of moving the individual robot properly and of coordinating each robots actions with those of the other robots in the swarm [7][8].

There are three major paradigms in robot motion planning. These three paradigms differ in how they sense information about the robot's operating environment, how they plan the robot's actions and how they actually command the robot to act. The *hierarchical paradigm* attempts to construct as complete a picture of the robot's operating environment as possible before planning the robot's action. It requires a very accurate map of that environment, and it does not work well when there are unknown aspects of that environment. After the information gathering step has been completed, the next step is to plan out the robot's action completely in regard to accomplished the desired task. The planning step produces a list of separate robot actions that will likely cause the robot to accomplish its task. The final step

is to have the robot perform the first, and only the first, action on the list of actions. Once this action is complete, the paradigm has the robot repeat the first two steps of sensing and planning completely. The hierarchical paradigm is very computationally expensive due to this need to re-perform the sensing and planning stages a large number of times, especially since these two steps are already so computationally expensive themselves [9].

The *reactive paradigm* is based on behaviors. A *behavior* is a "direct mapping of sensory inputs to a pattern of motor actions that are then used to achieve a task" [10]. Stated differently, a behavior defines what the robot should do when it sensing that a given thing is true in its environment. For example, if a robot senses that it is too close to an obstacle using an infrared range-finder, then it moves away from the obstacle at a velocity determined by a mathematical function. This behavior is called *repulsion*. This behaviors are also called *basis behaviors* because they are used as the basis of more complex behaviors, which are called *emergent behaviors*. The behaviors are often combined in a subsumption architecture that ranks the various behaviors from highest to lowest priority. The robot works its way through the list of priorities from highest to lowest [10].

The reactive paradigm uses only sensing and action, and it does not use a planning step. Thus, it is much less computationally expensive that the hierarchical paradigm. One of the major disadvantages of the reactive paradigm is that it often produces unexpected, unpredictable and unwanted emergent behaviors. This is due to the fact that the emergent behaviors cannot be fully predicted, demonstrated or otherwise analyzed using the methods of mathematical and logical proof.

The *hybrid deliberative-reactive paradigm* combines elements of the reactive paradigm with planning and deliberation methods in order to extend the reactive paradigm. Its use of planning is not as extensive as that found in the hierarchical paradigm. The most common version of the hybrid paradigm uses a method that has the robot perform the necessary planning and, then, the robot executes the entire plan (e.g., the entire list of actions) until it is complete. Only when the plan has been completed will the robot perform the planning

step again. The intent of the hybrid paradigm is to use the best elements of the hierarchical and reactive paradigms in order to achieve a useful compromise among the advantages and disadvantages of both of those paradigms [11].

Robot motion control is concerned with actually driving a mobile robot from one position to another. Usually, this involves either having the robot move to a single goal position or having the robot tracking a desired trajectory. In either case, the task is to design a controller for the robot that will drive the robot at the appropriate velocity (i.e., speed and heading) such that it successfully arrives at the goal position or correctly tracks the desired trajectory. While robot motion control is primarily an area of control theory, it is also an area of computer science and computer engineering since the robot is almost always controlled by a microcontroller. The microcontroller is an embedded system, which indicates that embedded systems theory is also a part of the overall theoretical structure of mobile robotics.

The theory of control laws for mobile robots is an area of control theory with a very large body of literature. The mobile robot is a nonlinear system. Thus, there has been much research on designing nonlinear control laws for mobile robots, especially those of the unicycle-type that are relatively less complex in terms of their kinematics. The control laws are generally designed using either Lyapunov control theory or sliding mode control theory [12][13][14][15]. There are also some linear controller designs in the literature [16].

1.3 Multi-Agent Systems and Robot Swarms

An agent is an entity that has some degree of autonomy. That is, an agent has the ability to act on its own to some degree. Multiagent systems are simply systems of agents. A swarm is a large system of agents that are all working on a single task [17]. These systems have a degree of autonomy in the same way that the individual agent does.

These swarms of robots or autonomous vehicles have significant advantages over con-

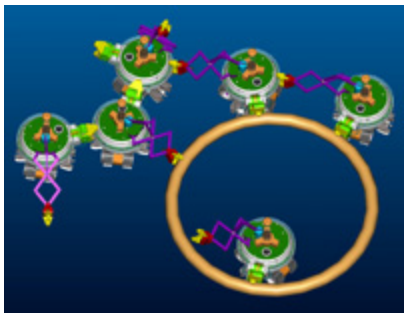

Image Courtesy of Swarm-Bots
<www.swarm-bots.com>

Figure 1.1: A Robot Swarm.

ventional kinds of teams. In conventional teams, there is usually a very complex central controller that uses deliberative control strategies to control the behavior of the team. Thus, the controller is complex and the system is not very flexible or scalable. If one robot fails, often the system will no longer be able to function properly. On the contrary, swarms are based on groups of simple, reactive agents. The agents can perform a small set of simple behaviors and can sense and react to their environment and each other. The interaction of the individual robots allows more complex behaviors to emerge. Thereby the swarm can perform complex tasks while using very simple control strategies for each agent and the group as a whole.

The macroscopic modeling of multiagent systems is an major area of research in robotics [18]. Our research uses the concept of a *basis behavior* [18] in order to develop the concept of a behavior state, of behavioral state residency time (BSRT), and of BSRT analysis. This BSRT analysis provides us with a metric for determining the amount of freedom of action that the robots in our robot swarm have.

A mobile robot is one kind of agent. By putting several of these robots into a system, one creates a multi-robot system or a robot swarm. The research for this dissertation concentrates on an investigation of how to create a form or loose formation behavior known as *flocking* in a robot swarm. V. Gazi and K. M. Passino have investigated the case of rigid formation behavior in a kind of robot swarm [20][21][22][23]. Their research treats the

robots as omnidirectional particles. These and other researchers have extended this line of research by considering noisy environments, foraging tasks, the use of sliding mode control for motion control and so forth [24][25][26][27][28][29][30][32].

A good overview of the work in multi-robotic swarms can be found in [32][33][34]. Each work contains papers on all the various areas of research within the field of multi-robotic systems. This field is closely related to the fields of multi-agent systems, distributed robotics, and mobile robots. The system of interest in current research is usually a system of some number of small mobile robots.

Localization includes all the tasks and behaviors associated with knowing the location of each robot in the swarm. This field also includes the areas of *mapping* and *navigation*. Localization has been studied extensively [35]. Research on localization concentrates on two major tasks. First, there is the task of localizing an individual robot using an a priori map of the environment. Second, there is the task of localizing a robot while building a map of the environment at the same time. Some research has been done on localization using multiple robots, which is called *cooperative localization*. Statistical and probabilistic techniques are the most common tools used in all of the methods researched. The use of landmarks and of methods based on the cooperation between various robots has been investigated [36]. The major result of this research that concerns us here is that effective strategies for knowing the location of all the robots in the swarm exist and are practical. If this were not so, then the use of social potential functions would be impractical because there would be no way to accurately measure the distance between the robots in the swarm.

Distributed surveillance involves using multiple mobile robots in order to accomplish a surveillance task [37]. An example is the use of unmanned aerial vehicles (UAVs) by the Army in order to gather intelligence information. Another example is the use of unmanned ground vehicles (UGVs) for the same task.

Distributed Manipulation involves using multiple mobile robots in order to manipulate an object [38]. Sometimes, the robots need to have special manipulation equipment.

Sometimes, the robots do not [39]. In this latter case, the robots can move or otherwise manipulate the object by pushing it or capturing it between two robots.

Coordination and Formations includes all the strategies used by the swarm in order to make the individual robots work together in order to perform a certain task [40][41]. The coordination strategy must maintain the *cohesion* of the swarm and prevent collisions between the swarm members. It must also allow the robots to act in such a way that they can perform the actions that they need to perform in order to accomplish their assigned task. The two strategies for coordination are the centralized approach and the distributed approach[42]. In the centralized approach, a central planner plans and controls the actions of all the robots. In the distributed approach, each agent is autonomous and controls its own actions. Some research has been done on economy-based architectures that use negotiation and other concepts from economics to coordinate a robot swarm.

The area of *human/robot interaction* deals with how human beings interact with multi-robot systems [43]. Other areas include *design and learning*, *sensor and hardware issues*, *planning and task allocation*, *large-scale robot teams*, and *communication constraint and networks* [32][33][34].

1.4 Switched Systems Theory

A *switched system* is a system in which some parameter of the system is changed in some way during the operation of the system. The changing of the parameter is called a switching event, and the parameter itself can be a controller constant, a control input, a constant in the dynamic model or many other things. In our multi-robotic system, we use a switched system in order get the desired flocking behavior. Liberzon [44] gives a very good overview of the control-theoretic aspects of switched systems.

1.5 Innovative Aspects of this Research

At this point, we delineate clearly what aspects of this research are new and innovative and, thus, constitute the original research of this project. Firstly, we will list the aspects of the research that we derive from previous research in the fields of control theory and robotics, to wit: (1) the nonlinear position tracking controller for the robot, (2) the quadratic attractive artificial potential function (APF), (3) the repulsive APF, (4) the concept of a social potential function for a robot in a robot swarm, (5) the reactive paradigm in robotics and (6) the fundamental concepts involved in the analysis of the behavior of robot swarms, including the use of Lyapunov control theory and sliding mode control theory to prove convergence to a specified region.

The innovative aspects of our research can be summed up in the following list: (1) using the position tracking controller in order implement artificial potential functions (APFs), (2) using a social potential function whose components are standard attractive and repulsive APFs, (3) the control synthesis method of combining the elements of the controller, SPF and switching rule in order to cause the robots to engage in flocking (as opposed to forming), (4) using differential-drive mobile robots with realistic kinematics (i.e., they have motion constraints) instead of omnidirectional robots that are kinematically modeled as omnidirectional particles, (5) the control analysis concepts of of behavioral state, behavioral state residency time (BSRT) and BSRT analysis as a metric for freedom of action and (6) the overall control analysis techniques for the analysis of convergence, collision avoidance and freedom of action in swarm of differential drive mobile robots that are engaging in flocking (i.e., for the kind of swarm that we define and analyze in our research).

The analysis techniques in item (6) include (a) the use of concentric circle graphs that show the trajectories of the other robots in the swarm relative to one, specified robot, (b) the use of a graph of the trajectories of all robots relative to the swarm center to show convergence within a certain radius of the center of the swarm and (c) the use of a table of maximum and minimum distances between each pair of robots in the swarm to demonstrate

collision avoidance and convergence characteristics. The BSRT analysis in item (5) refers to finding the percentage of time that the robot is in a given behavioral state.

Finally, this dissertation is a continuation of the research performed for the author's M.S. thesis [46].

1.6 Organization of Dissertation

This article-style dissertation has an introduction, a main body and a conclusion. Here in the introduction, the overall goals and methods of the research project are given after a brief overview of the field of multi-robotic systems (or robot swarms). The main body of this dissertation consists of three articles. The first article is a proof-of-concept article that lays the groundwork for the second and third articles. The second article contains the theory that will be used in the third article to create a large swarm of robots. The third article covers the analysis of a large swarm of robots and reports the results of various simulations that serve to verify the theory. The conclusion reports the overall results of the research project and suggests avenues for further research in the field based on those results.

The first article is a proof-of-concept article that concentrates on a simple, two-robot swarm of differential-drive wheeled mobile robots. The goal of this article is to demonstrate that we can apply the techniques of control theory to the problem of analyzing the behavior of a team of mobile robots and of designing appropriate control laws for each individual robot that produce the desired behavior of the team. We show that we can prove that we can achieve cohesiveness, prevent collisions and allow each robot freedom of action.

We make the project as realistic as possible by using a position tracking controller that is like the kind of controller that a real robot would use. We introduce the concept of the behavior state of the robot and of behavioral state residency time (BSRT) in order to obtain a metric for measuring the amount of freedom of action that each individual robot has.

The rather primitive method of the first article works very well for the two-robot swarm,

but they cannot be scaled up to be used on a larger swarm. Therefore, we must develop a method for implementing artificial potential functions that is practical for large swarms of robots, which is the goal of the second article. The second article uses a position tracking controller to implement both attractive and repulsive functions.

The third builds on the first and second articles by applying the theory of the second article in order to create a large swarm of robots. The two-robot swarm of the first article is scaled up to a swarm of a much larger size and the analysis and simulations of the first article are re-performed. The analysis shows that there limitations to the method, such that the method suffers from a problem that we call interstate chattering that prevents the robots from achieving freedom of action under certain conditions. We establish some conditions for the robot's workspace (i.e., operational environment) that are necessary if the robot is to have freedom of action.

The conclusion of the dissertation presents the results of the overall research project. It reviews the results of the three articles in the context of the whole research project and in the context of related research. Finally, it suggests avenues of further research in regard to this research project and to similar ones that are documented in the technical literature of robotics and control theory.

This dissertation is written in the style for articles that appear in the journals of the Institute of Electrical and Electronics Engineers (IEEE) or "IEEE style." Each chapter, including the Introduction and the Conclusion, is written as an IEEE-style article. Therefore, each chapter has its own bibliography. The Matlab code that was used for the simulations in the three articles is placed in three appendices in the back of the dissertation. The code is very long, which means that putting it after the articles would unnecessarily interrupt the flow of the text.

Bibliography

[1] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot Manipulator Control: Theory and Practice*, Second Edition, New York: Marcel Dekker, Inc, 2004.

[2] M. Xie. *Fundamentals of Robotics: Linking Perception to Action*, New Jersey: World Scientific, 2003.

[3] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000.

[4] Ronald C. Arkin, *Behavior-Based Robotics*, Cambridge, Massachusetts: The MIT Press, 1998.

[5] R. Siegwart and I. R. Nourbakh, *Introduction to Autonomous Mobile Robots*, Cambridge Massachusetts: The MIT Press, 2004.

[6] J. L. Jones and A. M. Flynn, *Mobile Robots: Inspiration to Implementation*, Wellesley, Massachusetts: A. K. Peters, 1993.

[7] J.-C. Latombe, *Robot Motion Planning*, Boston: Kluwer Academic Publishers, 1991.

[8] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. 2005, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, Cambridge, Massachusetts: The MIT Press.

[9] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, pp. 42-44.

[10] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, p. 108.

[11] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, pp. 108-110.

[12] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, pp. 259-261.

[13] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed Loop Steering of Unicycle-like Vehicles via Lyapunov Techniques," *IEEE Robotics and Automation Magazine*, March 1995.

[14] B. d'Andrea-Novel, G. Bastin, and G. Campion, "Dynamic Feedback Linearization of Nonholonomic Wheeled Mobile Robots," *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*. Nice, France, May 1992.

[15] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.J. You and S.-R. Oh, "A Stable Target-Tracking Control for Unicycle Mobile Robots," *Proceedings of the 2000 IEEE/RS International Conference on Intelligent Robots and Systems*, 1822-1827.

[16] J. M. Yang and J.-H. Kim, "Sliding Mode Control for Trajectory Tracking of Nonholonomic Wheeled Mobile Robots," *IEEE Transactions on Robotics and Automation, Vol. 15, No. 3, June 1999.*

[17] R. Siegwart and I. R. Nourbakh, *Introduction to Autonomous Mobile Robots*, Cambridge Massachusetts: The MIT Press, 2004, pp. 82-88.

[18] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, pp. 293-294.

[19] K. Lerman and A. Galstyan, "A General Methodology for Mathematical Analysis of Multi-Agent Systems," *USC Information Sciences Technical Report ISI-TR-529*, 2001.

[20] K. Lerman, "Design and Mathematical Analysis of Agent-based Systems," *Lecture Notes in Artificial Intelligence (LNAI) 1871*, pages 220 ff, Springer-Verlag, Berlin Heidelberg, 2001.

[21] V. Gazi and K. M. Passino, "Stability Analysis of Swarms," *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, April 2003, 692-697.

[22] V. Gazi and K. M. Passino, "A Class of Attraction/Repulsion Functions for Stable Swarm Aggregations," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, 2842-2847.

[23] V. Gazi and K. M. Passino, "Stability Analysis of Social Foraging Swarms: Combined Effects of Attractant/Repellent Profiles," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, 2848-2853.

[24] V. Gazi and K. M. Passino, "Stability of Social Foraging Swarms," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 34, No. 1, February 2004*, 539-557.

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

[26] Y. Liu, K. M. Passino, and M. Polycarpou, "Stability Analysis of One-Dimensional Asynchrounous Swarms," *IEEE Transactions on Automatic Control, Vol. 48, No. 10, October 2003*, 1848-1854.

[27] Y. Liu, K. M. Passino, and M. Polycarpou, "Stability Analysis of M-Dimensional Asynchronous Swarms With a Fixed Communication Topology," *IEEE Transactions on Automatic Control, Vol. 48, No. 1, January 2003*, 76-95.

[28] Y. Liu and K. M. Passino, "Stable Social Foraging Swarms in a Noisy Environment," *IEEE Transactions on Automatic Control*, Vol. 49, No. 1, January 2004.

[29] V. Gazi, "Swarm Aggregation Using Artificial Potentials and Sliding-Mode Control," *IEEE Transactions on Robotics,* Vol. 21, No.l 6, December 2005, 1208-1214.

[30] V. Gazi and R. Ordonez, "Target Tracking Using Artificial Potentials and Sliding Mode Control," *Proceedings of the 2004 American Control Conference,* Boston, Massachusetts, June 30 - July 2, 2004.

[31] J. Yao, R. Ordonez, and V. Gazi, . "Swarm Tracking Using Artificial Potentials and Sliding Mode Control," *Proceedings of the 45th IEEE Conference on Decision and Control*, Manchester Grand Hyatt Hotel, San Diego, CA, USA, December 13-15, 2006, 4670-4675.

[32] K. M. Passino, *Biomimicry for Optimization, Control, and Automation*, London: Springer-Verlag, 2005.

[33] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002.

[34] A. C. Schultz, L. E. Parker and F. E. Schneider (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata Volume II: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2003.

[35] A. C. Schultz, L. E. Parker and F. E. Schneider (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata Volume III: Proceedings from the 2005 International Workshop on Multi-Robot Systems,* New York: Springer, 2005.

[36] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 45-46.

[37] E. J. Barth, "A Dynamic Programming Approach to Robotic Swarm Navigation using Relay Markers," *Proceedings of the American Control Conference*, Denver, Colorado, June 4-6, 2003, 5264-5269.

[38] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 65-80.

[39] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 83-100.

[40] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 93-94.

[41] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 103-130.

[42] A. C. Schultz, L. E. Parker and F. E. Schneider (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata Volume II: Proceedings from the 2003 International Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2003, pp. 53-98.

[43] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 104-105.

[44] A. C. Schultz and L. E. Parker (eds.), *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 Naval Research Laboratory Workshop on Multi-Robot Systems*, Boston: Kluwer Academic Publishers, 2002, pp. 133-157.

[45] D. Liberzon, *Switching in Systems and Control*, Boston: Birkhauser, 2003.

[46] R. P. Samples, *Analysis and Simulation of a Multi-Robotic System with a Piecewise-Constant Social Potential Function,* M. S. thesis, The Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, Alabama, 2004.

Chapter 2

Investigation of Flocking in a Two-Robot Swarm of Wheeled Mobile Robots

2.1 Background

Robot swarms (which are also called *multi-robotic systems*, *multi-robot teams* and *multi-agent systems*) are groups of mobile robots that operate together. The robots in a swarm need a strategy that determines how a robot acts with regard to the other robots in the swarm. Such a strategy determines, among other things, how close to each other the robots should be and how they should cooperate in order to achieve a collective goal, such as searching or foraging.

Gazi and Passino developed a method of stability analysis for swarms that uses a specific class of artificial potential functions and Lyapunov control theory. Their method uses a social potential function that defines a desired velocity for robot i that is a function of the distance between robot i and another robot j. The overall desired velocity for robot i is found by summing up the individual contributions from each pair of robots. The function is zero at a single point, which corresponds to a single desired distance of separation for a given pair of robots i and j. This fact causes the members of the swarm to organize

themselves into a rigid formation, thus achieving *forming* behavior in the swarm. Gazi and Passino use Lyapunov control theory to prove that each individual member of the swarm converges to a hyperball in finite time, which demonstrates that the swarm achieves a stable formation. See [3][5][6][7][8][9] [10] for further details.

The method developed in this paper allows each robot i in the swarm freedom of action. It does this by controlling the action of robot i such that it remains between both a maximum and a minimum distance from the other robot k in the swarm. This leads to *flocking* [1][2] behavior instead of *forming* behavior. The principle on which the method is based is that the social potential function must have a dead zone in which the force placed on robot i by the function is zero. The function will cause robot i to converge to the free action region, then it will allow the robot to move freely while it stays within that region. With this freedom of action, robot i can wander, pick up objects, home, and engage in other *basis behaviors* [1][12] that are necessary in order to perform a given task. The concept of *behavioral state residency time* is introduced in order to establish a measure of the degree of freedom of action that a member of the swarm has. Finally, the results of simulations that demonstrate the method are given.

2.2 Control Problem Statement

We want to develop a coordination strategy for a swarm of differential-drive mobile robots that will prevent collisions between a robot and another robot, cause the robots to converge within a hyperball, and allow each robot sufficient freedom of action in order to accomplish a useful task. We will simplify the problem by assuming that there are no obstacles in the workspace. The fundamental problem that we will investigate is the problem of maintaining the proper spacing between the agents. If the agents get too close together, they will collide. If the agents get too far apart, they will effectively cease to be a swarm. Thus, we want a control law that will cause the robots to repel from each other when they are too close

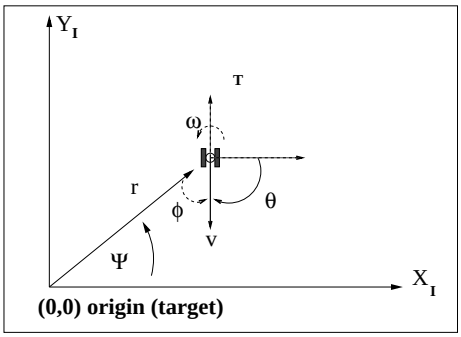

Figure 2.1: Tracking Coordinates

together, to attract toward each other when they are too far apart, and to wander freely when they are neither too close together nor too far apart.

2.3 Coordinate System

Our coordinate system uses inertial coordinates, tracking coordinates and relative coordinates. The inertial coordinates define the absolute position of the robot in the workspace. The tracking coordinates define the robot's position relative to a desired position, and, thus, they are used by the robot's controller in order to define control errors. The relative coordinates define the robot's position relative to other robots and obstacles in the workspace.

The *pose* of robot i in the inertial reference frame and Cartesian coordinates is represented by $\xi^i = \begin{bmatrix} x^i & \theta^i \end{bmatrix}^T$, where $x^i = \begin{bmatrix} x_1^i & x_2^i \end{bmatrix}^T$ is the position, x_1^i and x_2^i are the x-axis and y-axis components of the positions respectively, and the angle θ^i is the heading that is measured counterclockwise from the x-axis of the inertial reference frame to the forward direction of the robot.

The *tracking coordinates* $\left(r^i, \phi^i\right)$ are defined by the coordinate transformation [15]

$$\begin{aligned} r^i &= \left\|x^i - x_d^i\right\| = \sqrt{\left(x_1^i - x_{d1}^i\right)^2 + \left(x_2^i - x_{d2}^i\right)^2} \\ \Psi^i &= \arctan\left(\frac{x_2^i - x_{d2}^i}{x_1^i - x_{d1}^i}\right) \\ \phi^i &= 180^o + \theta^i - \Psi^i \end{aligned} \tag{2.1}$$

where r is the straight line distance from the robot to the *target* (i.e., another robot or an obstacle) and ϕ is the angle between the forward direction of the robot and the direction to the target. We define the *position error* for robot i (i.e., the the difference between the actual and desired positions for robot i) as $\tilde{x}^i = x^i - x_d^i$ in inertial coordinates and as $\tilde{r}^i = r^i - r_d^i$ in tracking coordinates.

Similarly, the *relative coordinates* $\left(r^{i,k}, \phi^{i,k}\right)$ are defined by the coordinate transformation

$$\begin{aligned} r^{i,k} &= \left\|x^i - x^k\right\| = \sqrt{\left(x_1^i - x_1^k\right)^2 + \left(x_2^i - x_2^k\right)^2} \\ \Psi^{i,k} &= \arctan\left(\frac{x_2^i - x_2^k}{x_1^i - x_1^k}\right) \\ \phi^{i,k} &= 180^o + \theta^i - \Psi^{i,k} \end{aligned} \tag{2.2}$$

where x^k is the *reference position* (i.e., the position of an obstacle or another robot), $r^{i,k}$ is the *separation* (or *relative displacement*) and $\phi^{i,k}$ is the *relative angle*. The *separation error* (i.e., the difference between the actual and desired distances of separation between robot i and robot k) is $\tilde{r}^{i,k} = r^{i,k} - r_d^{i,k}$. The *relative position* is $x^{i,k} = x^i - x^k$, which is a vector that points from robot k to robot i. Also, the *desired relative position* is $x_d^{i,k} = x_d^i - x^k$.

In our two-robot swarm, robot i uses the position of robot k (i.e., the other robot) as its reference position. The desired position x_d^i is defined by (2.9) in such a way that the vectors $\tilde{x}^i$, $x^{i,k}$ and $x_d^{i,k}$ are *always* collinear (See Figure 2.2). Since, r^i, $r^{i,k}$ and $r_d^{i,k}$ respectively are the magnitudes of these vectors, we have the following relationship between the tracking and relative coordinates.

$$r^i = \left\|x^i - x_d^i\right\| = \left|r^{i,k} - r_d^{i,k}\right| = \left|\tilde{r}^{i,k}\right| \tag{2.3}$$

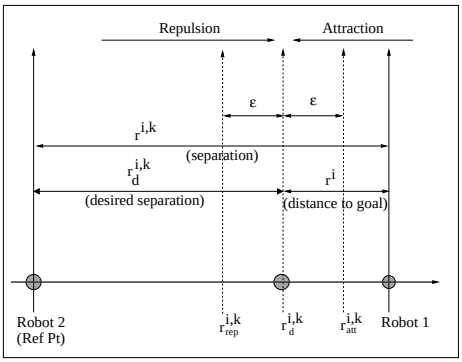

Figure 2.2: Tracking Coordinates and Relative Coordinates

We want the robot to converge to a *free action zone* A_{free} that is centered around $r_d^{i,k}$, which is defined by

$$A_{free} = \left\{ x^i : \; r_d^{i,k} - \varepsilon \leq r^{i,k} \leq r_d^{i,k} + \varepsilon \right\}$$

2.4 Social Potential Function and Swarm Kinematics

For robot i, we define a social potential function (SPF) $g^i(r^{i,k}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ that assigns a desired velocity to robot i (i.e., $\dot{x}^i = g^i(r^{i,k})$) according to

$$g^i(r^{i,k}) = \begin{cases} g^i_{rep}(r^{i,k}) & r^{i,k} \leq l_r \\ g^i_{free}(r^{i,k}) & l_r < r^{i,k} < l_a \\ g^i_{att}(r^{i,k}) & r^{i,k} \geq l_a \end{cases} \tag{2.4}$$

where $i,k = 1,\ldots,M$, M is the number of robots in the swarm, $g^i_{rep}(r^{i,k})$ is repulsive component, $g^i_{att}(r^{i,k})$ is the attractive component, $g^i_{free}(r^{i,k})$ is the free action component, l_a is the range of attraction, and l_r is the range of repulsion [3]. This SPF produces the following relative motion between robots i and k.

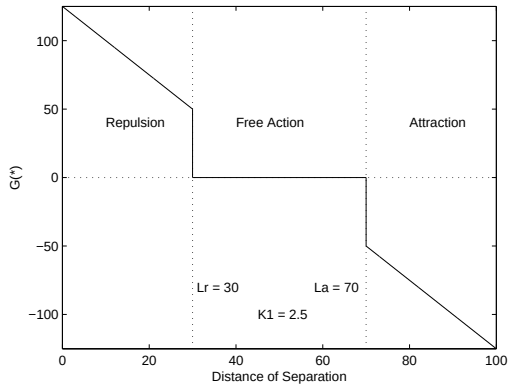

Figure 2.3: Social Potential Function g(·)

$$\dot{r}^{i,k} = \begin{cases} \dot{r}^{i,k} > 0: & r^{i,k} \leq l_r \\ \dot{r}^{i,k} = 0: & l_r < r^{i,k} < l_a \\ \dot{r}^{i,k} < 0: & r^{i,k} \geq l_a \end{cases}$$

A negative velocity indicates that the robots move toward each other (i.e., attract), while a positive velocity indicates that they move away from each other (i.e., repel). Figure (2.3) shows an example of this social potential function with $l_r = 30$, $l_a = 70$, $g^i_{rep}(r^{i,k}) = K_1 r^{i,k}$, $g^i_{att}(r^{i,k}) = -K_1 r^{i,k}$, and $K_1 = 2.5$.

Robot i uses a position tracking controller that drives it to the desired position x^i_d. Attraction and repulsion are implemented by defining x^i_d in such a way that robot i moves either closer to robot k or further away from it.

2.5 Differential-Drive Robot Kinematics

The kinematics of a differential-drive mobile robot i with coordinates in the inertial frame are given by [2]

$$\dot{\xi}^i = \begin{bmatrix} \dot{x}_1^i \\ \dot{x}_2^i \\ \dot{\theta}^i \end{bmatrix} = \begin{bmatrix} v^i \cos\theta^i \\ v^i \sin\theta^i \\ \omega^i \end{bmatrix} \tag{2.5}$$

where ξ^i is the pose, x_1^i and x_2^i are the x-axis and y-axis components of the robot's position, θ^i is the robot's heading, v is the robot's linear velocity, ω^i is the robot's angular velocity, and the dots indicate the time-derivative of each quantity. Also, v^i and ω^i are the control inputs for the robot.

When the tracking coordinate transformation (4.1) is applied to the kinematics (4.3), we robot, we get the *tracking kinematics* [15]

$$\begin{aligned} \dot{r}^i &= -v^i \cos(\phi^i) \\ \dot{\phi}^i &= \omega^i + \tfrac{v^i}{r^i} \sin(\phi^i) \end{aligned} \tag{2.6}$$

2.6 Position Tracking Controller

Lee et. al [15] have designed a stable position tracking controller for a differential-drive mobile robot that is based on the coordinate transformation (4.1). The following lemma is a restatement of the main result of [15].

Lemma 1: [15] Consider a differential-drive robot with kinematics (4.3) that uses the tracking coordinates (4.1) in order to obtain the tracking kinematics (4.4). If $\vee t \in [0,\infty)$ with controls laws given by (4.5)

$$\begin{aligned} v^i &= K_1 \, r^i \cos(\phi^i) \\ \omega^i &= -K_1 \, sin(\phi^i) \cos(\phi^i) - K_2 \, \phi^i \end{aligned} \tag{2.7}$$

then r^i and ϕ^i are uniformly bounded. Furthermore, we have $\lim_{t\to\infty} \left\| \left(r^i, \phi^i\right)^T \right\| = 0$.

Remark 1: This implies that $r^i = \left\| x^i - x_d^i \right\| \to 0$, and, therefore, that $x^i \to x_d^i$ as $t \to \infty$.

2.7 Definition of Swarm A

In this section, we define the our robot swarm more precisely.

Definition 1: Swarm A is a two-robot swarm of differential-drive robots that uses the tracking control law (4.5), the *generalized desired position* is defined by

$$x^i_{d,general} = \left[x^i_d\right]\sigma_1 + \left[x^i_{d,free}\right]\sigma_2 \tag{2.8}$$

in which the *desired position* x^i_d is defined by

$$x^i_d = x^k + r^{i,k}_d \left[\cos\Psi^{i,k} \quad \sin\Psi^{i,k}\right]^T \tag{2.9}$$

This choice of x^i_d is made in order to drive robot i to a desired range of distance from robot k. The *desired position for free action* $x^i_{d,free}$ is determined arbitrarily, and the switching variables σ_1 and σ_2 are controlled by the switching rule

$$(\sigma_1,\sigma_2) = \begin{cases} (1,0): & \left|\tilde{r}^{i,k}\right| > \varepsilon \\ (0,1): & \left|\tilde{r}^{i,k}\right| \le \varepsilon \end{cases} \tag{2.10}$$

2.8 Convergence to Free Action Zone

The following theorem demonstrates that each robot in Swarm A will converge to their own free action zone.

Theorem 1: For Swarm A as defined in Definition 2.7, robot i will converge to the free action area

$$A_{free} = \left\{x^i : \; r^{i,k}_d - \varepsilon \le r^{i,k} \le r^{i,k}_d + \varepsilon\right\} = \left\{x^i : \; \tilde{r}^{i,k} \le |\varepsilon|\right\}$$

in which desired separation $r^{i,k}_d$ is the radius of the circle that defines the center of the free action area and ε is the tolerance that defines the width of the free action area such that the

width is 2ε.

Proof: We consider two cases: (a) $\left|\tilde{r}^{i,k}\right| \leq \varepsilon$ and (b) $\left|\tilde{r}^{i,k}\right| > \varepsilon$. For case (a), $x^i \in A_{free}$. Therefore, we have convergence to A_{free} trivially.

For case (b), $(\sigma_1, \sigma_2) = (1,0)$ in equation (2.8), which means that $x^i_{d,general} = x^i_d$. From Lemma 1, we know that $x^i \to x^i_d$ as $t \to \infty$. Therefore, $x^i \to x^i_d$. Because $x^i_d \in A_{free}$, this implies that x^i converges to the boundary layer around $S_d = \left\{x^i : r^{i,k} = r^{i,k}_d\right\}$. In other words, we make the surface S_d attractive, which causes robot i to converge to that surface. The switching rule (2.10) creates a boundary layer of width 2ε around the attractive surface. Thus, with this switching rule, we achieve convergence to the surface S_d within a tolerance ε. The boundary layer around S_d is equivalent to the free action zone A_{free}. Therefore, we have that x^i converges to A_{free}.

2.9 Convergence to a Hyperball

Theorem 2: In the two-robot swarm described in Theorem (2.8), robot i will converge to the hyperball

$$B_r(\bar{x}) = \left\{x^i : \left\|x^i - \bar{x}\right\| \leq \frac{1}{2} l_a\right\}$$

where $\bar{x} = \frac{x^i + x^k}{2}$ is the center of the two-robot swarm and l_a is the radius of attraction for each robot. Since $i = 1 \ldots 2$, the whole swarm will converge to this hyperball.

Proof: From Equation (2.3) and Theorem 1, we know that

$$\left|\tilde{r}^{i,k}\right| = \left\|x^i - x^i_d\right\| \leq \varepsilon$$

When $\tilde{r}^{i,k} = \varepsilon$, robot i is at its maximum distance from robot k in the free action zone. Thus, when robot i is in the outer half of the free action zone (i.e., $0 < \tilde{r}^{i,k} \leq \varepsilon$), we have

$$\tilde{r}^{i,k} = \left\|x^i - x^i_d\right\| \leq \varepsilon$$

Next, we use the facts that $l_a = r^{i,k}_d + \varepsilon$ and $\tilde{r}^{i,k} = r^{i,k} - r^{i,k}_d$, which we rearrange as $r^{i,k} =$

$r_d^{i,k} + \tilde{r}^{i,k}$, in order to get

$$r^{i,k} = r_d^{i,k} + \tilde{r}^{i,k} = r_d^{i,k} + \left\|x^i - x_d^i\right\| \leq r_d^{i,k} + \varepsilon = l_a$$

Therefore $r^{i,k} = \left\|x^i - x^k\right\| \leq l_a$. Next, we substitute $\bar{x} = \frac{x^i + x^k}{2}$ into $\left\|x^i - \bar{x}\right\|$ to get

$$\left\|x^i - \bar{x}\right\| = \left\|x^i - \left(\frac{x^i + x^k}{2}\right)\right\| = \frac{1}{2}\left\|x^i - x^k\right\|$$

Then, we use the fact that $\left\|x^i - x^k\right\| \leq l_a$ to get

$$\left\|x^i - \bar{x}\right\| = \frac{1}{2}\left\|x^i - x^k\right\| \leq \frac{1}{2} l_a$$

2.10 Freedom of Action

In a two-robot swarm with social potential function (4.8), and control law (4.5), robot i will have freedom of action within the boundary layer and hyperball to which it converges. Within the free action zone, where there is neither an attractive force nor a repulsive velocity assigned to the robots, the robots of the swarm will wander freely, and the velocity of the center of the swarm is the mean of the velocities of each member

$$\dot{\bar{x}}(t) = \left(\tfrac{1}{M}\right) \sum_{i=1}^{M} \dot{x}^i(t)$$

This will cause the center of the swarm to wander freely. Consequently, the swarm, like robot i, will have freedom of motion.

2.11 Simulation Results

Using Matlab, several simulations of a two-robot swarm were performed in order to verify the theory. The results of one simulation are presented here as representative results in Figures (2.4), (2.5), and (2.6). In this simulation, robot 1 was started a position (0, 20) and robot 2 at (20, 0), with both having an initial heading of 0°. The desired separation was $r_d^{i,k} = 50$, the tolerance was $\varepsilon = 5$, and the controller constants were $K_1 = 20$, $K_2 = 20$, and $K_{random} = 5$. The controller constant K_{random} is used in the equations

$$x_{1d}^i(t+1) = x_1^i(t) + N_{1,sign}\, K_{random}\, N_{1,var}$$
$$x_{2d}^i(t+1) = x_2^i(t) + N_{2,sign}\, K_{random}\, N_{2,var}$$

in order to simulate random motion. In this equation, K_{random} is used to determine the *maximum size* of the random step, $N_{1,sign} \in \{+1,-1\}$ and $N_{2,sign} \in \{+1,-1\}$ are random variables that are used to determine the *sign* of the x-component and y-component of the step, and $N_{1,var} \subset [0 \ldots 1]$ and $N_{2,var} \subset [0 \ldots 1]$ are random variables that serve as *scaling factors*. Therefore, both the magnitude and the direction of the motion in both the x-axis and y-axis directions are determined randomly. This process determines the desired position for robot i, then the position controller drives the robot to that desired position.

The simulations demonstrate that the swarm converges to a hyperball whose radius is one-half the attraction radius for the robots and whose center is the center of the swarm. Figure (2.4) shows the plot of the trajectories for robot 1 and robot 2 relative to the swarm center with the swarm center as the center of the plot for simulation 1c.

Figure (2.5) shows the trajectory of robot 1 relative to robot 2 with robot 2 as the center of the plot. This plot shows (1) that robot 1 converges to an annulus whose inner radius is the repulsion radius l_r and whose outer radius is the attraction radius l_a, (2) that, due to this

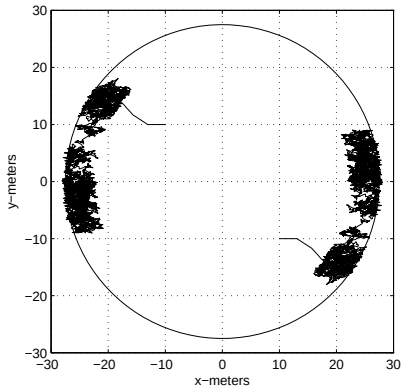

Figure 2.4: Robots 1 and 2 Trajectories Relative to Swarm Center

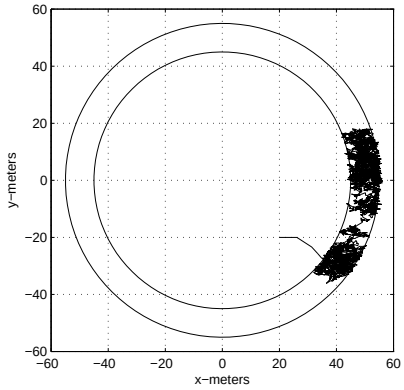

Figure 2.5: Trajectory of Robot 1 Relative to Robot 2

convergence, robot 1 is prevented from colliding with robot 2, and (3) that there is freedom of motion for robot 1 within the annulus.

Figure (2.6) shows the plot of the trajectory of the swarm center for simulation 1c, which is representative of all the simulations. This figure shows that the center of the swarm is not stationary for all time, but moves randomly.

In order to investigate the effect of varying the size of the free action area, a series of simulations was performed in which the tolerance ε and, consequently, the size of the

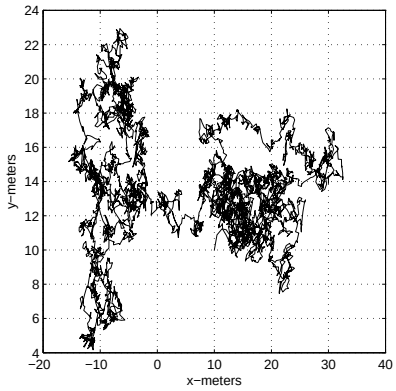

Figure 2.6: Trajectory of the Swarm Center

Simulation	1a	1b	1c	1d	1e
Desired Separation $r_d^{i,k}$	50	50	50	50	50
Tolerance ε	1	2	3	4	5
Free Action Area	2	4	6	8	10

Table 2.1: Simulation Parameters for Simulations 1a-1e

free action area, was varied from $\varepsilon = 1$ to $\varepsilon = 5$, with $r_d^{i,k} = 50$, $K_1 = 20$, $K_2 = 20$, and $K_{random} = 10$ for both robots. Also, robot 1 was started from position (20,0) and robot 2 was started from position (0,20) in all of the simulations. Table 2.1 shows the details of these simulations, which are labeled simulations 1a through 1e. These results were found by averaging the results from ten separate simulation runs.

For each *basis behavior* [14], there is a corresponding *behavioral state*. In our two-robot swarm, the basis behaviors are *convergence* and *free action*. We denote convergence as state 1 and free action as state 2. The *behavioral state residency time* (BSRT) is the total time that a given robot is in a state. The state residency times (in percentages of the total time of the simulation) for robot 1 for the simulations 1a-1e are given in Table 2.2.

Due to the fact that the width of the free wandering area varies from lesser to greater, one predicts that the amount of the time that the robots will spend in the free action state, state 2, will gradually increase from simulation 1a to simulation 1e. The state residency times do increase from simulation 1a to 1e, but not linearly. The curve is that of a rising exponential. Further research is necessary to develop a model for this phenomenon.

Sim	1a	1b	1c	1d	1e
Conv	59.84	35.71	22.61	15.35	11.15
Free	40.16	64.29	77.39	84.65	88.85

Table 2.2: State Residency Times for Robot 1

2.12 Conclusion

In this paper, we have presented a strategy for achieving swarm flocking behavior while simultaneously preventing collisions between the members and allowing freedom of motion for each member. While the results of both the theory and simulations show that this control strategy is effective at achieving flocking behavior for a two-robot swarm, further research is needed to extend this method to larger swarms. This research should concentrate on developing attractive and repulsive artificial potential functions that are implemented by the position tracking controller whose values can be added up in order to produce a single resultant desired position for robot i. Such functions could be used to implement a social potential function for robot i that allows for the desired flocking behavior in a large swarm of robots.

Bibliography

[1] J. C. Latombe, *Robot Motion Planning*, Boston: Kluwer Academic Publishers, 1991.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, P*rinciples of Robot Motion: Theory, Algorithms, and Implementations*, Cambridge, Massachusetts: The MIT Press, 2005.

[3] V. Gazi, and K. M. Passino, "Stability Analysis of Swarms," *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, April 2003, pp. 692-697.

[4] T. Balch and R. C. Arkin, "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, December 1998.

[5] V. Gazi, and K. M. Passino, "A Class of Attraction/Repulsion Functions for Stable Swarm Aggregations," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, pp. 2842-2847.

[6] V. Gazi, and K. M. Passino, "Stability Analysis of Social Foraging Swarms: Combined Effects of Attractant/Repellent Profiles," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, pp. 2848-2853.

[7] V. Gazi, and K. M. Passino, "Stability of Social Foraging Swarms," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 34, No. 1, February 2004, pp. 539-557.

[8] V. Gazi, "Swarm Aggregations Using Artificial Potentials and Sliding Mode Control," *Proceedings of the 42nd IEEE Conference on Decision and Control,* Maui, Hawaii, USA, December 2003, pp. 2041-2046.

[9] V. Gazi and R. Ordonez, "Target Tracking Using Artificial Potentials and Sliding Mode Control" *Proceedings of the 2004 American Control Conference,* Boston, Massachusetts*,* June 30 - July 2, 2004, pp 5588-5593.

[10] V. Gazi, "Swarm Aggregation Using Artificial Potentials and Sliding-Mode Control," *IEEE Transactions on Robotics*, Vol. 21, No.l 6, December 2005, pp. 1208-1214.

[11] R. C. Arkin, *Behavior-based Robotics*, Cambridge, Massachusetts: The MIT Press, 1999.

[12] K. Lerman and A. Galstyan, "A General Methodology for Mathematical Analysis of Multi-Agent Systems," *USC Information Sciences Technical Report ISI-TR-529*, 2001.

[13] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, Upper Saddle River, New Jersey: Prentice Hall, 1991, pp. 42-43, 276-307.

[14] K. Lerman, "Design and Mathematical Analysis of Agent-based Systems," in *Lecture Notes in Artificial Intelligence (LNAI) 1871*, p. 220 ff, Springer-Verlag, Berlin Heidelberg, 2001.

[15] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You, and S.-R. Oh, "A Stable Target-Tracking Control for Unicycle Mobile Robots," *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1822-1827.

[16] R. Seigwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Cambridge, Massachusetts: The MIT Press, 2004, pp. 84-85.

[17] Z. Li, Y. Jia, J. Du, and S. Yuan, "Flocking for Multi-agent Systems with Switching Topology in a Noisy Environment," *2008 American Control Conference*, Westin Seattle Hotel, Seattle, Washington, USA, June 11-13, 2008, pp. 111-116.

[18] A. Regmi, R. Sandoval, R. Byrne, H. Tanner, and C. T. Abdallah, "Experimental Implementation of Flocking Algorithms in Wheeled Mobile Robots," *2005 American Control Conference*, June 8-10, 2005, Portland, OR, USA, pp. 4917-4922.

Chapter 3

Implementation of Artificial Potential Functions Using A Position Tracking Controller

3.1 Introduction

In this paper, we develop a method for implementing artificial potential functions (AFPs) using a position tracking controller. Artificial potential functions are commonly used in motion planning for mobile robots. Attractive APFs are used to drive a robot toward a goal, while repulsive AFPs are used to cause the robot to avoid obstacles. By using both an attractive and a repulsive APF, one can construct a social potential function (SPF) that governs how a robot adjusts its position relative to other robots in a robot team. Such a SPF is useful for creating various kinds of swarming behavior.

The controller that we use is a Lyapunov-based controller that causes the robot to track either a stationary goal position or a time-varying desired trajectory. The position tracking controller is designed for use by a differential-drive mobile robot, which is a common type of mobile robot. The controller implements an attractive APF by causing the robot to move

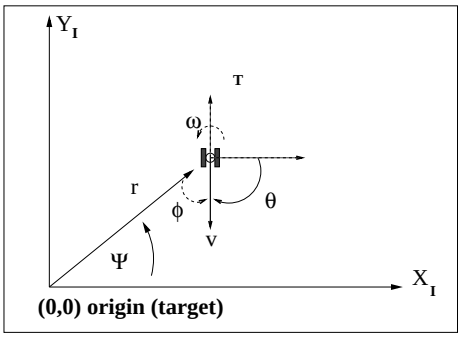

Figure 3.1: Tracking Coordinates

toward a goal position, and it implements a repulsive APF by causing the robot to move away from an obstacle.

3.2 Coordinate System

Our coordinate system uses inertial coordinates, tracking coordinates and relative coordinates. The inertial coordinates define the absolute position of the robot in the workspace. The tracking coordinates define the robot's position relative to a desired position, and, thus, they are used by the robot's controller in order to define control errors. The relative coordinates define the robot's position relative to other robots and obstacles in the workspace.

The *pose* of robot i in the inertial reference frame and Cartesian coordinates is represented by $\xi^i = \begin{bmatrix} x^i & \theta^i \end{bmatrix}^T$, where $x^i = \begin{bmatrix} x_1^i & x_2^i \end{bmatrix}^T$ is the position, x_1^i and x_2^i are the x-axis and y-axis components of the positions respectively, and the angle θ^i is the heading that is measured counterclockwise from the x-axis of the inertial reference frame to the forward direction of the robot.

The *tracking coordinates* $\left(r^i, \phi^i\right)$ are defined by the coordinate transformation [17]

$$
\begin{aligned}
r^i &= \left\| x^i - x^i_d \right\| = \sqrt{\left(x^i_1 - x^i_{d1}\right)^2 + \left(x^i_2 - x^i_{d2}\right)^2} \\
\Psi^i &= \arctan\left(\frac{x^i_2 - x^i_{d2}}{x^i_1 - x^i_{d1}}\right) \\
\phi^i &= 180^o + \theta^i - \Psi^i
\end{aligned}
\tag{3.1}
$$

where the *tracking displacement* r^i is the straight line distance from the robot to the *target* (i.e., another robot or an obstacle) and the *tracking angle* ϕ^i is the angle between the forward direction of the robot and the direction to the target. We define the *position error* for robot i (i.e., the the difference between the actual and desired positions for robot i) as $\tilde{x}^i = x^i - x^i_d$ in inertial coordinates and as $\tilde{r}^i = r^i - r^i_d$ in tracking coordinates, where the desired value of the tracking displacement is defined to be zero (i.e., $r^i_d \equiv 0$).

Similarly, the *relative coordinates* $\left(r^{i,k}, \phi^{i,k}\right)$ are defined by the coordinate transformation

$$
\begin{aligned}
r^{i,k} &= \left\| x^i - x^k \right\| = \sqrt{\left(x^i_1 - x^k_1\right)^2 + \left(x^i_2 - x^k_2\right)^2} \\
\Psi^{i,k} &= \arctan\left(\frac{x^i_2 - x^k_2}{x^i_1 - x^k_1}\right) \\
\phi^{i,k} &= 180^o + \theta^i - \Psi^{i,k}
\end{aligned}
\tag{3.2}
$$

where x^k is the *reference position* (i.e., the position of an obstacle or another robot), $r^{i,k}$ is the *separation* (or *relative displacement*) and $\phi^{i,k}$ is the *relative angle*. The *separation error* (i.e., the difference between the actual and desired distances of separation between robot i and robot k) is $\tilde{r}^{i,k} = r^{i,k} - r^{i,k}_d$. The *relative position* is $x^{i,k} = x^i - x^k$, which is a vector that points from robot k to robot i. Also, the *desired relative position* is $x^{i,k}_d = x^i_d - x^k$.

3.3 Differential-Drive Robot Kinematics

The kinematics of a differential-drive mobile robot i with coordinates in the inertial frame are given by [18]

$$\dot{\xi}^i = \begin{bmatrix} \dot{x}_1^i \\ \dot{x}_2^i \\ \dot{\theta}^i \end{bmatrix} = \begin{bmatrix} v^i \cos\theta^i \\ v^i \sin\theta^i \\ \omega^i \end{bmatrix} \tag{3.3}$$

where ξ^i is the pose, x_1^i and x_2^i are the x-axis and y-axis components of the robot's position, θ^i is the robot's heading, v is the robot's linear velocity, ω^i is the robot's angular velocity, and the dots indicate the time-derivative of each quantity. Also, v^i and ω^i are the control inputs for the robot.

When the tracking coordinate transformation (4.1) is applied to the kinematics (4.3), we robot, we get the *tracking kinematics* [17]

$$\begin{aligned} \dot{r}^i &= -v^i \cos(\phi^i) \\ \dot{\phi}^i &= \omega^i + \tfrac{v^i}{r^i} \sin(\phi^i) \end{aligned} \tag{3.4}$$

3.4 Position Tracking Controller

Lee et. al [17] have designed a stable position tracking controller for a differential-drive mobile robot that is based on the coordinate transformation (4.1). The following lemma is a restatement of the main result of [17].

Lemma 1: [17] Consider a differential-drive robot with kinematics (4.3) that uses the tracking coordinates (4.1) in order to obtain the tracking kinematics (4.4). If $\vee t \in [0,\infty)$ with controls laws given by (4.5)

$$\begin{aligned} v^i &= K_1\, r^i \cos(\phi^i) \\ \omega^i &= -K_1\, \sin(\phi^i) \cos(\phi^i) - K_2\, \phi^i \end{aligned} \tag{3.5}$$

then r^i and ϕ^i are uniformly bounded. Furthermore, we have $\lim_{t\to\infty} \left\| \left(r^i, \phi^i\right)^T \right\| = 0$.

Remark 1: This implies that $r^i = \left\| x^i - x_d^i \right\| \to 0$, and, therefore, that $x^i \to x_d^i$ as $t \to \infty$.

3.5 Quadratic Attractive Function

An attractive APF causes the robot to engage in attractive behavior. We define *attractive behavior* for robot i as follows: If robot i (at position x^i) is outside of the *radius of influence for attraction* r_{att}, the robot i moves closer to the reference position x^k (i.e., $x^i \rightarrow x^k$and $r^i \rightarrow r^i_d$) at a desired velocity $\dot{x}^i_d$ (or speed $\dot{r}^i_d$) via a path that approximates a straight-line as much as possible (i.e., $\phi^{i,k} \rightarrow 0$); if robot i is inside r_{att}, then robot i does not move at all (i.e., $\dot{x}^i = 0$). We implement the quadratic attractive APF by using the position tracking controller in concert with an appropriate switching rule.

We need to derive the desired position and desired velocity for the robot. We will do this by analyzing the standard attractive APF [4] to which an appropriate switching rule has been applied. This function is give by

$$U_{att} = \left[\frac{1}{2} K_1 \left(r^{i,k}\right)^2\right] \sigma_1 + [0]\, \sigma_2 \tag{3.6}$$

with the switching rule defined by

$$(\sigma_1, \sigma_2) = \begin{cases} (1,0): & r^{i,k} \geq r_{att} \\ (0,1): & r^{i,k} < r_{att} \end{cases} \tag{3.7}$$

Using the gradient descent method, we define the desired velocity for robot i as the negative of the gradient of (3.6), which is given by

$$\dot{x}^i_d = -\nabla U_{att} = \left[-K_1 \left(x^i - x^k\right)\right] \sigma_1 + [0]\, \sigma_2$$

(Note that the desired velocity is defined relative to the reference position.)

Finally, the desired value $\left(\dot{r}^i\right)_d$ of the tracking speed is a scalar whose magnitude is the same as the magnitude of the desired tracking velocity $\left(\dot{x}^i\right)_d$, and the desired value for the tracking angle ϕ^i_d is zero since we want the robot to move toward the reference position in

a straight line.

$$\begin{aligned} \left(\dot{r}^i\right)_d &= \left[-K_1\ r^{i,k}\right]\sigma_1 + [0]\,\sigma_2 \\ \phi_d^i &= 0 \end{aligned} \tag{3.8}$$

In order to get attractive behavior, we define the desired position for robot i as the reference position (i.e., $x_d^i \equiv x^k$) and use the control law (4.5) in concert with the switching rule (3.7). This gives us the switched control law $u = [v\ \ \omega]^T$ for the attractive APF.

$$\begin{bmatrix} v^i \\ \omega^i \end{bmatrix} = \begin{bmatrix} K_1\, r^i\, \cos(\phi^i) \\ -K_1\, sin(\phi^i)\, \cos(\phi^i) - K_2\, \phi^i \end{bmatrix} \sigma_1 + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \sigma_2 \tag{3.9}$$

3.6 Tracking Theorems For Attractive Function

At this point, we present two theorems. The first theorem shows that the position tracking controller cause the robot to track the desired position and, thereby, implements the quadratic attractive function. The second theorem shows that the position tracking controller form of the quadratic attractive function is equivalent to the standard vector form of the function and that the controller form converges to the standard form.

3.6.1 Position Tracking Theorem

The following lemma demonstrates the control law (3.9) implements the attractive APF properly.

Lemma 2: The control law (3.9) causes the differential-drive robot i to engage in attractive behavior in regard to the reference position x^k. That is, $r^i \to 0$ and $\phi^i \to 0$ as $t \to \infty$, and, consequently, $x^i \to x^k$ as $t \to \infty$ as long as $r^{i,k} \geq r_{att}$.

Proof: We consider the case when $r^{i,k} \geq r_{att}$. In this case, the control law (3.9) reduces to the control law (4.5) for the position tracking controller. Lemma 1 shows us that $r^i \to 0$ and $\phi^i \to 0$ as $t \to \infty$, which implies that $x^i \to x_d^i$. Since $x_d^i \equiv x^k$, this means that $x^i \to x^k$.

The attractive function makes the surface $S^i_{att} = \left\{x^i : r^{i,k} = r_{att}\right\}$ an attractive surface for all points $\left\{x^i : r^{i,k} \geq r_{att}\right\}$. Therefore, we have $x^i \rightarrow A^i_{d(att)}$, where $A^i_{d(att)}$ is the desired area of convergence that is defined by $A^i_{d(att)} = \left\{x^i : r^i < r_{att}\right\}$.

3.6.2 Velocity Tracking Theorem

Lemma 3: The actual tracking speed $\dot{r}^i$ will converge to the desired tracking speed $\left(\dot{r}^i\right)_d$ and the tracking angle ϕ^i will converge to the the desired tracking angle $\phi^i_d \equiv 0$. That is, $\dot{r}^i \rightarrow \left(\dot{r}^i\right)_d$ and $\phi^i \rightarrow 0$ as $t \rightarrow \infty$.

Proof: Lemma 2 shows that $\phi^i \rightarrow \phi^i_d \equiv 0$. Therefore, we have $\phi^i \rightarrow 0$ as $t \rightarrow \infty$. While $r^{i,k} \geq r_{att}$, the actual tracking speed is given by

$$\dot{r}^i = -K_1 \cos^2(\phi^i)\, r^i$$

The term $\cos^2(\phi^i) \rightarrow 1$ as $\phi^i \rightarrow 0$. Therefore, $\dot{r}^i \rightarrow \left(\dot{r}^i\right)_d$, where $\left(\dot{r}^i\right)_d$ is given by equation (3.8).

3.7 Repulsive Potential Function

A repulsive APF causes the robot to engage in repulsive behavior. We define *repulsive behavior* for robot i as follows: If robot i (at position x^i) is inside of the *radius of influence for repulsion* r_{rep}, the robot i moves away from the reference position x^k (i.e., $\neg\left(x^i \rightarrow x^k\right)$ at a desired velocity $\dot{x}^i_d$ (or speed $\dot{r}^i_d$) and toward the desired position x^i_d (i.e., $x^i \rightarrow x^i_d$) via a path that approximates a straight-line as much as possible (i.e., $\alpha^{i,k} \rightarrow 0$); if robot i is outside r_{rep}, then robot i does not move at all (i.e., $\dot{x}^i = 0$). (Note that $x^i_d \neq x^k$ for repulsive behavior.) We implement the repulsive APF by using a modified version of the position tracking controller in concert with an appropriate switching rule.

We need to derive the desired values for the repulsive APF. Also, we need to derive a

formula for the desired position that will cause the robot to move away from the reference position. The switching rule for the repulsive APF is defined by

$$(\sigma_3, \sigma_4) = \begin{cases} (1,0): & r^{i,k} \leq r_{rep} \\ (0,1): & r^{i,k} > r_{rep} \end{cases} \tag{3.10}$$

3.7.1 Desired Speed and Tracking Angle for Repulsive Function

The standard form of the repulsive function [4] is

$$U_{rep} = \left[\frac{1}{2}K_3\left(\frac{1}{r^{i,k}} - \frac{1}{r_{rep}}\right)^2\right]\sigma_3 + [0]\,\sigma_4 \tag{3.11}$$

where $r^{i,k}$ is the distance from the robot to the obstacle and r_{rep} is the radius of influence for repulsion. The gradient of this function is

$$\nabla U_{rep} = \left[K_3 g_3 \nabla r^{i,k}\right]\sigma_3 + [0]\,\sigma_4 \tag{3.12}$$

where $\nabla r^{i,k} = \frac{x^i - x^k}{r^{i,k}}$, $g_3 = \left(\frac{1}{r_{rep}} - \frac{1}{r^{i,k}}\right)\frac{1}{\left(r^{i,k}\right)^2}$ (with $g_3 < 0$). Using these relations, we can rewrite the above as

$$\nabla U_{rep} = \left[-K_3 f_3\,(x^i - x^k)\right]\sigma_3 + [0]\,\sigma_4 \tag{3.13}$$

where $f_3^{i,k} = -g_3\,\frac{1}{r^{i,k}} = -\left(\frac{1}{r_{rep}} - \frac{1}{r^{i,k}}\right)\frac{1}{\left(r^{i,k}\right)^3}$ (with $f_3 > 0$). Using the gradient descent method, we find the desired velocity for robot i from the following equation

$$\left(\dot{x}^{i,k}\right)_d = -\nabla U_{rep} = \left[K_3 f_3^{i,k}\,(x^i - x^k)\right]\sigma_3 + [0]\,\sigma_4$$

We calculate the desired speed and tracking angle as follows.

Finally, the desired value $\left(\dot{r}^{i,k}\right)_d$ of the relative speed is a scalar whose magnitude is the same as the magnitude of the desired relative velocity $\left(\dot{x}^{i,k}\right)_d$, and the desired value for

the tracking angle $\phi^i_{d(att)}$ is zero since we want the robot to move away from the reference position and toward the desired position in a straight line.

$$\begin{gathered}\left(\dot{r}^{i,k}_d\right)_d = \left[K_3 f^{i,k}_3 \; r^{i,k}\right] \sigma_3 + [0]\, \sigma_4 \\ \phi^i_d = 0\end{gathered} \tag{3.14}$$

3.7.2 Desired Position for the Repulsive Function

In order to implement the repulsive function with the position tracking controller, we must derive a formula for calculating a desired position for the robot that is further away from the obstacle than the robot's current position. The controller will drive the robot to this desired position, thereby moving it further way from the obstacle. The result is the repulsive behavior that we want.

We derive the desired position formula for the repulsive function by equating the gradients of the attractive function and the repulsive function and then solving for the the reference position of the gradient of the attractive function.

$$\dot{x}^i = -\nabla U_{att} = -\nabla U_{rep} \tag{3.15}$$

We set the gradients equal to each other ($\nabla U_{att} = \nabla U_{rep}$). Here, we will consider the non-switched version of the functions.

$$K_1 \left(x^i - x^i_{d(rep)}\right) = -K_3 f^{i,k}_3 \left(x^i - x^k\right)$$

Next, we solve for $x^i_{d(rep)}$ in order to get

$$x^i_{d(rep)} = x^i + \frac{K_3}{K_1} f^{i,k}_3 \left(x^i - x^k\right) \tag{3.16}$$

3.7.3 Control Law for Repulsive Function

We implement the repulsive APF by using the control law

$$\begin{bmatrix} v^i \\ \omega^i \end{bmatrix} = \begin{bmatrix} K_1 \cos(\phi^i)\, r^i \\ -K_1 \cos(\phi^i)\sin(\phi^i) - K_2\, \phi^i \end{bmatrix} \sigma_3 + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \sigma_4 \tag{3.17}$$

where r^i and ϕ^i are found using the formula for the desired position for repulsion (3.16).

3.7.4 Closed-Loop Kinematics for the Repulsive Function

This control law gives us the following closed-loop kinematics

$$\begin{bmatrix} \dot{r}^i \\ \dot{\phi}^i \end{bmatrix} = \begin{bmatrix} -K_1 \cos^2(\phi^i)\, r^i \\ -K_2\, \phi^i \end{bmatrix} \sigma_3 + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \sigma_4 \tag{3.18}$$

We also need the closed-loop kinematics for the relative distance $r^{i,k}$. We begin with the tracking coordinates in terms of the relative distance from the reference position.

$$\dot{r}^{i,k} = -v^i \cos(\phi^{i,k})$$

We use the the fact that $\cos\left(\phi^{i,k} + \pi\right) = \cos\left(\phi^i\right)$ and the trigonometric identity $\cos\left(\phi^{i,k} + \pi\right) = -\cos\left(\phi^{i,k}\right)$ in order to derive the relationship $\cos\left(\phi^{i,k}\right) = -\cos\left(\phi^i\right)$. Using this relation, we get

$$\dot{r}^{i,k} = v^i \cos(\phi^i)$$

We substitute the control law for the velocity in to the above in order to get

$$\dot{r}^{i,k} = K_1 \cos^2(\phi^i)\, r^i$$

Using (3.16). we calculate $r^i = \left\| x^i - x_d^i \right\|$ as

$$r^i = \left\| x^i - \left[x^i + \frac{K_3}{K_1} f_3^{i,k} \left(x^i - x^k \right) \right] \right\| = \frac{K_3 f_3^{i,k}}{K_1} r^{i,k}$$

This gives us the relationship between r^i and $r^{i,k}$ that we need in order to obtain the final closed-loop kinematics

$$\dot{r}^{i,k} = K_3 f_3^{i,k} \cos^2(\phi^i)\, r^{i,k}$$

The equation for the full, switched closed-loop kinematics becomes

$$\dot{r}^{i,k} = \left[K_3 f_3^{i,k} \cos^2(\phi^i)\, r^{i,k} \right] \sigma_3 + [0]\, \sigma_4 \tag{3.19}$$

3.8 Tracking Theorems for Repulsive Function

We now present two tracking theorems for the repulsive function. The first theorem shows that the position tracking controller cause the robot to track the desired position and, thereby, implements the repulsive function. The second theorem shows that the position tracking controller form of the repulsive function is equivalent to the standard vector form of the function and that the controller form converges to the standard form.

3.8.1 Position Tracking Theorem for Repulsive Function

Lemma 4: The control law (3.17) causes the differential-drive robot i to engage in repulsive behavior in regard to the reference position x^k. That is, $r^i \to 0$, $\phi^i \to 0$ and $r^{i,k} = \left\| x^i - x^k \right\| \to \infty$ as $t \to \infty$, and, consequently, robot i will move away from x^k as $t \to \infty$ as long as $r^{i,k} \leq r_{rep}$. In other words, as robot i moves closer to the desired position $x^i_{d(rep)}$, it moves further away from the reference position x^k.

Proof: The desired position $x^i_{d(rep)}$ is defined by the formula (3.16). We use Lemma 1

to show that $r^i \to 0$, $\phi^i \to 0$ as $t \to \infty$. This is true for $r^{i,k} \leq r_{rep}$. When $r^{i,k} > r_{rep}$, the robot will stop moving. In other words, the robot will move toward the desired position while it is inside the radius of influence for repulsion, and, once it is outside the radius of influence, it will stop and remain stationary.

Consider the Lyapunov function

$$V\left(\frac{1}{r^{i,k}}\right) = \frac{1}{2}\left(\frac{1}{r^{i,k}}\right)^2$$

The time derivative of this function is

$$\dot{V}\left(\frac{1}{r^{i,k}}\right) = -\frac{1}{\left(r^{i,k}\right)^3}\dot{r}^{i,k}$$

The relative kinematics (i.e., the kinematics for the relative motion between robot i and the reference position k) are given by (3.19). Thus, we get

$$\dot{V}\left(\frac{1}{r^{i,k}}\right) = -K_3 f_3^{i,k}\frac{1}{\left(r^{i,k}\right)^2}\cos^2(\phi^i) < 0$$

We have that $\dot{V}\left(\frac{1}{r^{i,k}}\right) < 0$ for $\left(0 < r^{i,k} \leq r_{rep}\right)$. (Note $f_3^{i,k}\frac{1}{\left(r^{i,k}\right)^2}$ becomes infinite when $r^{i,k} = 0$.) Thus, we conclude that $\frac{1}{r^{i,k}} \to 0$ as $t \to \infty$. This implies that $r^{i,k} = \left\|x^i - x^k\right\| \to \infty$ as $t \to \infty$, which, in turn, implies that robot i moves away from x^k as $t \to \infty$.

Finally, the repulsive function makes the reference position x^k repulsive within the region inside the radius r_{rep}, which will cause $x^i \to B_d$ where

$$B_d = \left\{x^i : r^{i,k} \geq r_{rep}\right\}$$

3.8.2 Velocity Tracking Theorem for Repulsive Function

Lemma 5: The actual tracking speed $\dot{r}^i$ will converge to the desired tracking speed $\left(\dot{r}^i\right)_d$ and the tracking angle ϕ^i will converge to the the desired tracking angle $\phi^i_d \equiv 0$. That is, $\dot{r}^i \rightarrow \left(\dot{r}^i\right)_d$ and $\phi^i \rightarrow 0$ as $t \rightarrow \infty$.

Proof: Lemma 4 shows that $\phi^i \rightarrow \phi^i_d \equiv 0$. Therefore, we have $\phi^i \rightarrow 0$ as $t \rightarrow \infty$. While $r^{i,k} \leq r_{rep}$, the actual tracking speed is given by

$$\dot{r}^{i,k} = K_3 f_3^{i,k} \cos^2(\phi^i)\, r^{i,k}$$

The term $\cos^2(\phi^i) \rightarrow 1$ as $\phi^i \rightarrow 0$. Therefore, $\dot{r}^i \rightarrow \left(\dot{r}^i\right)_d$, where $\left(\dot{r}^i\right)_d$ is given by equation (3.14).

3.9 Simulation Results

Figure (3.2) shows the results of a simulation for the quadratic attractive function. It shows the results of eight separate simulations in which the robot was started from eight different initial poses (i.e., positions and headings) around the origin, and the robot was driven toward the origin. Both the ideal trajectories (dotted lines) and the actual trajectories (solid lines) are shown. All of the starting positions are on a circle of radius 1000 at the eight cardinal points. The initial heading is $\theta^i = 0^\circ$ for all of the simulations. This produces the star pattern shown in the figure. In these simulations, the controller constants are $K_1 = 1$, $K_2 = 5$ and $r_{att} = 1$.

Figure (3.2) shows us that the robot will move both forward and backward as it moves toward the goal position. It will also turn right in order to get itself moving forward in the direction of the goal. Notice that the robot does not turn left due to the nature of the position tracking controller.

The best results are obtained when we set the controller constants such that $K_2 \gg K_1$.

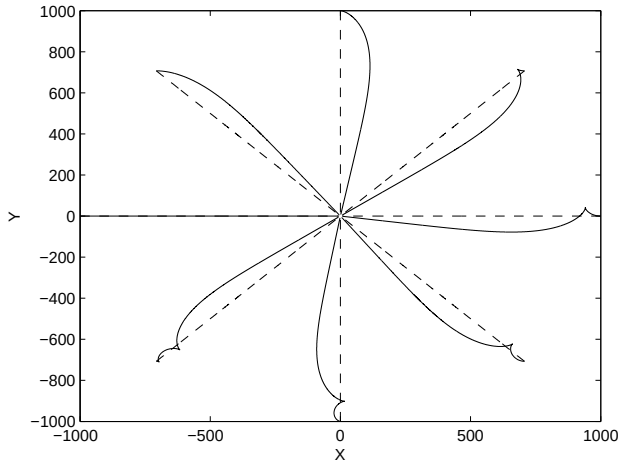

Figure 3.2: Simulation for attractive function with the robots moving toward the center of the plot from starting points at the eight cardinal points. The dotted lines indicate the ideal trajectories for the robots.

The constant K_1 controls the robots translational speed, and the constants K_1 and K_2 together control its rotational speed (i.e., turning speed). Thus, when we set $K_2 \gg K_1$, then the robot can turn in a relatively smaller turning radius, which gives it better maneuverability.

Figure (3.3) shows a plot of the function f_3 with $r_{rep} = 5$ and $K_3 = 2$. Notice that the magnitude of the function approaches zero very rapidly as $r^{i,k} \rightarrow r_{rep}$. When $r^{i,k} \approx 4$, then $f_3 \approx 0$. This means that robot will slow down and almost stop moving before it is actually outside of the repulsion radius. Thus, one will have to make r_{rep} slightly larger than actual desired repulsion radius. One can do this by plotting f_3 for several values of r_{rep} in order to select the best value of it.

Figure (3.4) shows the results of a simulation with the repulsive function. In each simulation, the robot starts on the unit circle and drives away from the origin. This is good, but not perfect, tracking of the ideal trajectories (dotted lines) by the actual trajectories (solid lines). The tracking error is symmetrical, which makes the errors effectively cancel each other out. The overall result is good repulsive behavior by the robot.

These simulations show that the position tracking controller can produce the desired

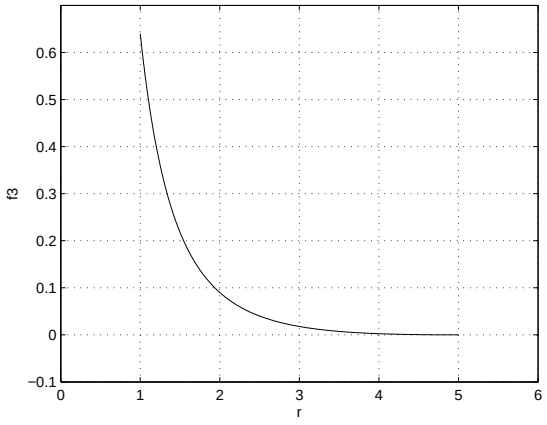

Figure 3.3: Plot of $f_3^{i,k}$ showing its variation near the reference point.

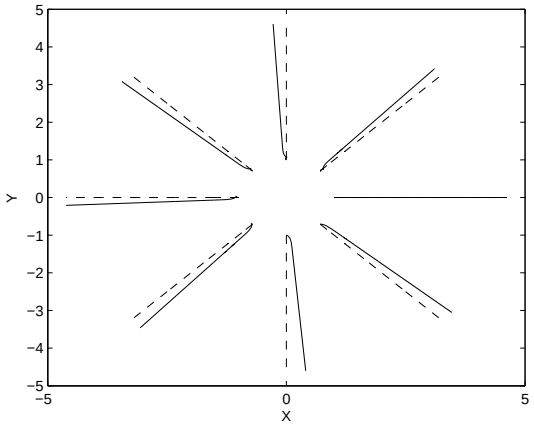

Figure 3.4: Simulation for repulsive function with the robots moving away from the center of the plot toward the eight cardinal points. The dotted lines indicate the ideal trajectories for the robots.

attractive or repulsive behavior. The controller implements the attractive and repulsive functions well. Overall, the simulations show that our method for implementing these artificial potential functions is successful.

3.10 Conclusion

In this paper, we have developed a method for the implementation of both attractive and repulsive artificial potential functions. We used a position tracking controller in order to derive control laws for a quadratic attractive function and for a repulsive function. Both theoretical and simulations results show that the method is practical and successful.

Bibliography

[1] R. P. Samples and R. P. Leland, "Investigation of Flocking in a Small Swarm of Wheeled Mobile Robots Using Lyapunov Control Theory," in review.

[2] R. P. Samples and R. P. Leland, "Analysis of Flocking in a Large Swarm of Wheeled Mobile Robots Using Lyapunov Control Theory," in review.

[3] J. C. Latombe, *Robot Motion Planning*, Boston: Kluwer Academic Publishers, 1991.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, Cambridge, Massachusetts: The MIT Press, 2005, pp. 80-85.

[5] V. Gazi, and K. M. Passino, "Stability Analysis of Swarms," *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, April 2003, pp. 692-697.

[6] T. Balch and R. C. Arkin, "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, December 1998.

[7] V. Gazi, and K. M. Passino, "A Class of Attraction/Repulsion Functions for Stable Swarm Aggregations," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, pp. 2842-2847.

[8] V. Gazi, and K. M. Passino, "Stability Analysis of Social Foraging Swarms: Combined Effects of Attractant/Repellent Profiles," *Proceedings of the 41st IEEE Con-*

ference on Decision and Control, Las Vegas, Nevada, USA, December 2002, pp. 2848-2853.

[9] V. Gazi, and K. M. Passino, "Stability of Social Foraging Swarms," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 34, No. 1, February 2004, pp. 539-557.

[10] V. Gazi, "Swarm Aggregations Using Artificial Potentials and Sliding Mode Control," *Proceedings of the 42nd IEEE Conference on Decision and Control,* Maui, Hawaii, USA, December 2003, pp. 2041-2046.

[11] V. Gazi and R. Ordonez, "Target Tracking Using Artificial Potentials and Sliding Mode Control" *Proceedings of the 2004 American Control Conference,* Boston, Massachusetts, June 30 - July 2, 2004, pp 5588-5593.

[12] V. Gazi, "Swarm Aggregation Using Artificial Potentials and Sliding-Mode Control," *IEEE Transactions on Robotics*, Vol. 21, No.l 6, December 2005, pp. 1208-1214.

[13] R. C. Arkin, *Behavior-based Robotics*, Cambridge, Massachusetts: The MIT Press, 1999.

[14] K. Lerman and A. Galstyan, "A General Methodology for Mathematical Analysis of Multi-Agent Systems," *USC Information Sciences Technical Report ISI-TR-529*, 2001.

[15] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, Upper Saddle River, New Jersey: Prentice Hall, 1991, pp. 42-43, 276-307.

[16] K. Lerman, "Design and Mathematical Analysis of Agent-based Systems," in *Lecture Notes in Artificial Intelligence (LNAI) 1871*, p. 220 ff, Springer-Verlag, Berlin Heidelberg, 2001.

[17] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You, and S.-R. Oh, "A Stable Target-Tracking Control for Unicycle Mobile Robots," *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1822-1827.

[18] R. Seigwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Cambridge, Massachusetts: The MIT Press, 2004, pp. 84-85.

[19] Z. Li, Y. Jia, J. Du, and S. Yuan, "Flocking for Multi-agent Systems with Switching Topology in a Noisy Environment," *2008 American Control Conference*, Westin Seattle Hotel, Seattle, Washington, USA, June 11-13, 2008, pp. 111-116.

[20] A. Regmi, R. Sandoval, R. Byrne, H. Tanner, and C. T. Abdallah, "Experimental Implementation of Flocking Algorithms in Wheeled Mobile Robots," *2005 American Control Conference*, June 8-10, 2005, Portland, OR, USA, pp. 4917-4922.

Chapter 4

Stability Analysis of Flocking in a Swarm of Wheeled Mobile Robots

4.1 Introduction

In this paper, we study the *flocking* behavior of a M-member swarm S^M of differential-drive mobile robots. In flocking behavior, each robot keeps itself within both a maximum and a minimum distance away from the other robots, but the swarm does not organize itself into a rigid formation. Flocking creates a "loose formation" in which each robot has some freedom of action. On the other hand, *forming* creates a rigid formation in which each robot is constrained in terms of its relative distance from other robots, which means that individual robots have little or no freedom of action.

The use of artificial potential functions (APFs) in the control of robots was explored in [1]. The theory was later extended in various ways [2][3][4]. Today, the use of APFs is a common method for the control of mobile robots [5][6].

V. Gazi and K. M. Passino addressed the stability analysis of swarms [7]. Their method concentrated on a swarm of robots that were modeled as omnidirectional particles. They developed their method for swarms that engage in foraging, swarms in noisy environments,

and other situations [10][11]. They also defined a class of functions that can be used to construct social potential functions for robots [9]. Gazi and Ordonez further developed the theory by adding the use of sliding mode control [13][14].

Our research builds on that of Gazi and Passino [7] by extending the analysis of robot swarms in several ways. Firstly, we study the case of *flocking*, whereas Gazi and Passino studied *forming*. Secondly, we use a swarm of differential-drive mobile robots instead of omnidirectional particle robots. These robots use a a position tracking controller [19] that was derived using Lyapunov control theory in order to drive the robots. Thirdly, we adapt their analytical methods to the kind of swarm that we use. Fourthly, we develop additional analytical and graphical methods for demonstrating that a robot has a degree of freedom of motion in our swarm. Finally, we use standard artificial potential functions [6] in order to construct a social potential function (SPF) for the robots.

Overall, our research synthesizes and extends previous research in the fields of (a) the control-theoretic analysis of swarms, (b) the use of artificial potential functions, the *reactive paradigm* [23] and the subsumption architecture for robot motion planning, (c) the application of position controllers for differential-drive (or unicycle-style) mobile robots and (d) the use of switching in the control of mobile robots.

We assume that the workspace has no obstacles and that all of the robots in the swarm are identical (i.e., we assume a homogeneous swarm). Each robot i uses a social potential function composed of standard artificial potential functions (APFs) and the gradient descent method in order to plan its motion. The motion of the robot is controlled by a position tracking controller that drives robot i from its current position x^i to the desired position x^i_d.

We use three-step process in this paper for the analysis and design of flocking behavior in robot swarms: (1) use control-theoretic techniques to analyze the ideal behavior of each robot and the swarm as a whole; (2) use insights from the control-theoretic analysis in step 1 in order to develop a flocking algorithm; and (3) use simulations to demonstrate our method.

Path planning is determining how the robot ought to move in the workspace, and *Motion control* is causing the robot to move along a path is as close as possible to the desired path that the motion planning specifies [3]. In our method, path planning is simplified by the insights gained from a preliminary analysis of the behavior of the robot under certain conditions, namely, additive repulsive and attractive behavior. This analysis tells us that the desired path for a robot is simply a straight line directly away from the mean of the positions of a subset of robots. This insight allows us to obtain the desired repulsive or attractive behavior by calculating a simple mean instead of calculating the sum of multiple mathematical functions, which is much less computationally expensive. The motion control is accomplished by a position tracking controller.

4.2 Coordinate System

Our coordinate system uses inertial coordinates, tracking coordinates and relative coordinates. The inertial coordinates define the absolute position of the robot in the workspace. The tracking coordinates define the robot's position relative to a desired position, and, thus, they are used by the robot's controller in order to define control errors. The relative coordinates define the robot's position relative to other robots and obstacles in the workspace.

The *pose* of robot i in the inertial reference frame and Cartesian coordinates is represented by $\xi^i = \begin{bmatrix} x^i & \theta^i \end{bmatrix}^T$, where $x^i = \begin{bmatrix} x_1^i & x_2^i \end{bmatrix}^T$ is the position, x_1^i and x_2^i are the x-axis and y-axis components of the positions respectively, and the angle θ^i is the heading that is measured counterclockwise from the x-axis of the inertial reference frame to the forward direction of the robot.

The *tracking coordinates* $\left(r^i, \phi^i\right)$ are defined by the coordinate transformation [19]

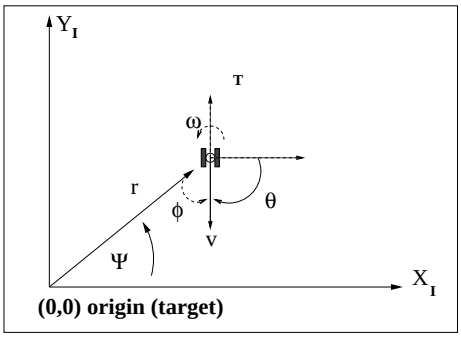

Figure 4.1: Tracking Coordinates

$$\begin{aligned} r^i &= \left\| x^i - x^i_d \right\| = \sqrt{\left(x^i_1 - x^i_{d1}\right)^2 + \left(x^i_2 - x^i_{d2}\right)^2} \\ \Psi^i &= \arctan\left(\frac{x^i_2 - x^i_{d2}}{x^i_1 - x^i_{d1}}\right) \\ \phi^i &= 180^o + \theta^i - \Psi^i \end{aligned} \tag{4.1}$$

where r^i is the straight line distance from the robot to the *target* (i.e., another robot or an obstacle) and ϕ^i is the angle between the forward direction of the robot and the direction to the target.

Similarly, the *relative coordinates* $\left(r^{i,k}, \phi^{i,k}\right)$ are defined by the coordinate transformation

$$\begin{aligned} r^{i,k} &= \left\| x^i - x^k \right\| = \sqrt{\left(x^i_1 - x^k_1\right)^2 + \left(x^i_2 - x^k_2\right)^2} \\ \Psi^{i,k} &= \arctan\left(\frac{x^i_2 - x^k_2}{x^i_1 - x^k_1}\right) \\ \phi^{i,k} &= 180^o + \theta^i - \Psi^{i,k} \end{aligned} \tag{4.2}$$

where x^k is the *reference position* (i.e., the position of an obstacle or another robot), $r^{i,k}$ is the *separation* (or *relative displacement*) and $\phi^{i,k}$ is the *relative angle*. The *relative position* is $x^{i,k} = x^i - x^k$, which is a vector that points from robot k to robot i. Also, the *desired relative position* is $x^{i,k}_d = x^i_d - x^k$.

4.3 Differential-Drive Robot Kinematics

The kinematics of a differential-drive mobile robot i with coordinates in the inertial frame are given by [20]

$$\dot{\xi}^i = \begin{bmatrix} \dot{x}_1^i \\ \dot{x}_2^i \\ \dot{\theta}^i \end{bmatrix} = \begin{bmatrix} v^i \cos\theta^i \\ v^i \sin\theta^i \\ \omega^i \end{bmatrix} \tag{4.3}$$

where ξ^i is the pose, x_1^i and x_2^i are the x-axis and y-axis components of the robot's position, θ^i is the robot's heading, v is the robot's linear velocity, ω^i is the robot's angular velocity, and the dots indicate the time-derivative of each quantity. Also, v^i and ω^i are the control inputs for the robot.

When the tracking coordinate transformation (4.1) is applied to the kinematics (4.3), we robot, we get the *tracking kinematics* [19]

$$\begin{aligned} \dot{r}^i &= -v^i \cos(\phi^i) \\ \dot{\phi}^i &= \omega^i + \tfrac{v^i}{r^i} \sin(\phi^i) \end{aligned} \tag{4.4}$$

which are valid for both stationary and moving targets.

4.4 Position Tracking Controller

Lee et. al [19] have designed a stable position tracking controller for a differential-drive mobile robot that is based on the coordinate transformation (4.1). The following lemma is a restatement of the main result of [19].

Lemma 1: [19] Consider a differential-drive robot with kinematics (4.3) that uses the tracking coordinates (4.1) in order to obtain the tracking kinematics (4.4). If $\forall t \in [0,\infty)$

with controls laws given by (4.5)

$$\begin{aligned} v^i &= K_1\, r^i \cos(\phi^i) \\ \omega^i &= -K_1\, \sin(\phi^i)\cos(\phi^i) - K_2\, \phi^i \end{aligned} \tag{4.5}$$

then r^i and ϕ^i are uniformly bounded. Furthermore, we have $\lim_{t\to\infty} \left\| \left(r^i, \phi^i\right)^T \right\| = 0$.

Remark 1: This implies that $r^i = \left\| x^i - x_d^i \right\| \to 0$, and, therefore, that $x^i \to x_d^i$ as $t \to \infty$. Also, the closed-loop kinematics for r^i and ϕ^i

$$\begin{aligned} \dot{r}^i &= -K_1\, \cos^2(\phi^i)\, r^i \\ \dot{\phi}^i &= -K_2\, \phi^i \end{aligned} \tag{4.6}$$

indicate that the closed-loop system is stable and that r^i and ϕ^i will both converge to zero as $t \to \infty$.

4.5 Model of the Swarm

We consider a swarm S^M of M differential-drive robots in a two-dimensional Euclidean space. The position of robot (or member) i is given by $x^i \in R^2$, and the kinematics [20] of robot i are are described by (4.3). We assume synchronous motion and no time delays (i.e., the robots move simultaneously and each robot knows the position of every other robot) and that the motion of the robots occurs in continuous time [7].

4.5.1 Social Potential Function

An *artificial potential function* (APF) is a differentiable real-valued function $U : R^m \to R$ that is used to create an artificial velocity field that directs the motion of the robot. The velocity is found by taking the *gradient* ∇U

$$\nabla U(q) = \left[\frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q) \right]^T$$

which is a vector that points in the direction that locally maximally increases U [6]. (Note that in robotics the gradient of an artificial potential function is defined as a *velocity*, whereas in physics the gradient of a potential function is usually defined as a *force*.)

A *social potential function* is an artificial potential function that governs the relative distance between the members of the swarm [7]. A social potential function that produces flocking behavior needs to have a component for short-range repulsion that works to prevent collisions, a component for long-range attraction that works to maintain the cohesiveness of the swarm and a component for a zero-valued mid-range region in which the robot acts freely. Such a SPF should be designed to produce the following relative motion between robots i and k.

$$\dot{r}^{i,k} = \begin{cases} \dot{r}^{i,k} > 0: & r^{i,k} \leq r_{rep} \\ \dot{r}^{i,k} = 0: & r_{rep} < r^{i,k} < r_{att} \\ \dot{r}^{i,k} < 0: & r^{i,k} \geq r_{att} \end{cases} \tag{4.7}$$

A negative velocity indicates that the robots move toward each other (i.e., attract), while a positive velocity indicates that they move away from each other (i.e., repel).

For robot i, we define a social potential function (SPF) $g^{i,k}(r^{i,k}) : \mathbb{R}^2 \rightarrow \mathbb{R}$ as the following

$$g^{i,k} = \begin{cases} U_{rep}^{i,k} & r^{i,k} \leq r_{rep} \\ 0 & r_{rep} < r^{i,k} < r_{att} \\ U_{att}^{i,k} & r^{i,k} \geq r_{att} \end{cases} \tag{4.8}$$

where M is the number of robots in the swarm, $i,k = 1,\ldots,M$, $U_{rep}^{i,k}$ is repulsive component, $U_{att}^{i,k}$ is the attractive component, the free action component is zero, the superscripts on $r^{i,k}$ indicate that it operates on pairs of robots i and k (i.e., it is pairwise), and the radius of

repulsion r_{rep} and the radius of attraction r_{att} are the same for all robots, and $r_{rep} \leq r_{att}$. [7]. The repulsive and attractive components of $g^{i,k}$ are given by

$$U_{rep}^{i,k} = -\frac{1}{2} K_3 \left(r^{i,k}\right)^2$$
$$U_{att}^{i,k} = \frac{1}{2} K_1 \left(r^{i,k}\right)^2$$

The attractive component is a standard attractive function [5][6], and the repulsive function is the negated form of the attractive function.

The gradient of the SPF $\nabla g^{i,k}$

$$\nabla g^{i,k} = \begin{cases} \nabla U_{rep}^{i,k} & r^{i,k} \leq r_{rep} \\ 0 & r_{rep} < r^{i,k} < r_{att} \\ \nabla U_{att}^{i,k} & r^{i,k} \geq r_{att} \end{cases}$$

assigns to robot i the desired velocity from (4.7) by the *gradient descent method* [6]. In this method, we set the desired velocity of robot i equal to the negated gradient of the APF (i.e., $\dot{x}^{i,k} = -\nabla U^{i,k}$). Thus, the kinematics for robot i with respect to another robot k in the states of repulsion and attraction are respectively

$$\begin{aligned} \dot{x}^{i,k} &= -\nabla U_{rep}^{i,k} = K_3 \left(x^i - x^k\right) \\ \dot{x}^{i,k} &= -\nabla U_{att}^{i,k} = -K_1 \left(x^i - x^k\right) \end{aligned} \tag{4.9}$$

The SPF (4.8) defines the kinematics for robot i with respect to one other robot k. We get the overall kinematics $\dot{x}^i$ for robot i by summing up all of the individual $\dot{x}^{i,k}$ terms. Therefore, the overall kinematics for robot i are for repulsion

$$\dot{x}^i = \sum_{k \in S_{rep}^i} \left(-\nabla U_{rep}^{i,k}\right) = \sum_{k \in S_{rep}^i} \left[K_3 \left(x^i - x^k\right)\right] \tag{4.10}$$

and

$$\dot{x}^i = \sum_{k \in S_{att}^i} \left(-\nabla U_{att}^{i,k}\right) = \sum_{k \in S_{att}^i} \left[-K_1 \left(x^i - x^k\right)\right] \tag{4.11}$$

for attraction, where the *repulsive subswarm S^i_{rep} for robot i* is the subswarm of M^i_{rep} robots that includes all robots k such that $r^{i,k} = \left\|x^i - x^k\right\| \leq r_{rep}$ (but does not include robot i), and the *attractive subswarm S^i_{att} for robot i* is the subswarm of M^i_{att} robots that includes all robots k such that $r^{i,k} = \left\|x^i - x^k\right\| \geq r_{att}$ (but does not include robot i).

4.5.2 Definition of Repulsive and Attractive Behavior

We define *repulsive behavior* for robot i as follows: If the distance between robot i (at position x^i) and a reference position x^r is less than the *radius of influence for repulsion* r_{rep}, then the robot i moves away from the reference position x^r and toward the desired position x^i_d (i.e., $x^i \rightarrow x^i_d$) via a path that approximates a straight-line as much as possible (i.e., $\phi^{i,r} \rightarrow 0$); if it moves outside r_{rep}, then the behavior of robot i will be determined by other control goals. (Note that $x^i_d \neq x^r$ for repulsive behavior.) *Additive repulsive behavior* is the repulsive behavior of robot i when it has a repulsive subswarm and its velocity is defined by (4.10).

We define *attractive behavior* for robot i as follows: If the distance between robot i (at position x^i) and a reference position x^r is greater than the *radius of influence for attraction* r_{att}, then the robot i moves closer to the reference position x^r (i.e., $x^i \rightarrow x^r$and $r^i \rightarrow r^i_d$) via a path that approximates a straight-line as much as possible (i.e., $\phi^{i,r} \rightarrow 0$); if it moves inside r_{att}, then the behavior of robot i will be determined by other control goals. *Additive attractive behavior* is the attractive behavior of robot i when it has an attractive subswarm and its velocity is defined by (4.11).

4.5.3 Path Planning Model for Robot

We superimpose a component for controlling the robot's motion the in the free action state onto the social potential function in order to obtain the *path planning model* for robot i.

$$g^i = \left[g^i_{rep}\right]\sigma^i_1 + \left[g^i_{att}\right]\sigma^i_2 + \left[g^i_{free}\right]\sigma^i_3 \tag{4.12}$$

where the switching rule is given by

$$\left(\sigma^i_1, \sigma^i_2, \sigma^i_3\right) = \begin{cases} (1,0,0): & repulsion \\ (0,1,0): & attraction \\ (0,0,1): & free\ action \end{cases} \tag{4.13}$$

and the states of repulsion, attraction and free action for robot i are defined as follows:

- repulsion: $r^{i,k} \leq r_{rep}$ for any k.
- attraction: $r^{i,k} > r_{rep}$ for all k and $r^{i,k} \geq r_{att}$ for any k.
- free action: $r_{rep} < r^{i,k} < r_{att}$ for all k.

This switching rule uses *subsumption* by prioritizing the states. A state with a higher priority can override or subsume a state with a lower priority [24]. We give the state of repulsion the highest priority because we want to ensure collision avoidance. We give the state of attraction the second-highest priority because we deem the maintenance of the cohesiveness of the swarm to be more important than free action, which is also why the state of free action is given the lowest priority.

The equations (4.3), (4.7), (4.8), (4.10), (4.11), (4.12), and (4.13) define the desired behavior for the swarm and its members.

4.6 Target Evasion

4.6.1 Desired Position Formula

In order to implement repulsive behavior with the position tracking controller, we must derive a formula for calculating a desired position for the robot that is further away from a

reference position x^r than the robot's current position. The controller will drive the robot to this desired position, thereby moving it further way from x^r. We derive the desired position formula for the repulsive function by equating the gradients of the attractive function and the repulsive function and then solving for the the reference position of the gradient of the attractive function.

$$\dot{x}^i = -\nabla U_{att}^{i,d} = -\nabla U_{rep}^{i,r} \tag{4.14}$$

We set the gradients equal to each other ($\nabla U_{att}^{i,d} = \nabla U_{rep}^{i,r}$) in order to find the value of x_d^i that, when used in the attractive APF, will produce the same repulsive behavior that the repulsive APF produces.

$$K_1\left(x^i - x_d^i\right) = -K_3\left(x^i - x^r\right)$$

Next, we solve for x_d^i in order to get

$$x_d^i = x^i + \frac{K_3}{K_1}\left(x^i - x^r\right) \tag{4.15}$$

The main advantage of using this formula is that we can implement a repulsive APF by using the position tracking controller (4.5) in concert with this formula. Otherwise, we would have to use two separate controllers, one for attractive behavior and another for repulsive behavior.

4.6.2 Closed-Loop Kinematics

Using the position tracking controller (4.5), the closed-loop kinematics for additive repulsive behavior are

$$\begin{bmatrix} \dot{r}^i \\ \dot{\phi}^i \end{bmatrix} = \begin{bmatrix} -K_1 \cos^2(\phi^i)\, r^i \\ -K_2\, \phi^i \end{bmatrix} \tag{4.16}$$

For the lemma in the next subsection, we will need the closed-loop kinematics for the relative distance $r^{i,r} = \left\|x^i - x^r\right\|$ from robot i to the center of the repulsive subswarm. We

begin with the tracking coordinates in terms of the relative distance from the reference position.

$$\dot{r}^{i,r} = -v^i \cos(\phi^{i,r})$$

We use the the fact that $\cos\left(\phi^{i,r} + \pi\right) = \cos\left(\phi^i\right)$ and the trigonometric identity $\cos\left(\phi^{i,r} + \pi\right) = -\cos\left(\phi^{i,r}\right)$ in order to derive the relationship $\cos\left(\phi^{i,r}\right) = -\cos\left(\phi^i\right)$. Using this relation, we get

$$\dot{r}^{i,r} = v^i \cos(\phi^i)$$

We substitute the control law for the velocity into the above in order to get

$$\dot{r}^{i,r} = K_1 \cos^2(\phi^i)\, r^i$$

Using (4.15). we calculate $r^i = \left\|x^i - x_d^i\right\|$ as

$$r^i = \left\| x^i - \left[x^i + \frac{K_3}{K_1}\left(x^i - x^r\right)\right]\right\| = \frac{K_3}{K_1}\, r^{i,r}$$

This gives us the relationship between r^i and $r^{i,r}$ that we need in order to obtain the final closed-loop kinematics

$$\dot{r}^{i,r} = K_3 \cos^2(\phi^i)\, r^{i,r} \tag{4.17}$$

4.6.3 Target Evasion

Lemma 2: Using formula (4.15) in concert with the position tracking controller (4.5) causes robot i to move away from the reference position x^r.

Proof: The desired position x_d^i is defined by the formula (4.15). By Lemma 1, $r^i =$

$\left\|x^i - x_d^i\right\| \to 0$, $\phi^i \to 0$ as $t \to \infty$. The closed-loop kinematics (4.17) show that $r^{i,r}$ grows exponentially

$$r^{i,r}(t) = \left[e^{K_3 \int_0^t \cos^2(\phi^i)\, dt}\right] r^{i,r}(0)$$

as $t \to \infty$. This implies that robot i moves away from x^r as $t \to \infty$.

4.7 Preliminary Analysis of Additive Repulsive Behavior

In the case of additive repulsion, a robot i that has a repulsive subswarm will move away from the center of its repulsive subswarm. We exploit this fact in our method by using the position tracking controller to drive robot i away from the center of its repulsive subswarm in order to implement additive repulsive behavior.

4.7.1 Direction of Robot's Motion for Additive Repulsive Behavior

Lemma 3: For a robot i that has a repulsive subswarm S_{rep}^i with M_{rep}^i members whose kinematics are given by (4.10), we have that robot i moves away from the center of its repulsive subswarm $\bar{x}_{rep}^i$ where

$$\bar{x}_{rep}^i = \frac{1}{M_{rep}^i} \sum_{k \in S_{rep}^i} x^k \tag{4.18}$$

Proof: In the state of repulsion, the kinematics of robot i are

$$\dot{x}^i = K_3 \sum_{k \in S_{rep}^i} \left(x^i - x^k\right) \tag{4.19}$$

Using (4.18) we simplify (4.19) as follows

$$\dot{x}^i = K_3 \textstyle\sum_{k \in S_{rep}^i} \left(x^i - x^k\right)$$
$$\dot{x}^i = K_3 M_{rep}^i \left[\frac{1}{M_{rep}^i} \textstyle\sum_{k \in S_{rep}^i} x^i - \frac{1}{M_{rep}^i} \textstyle\sum_{k \in S_{att}^i} x^k\right]$$

Thus, (4.19) becomes

$$\dot{x}^i = K_3 M^i_{rep} \left(x^i - \bar{x}^i_{rep} \right) \tag{4.20}$$

Equation (4.20) tells us that robot i moves away from the center of its repulsive subswarm $\bar{x}^i_{rep}$.

Remark 2: If robot i is located at the center of its repulsive subswarm (i.e., $x^i = \bar{x}^i_{rep}$), then it will *not* move. However, all of the other robots k in the repulsive subswarm for robot i will move away from their corresponding $\bar{x}^k_{rep}$. The overall result of this is that all of the other robots k will move away from robot i.

4.7.2 Convergence to Non-Repulsion Region

We define the *state of the swarm* X as

$$X = \left[x^{1T} \; x^{2T} \; x^{3T} \ldots x^{(M-1)T} \; x^{MT} \right]^T \tag{4.21}$$

which is the combination of the states of each individual robot i. The following lemma tells us that the state of the swarm will converge to a global non-repulsion region Ω_{rep} while each individual robot i will converge to its own local non-repulsion region Ω^i_{rep}.

Lemma 4: If a swarm of robots uses the repulsive kinematics (4.10), then $X \rightarrow \Omega_{rep}$ as $t \rightarrow \infty$ where $\Omega_{rep} = \left\{ X : r^{i,k} > r_{rep} \; for \; all \; i,k \right\}$. Also, for all i, $x^i \rightarrow \Omega^i_{rep}$ as $t \rightarrow \infty$ where $\Omega^i_{rep} = \left\{ x^i : r^{i,k} > r_{rep} \; for \; all \; k \right\}$.

Proof: We use the Lyapunov-like function for $r^{i,k}$

$$V(X) = \sum_{i \in S_{rep}} \sum_{k \in S^i_{rep}} \frac{1}{2} \left[(r_{rep})^2 - \left(r^{i,k} \right)^2 \right]$$

where S_{rep} is the subswarm of all robots i that are in the state of repulsion. As the robots in the repulsive subswarm move away from each other and the relative distance between them increases above r_{rep}, then robots will leave the subswarm until there are no robots left in it.

The form of $V(X)$ has been chosen so that when robots leave a given repulsive subswarm, then they will not cause unwanted jumps in the value of $V(X)$.

The time derivative of $V(X)$ is

$$\dot{V} = \sum_{i \in S_{rep}} -\left[\nabla_{x^i} V\right]^T \dot{x}^i \tag{4.22}$$

where the gradient $\nabla_{x^i} V$ is derived as follows.

$$\frac{\partial V}{\partial x^j} = \sum_{k \in S^j_{rep}} \left(x^j - x^k\right) + \sum_{i:\, j \in S^i_{rep}} \left(x^j - x^i\right)$$

By substituting k for i in the second sum and using the fact that $i \in S^j_{rep}$ if and only if $j \in S^i_{rep}$, we get

$$\frac{\partial V}{\partial x^j} = 2 \sum_{k \in S^j_{rep}} \left(x^j - x^k\right) = 2M^j_{rep}\left(x^j - \bar{x}^j_{rep}\right)$$

Using this gradient, (4.22) becomes

$$\dot{V} = \sum_i -2M^i_{rep}\left(x^i - \bar{x}^i_{rep}\right)^T \dot{x}^i$$

By substituting (4.20) into the above equation, we get

$$\dot{V} = \sum_i -2K_3\left(M^i_{rep}\right)^2 \left\|x^i - \bar{x}^i_{rep}\right\|^2 < 0$$

if $M^i_{rep} > 0$ for any i. If $M^i_{rep} = 0$ for all i, then $V = 0$. Otherwise, $\dot{V}$ does not approach zero until $M^i_{rep} = 0$ for all i (there are no robots in the repulsive subswarm for robot i). From this, we conclude that $X \to \Omega_{rep}$ as $t \to \infty$. Furthermore, we conclude that $x^i \to \Omega^i_{rep}$ as $t \to \infty$.

Remark 3: Since $\dot{V}$ does not approach zero until $M^i_{rep} = 0$, $V \to 0$ in finite time. Thus,

each robot i will converge to Ω^i_{rep} in finite time.

4.8 Preliminary Analysis of Additive Attractive Behavior

In the case of additive attractive behavior, a robot i that has an attractive subswarm will move toward the center of that attractive subswarm. The fact simplifies our implementation of additive attraction. Instead of having to sum up a number of individual components, we can have the position tracking controller drive robot i toward the center of the attractive subswarm.

4.8.1 Direction of Robot's Motion for Additive Attractive Behavior

Lemma 5: For a robot i that has an attractive subswarm S^i_{att} with M^i_{att} members whose kinematics are given by (4.11), we have that robot i moves toward the center of its attractive subswarm $\bar{x}^i_{att}$ where

$$\bar{x}^i_{att} = \frac{1}{M^i_{att}} \sum_{k \in S^i_{att}} x^k \tag{4.23}$$

Proof: For the case of additive attraction using (4.11), the kinematics of robot i are

$$\dot{x}^i = \sum_{k \in S^i_{att}} \left[-K_1 \left(x^i - x^k \right) \right] \tag{4.24}$$

Using equation (4.23), we can simplify (4.24) as follows.

$$\dot{x}^i = -K_1 \textstyle\sum_{k \in S^i_{att}} \left(x^i - x^k \right)$$
$$\dot{x}^i = -K_1 M^i_{att} \left[\frac{1}{M^i_{att}} \textstyle\sum_{k \in S^i_{att}} x^i - \frac{1}{M^i_{att}} \sum_{k \in S^i_{att}} x^k \right]$$

Using this result, (4.24) becomes

$$\dot{x}^i = -K_1 M^i_{att} \left(x^i - \bar{x}^i_{att} \right) \tag{4.25}$$

From (4.25), we conclude that robot i moves toward $\bar{x}_{att}^{i}$ while we have at least one $r^{i,k} \geq r_{att}$ (i.e., $M_{att}^{i} \geq 1$).

Remark 4: If robot i is located at the center of the attractive subswarm (i.e., $x^{i} = \bar{x}_{att}^{i}$), then robot i will *not* move. However, all other robots k in the attractive subswarm for robot i move toward their corresponding $\bar{x}_{att}^{k}$. The overall result of this is that all of the other robots k will move closer to robot i.

4.8.2 Convergence to Non-Attraction Region

The following lemma tells us that the state will converge to a global non-attraction region Ω_{att} while each robot i will converge to its own local non-attraction region Ω_{att}^{i}

Lemma 6: If a swarm of robots uses the attractive kinematics (4.11), then $X \rightarrow \Omega_{att}$ as $t \rightarrow \infty$ where $\Omega_{att} = \left\{X : r^{i,k} < r_{att} \; for \; all \; i \; and \; all \; k\right\}$. Also, for all i, $x^{i} \rightarrow \Omega_{att}^{i}$ as $t \rightarrow \infty$ where $\Omega_{att}^{i} = \left\{x^{i} : r^{i,k} < r_{att} \; for \; all \; k\right\}$ for i.

Proof: We use the Lyapunov-like function

$$V(X) = \sum_{i \in S_{att}} \sum_{k \in S_{att}^{i}} \frac{1}{2} \left[\left(r^{i,k} \right)^{2} - \left(r_{att} \right)^{2} \right]$$

where S_{att} is the subswarm of all robots i that are in the state of attraction. As the robots in the attractive subswarm move closer to each other and the relative distance between them decreases below r_{att}, then robots will leave the subswarm until there are no robots left in it. The form of $V(X)$ has been chosen so that when robots leave a given attractive subswarm, then they will not cause unwanted jumps in the value of $V(X)$. The time derivative of $V(X)$ is

$$\dot{V} = \sum_{k \in S_{att}^{i}} \left[\nabla_{x^{i}} V \right]^{T} \dot{x}^{i} \tag{4.26}$$

where the gradient $\nabla_{x^i} V$ is derived as follows.

$$\frac{\partial V}{\partial x^j} = \sum_{k \in S^j_{att}} \left(x^j - x^k \right) + \sum_{i: j \in S^i_{att}} \left(x^j - x^i \right)$$

By substituting k for i in the second sum and using the fact that $i \in S^j_{att}$ if and only if $j \in S^i_{att}$, we get

$$\frac{\partial V}{\partial x^j} = 2 \sum_{k \in S^j_{att}} \left(x^j - x^k \right) = 2 M^j_{att} \left(x^j - \bar{x}^j_{att} \right)$$

Using this gradient, (4.26) becomes

$$\dot{V} = \sum_i -2 M^i_{att} \left(x^i - \bar{x}^i_{att} \right)^T \dot{x}^i$$

By substituting (4.25) into the above equation, we get

$$\dot{V} = \sum_i -2 K_1 \left(M^i_{att} \right)^2 \left\| x^i - \bar{x}^i_{att} \right\|^2 < 0$$

if $M^i_{att} > 0$ for any i. If $M^i_{att} = 0$ for all i, then $V = 0$. Otherwise, $\dot{V}$ does not approach zero until $M^i_{att} = 0$ for all i (there are no robots in the attractive subswarm for robot i). Therefore, we conclude that $X \to \Omega_{att}$ as $t \to \infty$. From this, we conclude that $x^i \to \Omega^i_{att}$ for all i as $t \to \infty$.

Remark 5: In this lemma and its proof, we have assumed that none of robots in the attractive subswarm for any robot i are in the state of repulsion. Due to the use of subsumption in the definition of the kinematics (4.12) and the switching rule (4.13), it is possible to have a robot k in a given attractive subswarm that is in the state of repulsion. Lemma 4 demonstrates that each robot i will eventually converge to its non-repulsion region and then switch into the state of attraction if it has an attractive subswarm. Also, since $\dot{V}$ does not approach zero until $M^i_{att} = 0$ for all i, $V \to 0$ in finite time. Thus, each robot i will converge

to Ω^i_{att} in finite time.

4.9 Control Strategy for Additive Repulsive Behavior

In order to implement additive repulsive behavior for robot i, we use the position tracking controller (4.5) with x^i_d defined by the formula (4.15) in which we define the reference position to be the center of the repulsive subswarm S^i_{rep} (i.e., $x^r = \bar{x}^i_{rep}$).

$$x^i_d = x^i + \frac{K_3}{K_1}\left(x^i - \bar{x}^i_{rep}\right)$$

This will cause robot i to move away from $\bar{x}^i_{rep}$.

4.10 Control Strategy for Additive Attractive Behavior

In order to implement additive attractive behavior for robot i, we use the position tracking controller (4.5) with the desired position x^i_d defined as the center of the attractive subswarm S^i_{att} (i.e., $x^i_d \equiv \bar{x}^i_{att}$). This will cause robot i to move toward $\bar{x}^i_{att}$.

4.11 Radius of Convergence Theorem

The following lemma tells us that each robot i in the swarm will converge within a maximum *possible* relative distance from the center of the swarm, which distance we call the *radius of convergence* r_{conv}.

Lemma 7: The *radius of convergence* r_{conv} for a M member subswarm S^M is given by the formula

$$r^{i,\bar{x}} \leq r_{conv} = \frac{(M-1)}{M} r_{sep} \tag{4.27}$$

where $r^{i,\bar{x}} = \left\|x^i - \bar{x}\right\|$ and $r_{sep} = \max\left\|x^i - x^k\right\|$ for all pairs of robots i and k in the the subswarm and $\bar{x}$ is the center of the swarm that is given by

$$\bar{x} = \frac{1}{M} \sum_{m=1}^{M} x^m$$

Proof: We substitute the formula for the center of the swarm $\bar{x} = \frac{1}{M} \sum_{m=1}^{M} x^m$ into the definition of $r^{i,\bar{x}}$ in order to get

$$r^{i,\bar{x}} = \left\| x^i - \bar{x} \right\| = \left\| x^i - \tfrac{1}{M} \textstyle\sum_{m=1}^{M} x^m \right\|$$
$$r^{i,\bar{x}} = \left\| \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \left(x^i - x^m \right) \right\| \leq \tfrac{1}{M} \textstyle\sum_{m=1}^{M} \left\| x^i - x^m \right\|$$

Since the term for $i = m$ in the right hand side of the equation above is zero, we get

$$r^{i,\bar{x}} \leq \frac{M-1}{M} r_{sep}$$

which establishes the relation (4.27).

4.12 Analysis of Additive Repulsive Behavior

In this section, we present an argument that our method properly implements the control strategy for additive repulsive behavior from Section 4.9. A rigorous proof of the convergence of each robot to its non-repulsion region is beyond the scope of this article.

We observe from the preliminary analysis of additive repulsive behavior in Section 4.7 that the most important aspect of the robot's motion is that it moves away from the center of its repulsive subswarm (per Lemma 3). The robot does not have to move in a straight line that intersects the center and the robot's initial position when it began its repulsive motion. Therefore, the robot can move along a curved path and still accomplish the task of moving away from the other robots in its repulsive subswarm. Also, the translational motion of the robot can stop momentarily as long as it resumes within some finite time period.

The motion of robot i that results from the use of the control strategy in Section 4.9

meets all of these criteria. The robot moves away from the center of its repulsive subswarm $\bar{x}^i_{rep}$, although it may not do so in a straight line. It may have to turn, during which time it may stop momentarily. Also, it may move backward for a period of time while it is turning away from $\bar{x}^i_{rep}$. However, the robot is always moving away from $\bar{x}^i_{rep}$, except for the short time that it stops during the turning maneuver.

Another fact to note is that the translational speed of the robot increases as it moves further away from that center which is what also happens in the ideal case. This can be seen by comparing the closed-loop kinematics (4.17) with the kinematics for the ideal case (4.20).

Lemma 4 tells us that $x \to \Omega_{rep}$ and $x^i \to \Omega^i_{rep}$ as $t \to \infty$ for robot i in the ideal case. Because the similarity between the motion of the robot in the ideal case and in our method, we expect that the same convergent behavior occurs with our method. Thus, we conclude that each robot i should converge to its non-repulsion region Ω^i_{rep}. Our simulations exhibit this behavior.

4.13 Analysis of Additive Attractive Behavior

In this section, we present an argument that our method properly implements the control strategy for additive repulsive behavior from Section 4.10. As in the case of additive repulsive behavior, a rigorous proof of the convergence of each robot to its non-repulsion region is beyond the scope of this article.

We observe from the preliminary analysis of additive attractive behavior in Section 4.8 that the most important aspect of the robot's motion is that it moves toward the center of its attractive subswarm $\bar{x}^i_{att}$ (per Lemma 5). As in the case of additive repulsive behavior, the robot does not have to move in a straight line and can stop momentarily.

Additionally, the translational speed of the robot should decrease as it gets closer to $\bar{x}^i_{att}$. This can be seen by comparing the closed-loop kinematics for the translational component

that are found by substituting (4.5) into (4.4)

$$\dot{r}^i = -K_1 \, r^i \cos^2(\phi^i)$$

with the kinematics for the ideal case (4.25).

The motion of robot i that results from the use of the control strategy in Section 4.10 meets all of these criteria.

Lemma 6 tells us that $x \rightarrow \Omega_{att}$ and $x^i \rightarrow \Omega^i_{att}$ as $t \rightarrow \infty$ for robot i in the ideal case. Because the similarity between the motion of the robot in the ideal case and in our method, we expect that the same convergent behavior occurs with our method. Thus, we conclude that each robot i should converge to its non-attraction region Ω^i_{att}. Our simulations exhibit this behavior. Furthermore, we conclude from Lemma 7 that each robot i should converge to a hyperball with radius r_{conv}

$$B_{r_{conv}} = \left\{ x^i : \left\| x^i - \bar{x} \right\| \leq r_{conv} \right\}$$

where $r_{conv} = \frac{M-1}{M} r_{att}$.

4.14 Analysis of Overall Behavior

We want the overall behavior of robot i to be such that it converges to the free action region

$$\Omega^i_{free} = \left\{ x^i : r_{rep} < r^{i,k} < r_{att} \; for \; all \; k \right\}$$

where it is outside the attraction range and repulsion range of all other robots k. The free action region Ω^i_{free} is the intersection of the desired regions for attraction Ω^i_{att} and repulsion Ω^i_{rep}.

In order to have flocking behavior, the free action must exist (i.e., $\Omega^i_{free} = \Omega^i_{att} \cap \Omega^i_{rep} \neq$

$\emptyset$). On the other hand, if the free action region does not exist (i.e., $\Omega^i_{free} = \Omega^i_{att} \cap \Omega^i_{rep} = \emptyset$), then proper flocking behavior is impossible. In this case, no robot will be able to converge to its free action region because the region simply does not exist. In this case, the robot will experience *interstate chattering* as it switches continuously between the states of attraction and repulsion.

The *local minima problem* occurs when the summation of the gradients of the potential functions produces a velocity $\dot{x}^i$ that is zero at some position other than the goal position, which causes robot i to get stuck at that position. Our method does not experience the local minima problem [5][6] because it does use the summations of multiple potentials in order to define the velocity of robot i. Instead, the method exploits certain mathematical properties of such summations in order to define a nonzero velocity for the robot at all times. The robot will always be able to move, even though it may not be able to engage in free action due to interstate chattering.

4.14.1 Convergence to Free Action Region

We now present an argument for the following claim: If its free action region exists, then robot i will converge to its free action region. That is, we will have $x^i \rightarrow \Omega^i_{free}$.

We assume that the free action region exists (i.e., $\Omega^i_{free} = \Omega^i_{att} \cap \Omega^i_{rep} \neq \emptyset$). Our system uses a subsumption architecture with states in the following hierarchy from highest to lowest priority: (1) repulsive state, (2) attractive state and (3) free action state.

Step 1: Robot i (in the state of repulsion) converges to the non-repulsion zone (i.e., $x^i \rightarrow \Omega^i_{rep}$), as described in Section 4.12.

Step 2: Robot i (in the state of attraction) converges to the non-attraction zone (as described in Section 4.13) and does not leave the non-repulsion zone. Thus, robot i converges to the intersection of the non-attraction and non-repulsion zones, which is the free action zone. (i.e., $x^i \rightarrow \Omega^i_{free} = \Omega^i_{att} \cap \Omega^i_{rep}$). Thus, robot i enters the free action zone, which is described in Step 3.

Step 3: Robot i (in the state of free action) is in the free action zone, in which it acts freely. If robot i leaves the free action region, then it will return either Step 1 or Step 2, and then it will eventually converge again to the free action region.

Therefore, we have that $x^i \rightarrow \Omega^i_{free}$ as $t \rightarrow \infty$.

If a disturbance causes robot i to leave Ω^i_{free} (i.e., it causes $x^i \notin \Omega^i_{free}$), then robot i returns to either Step 1 or Step 2, depending on whether we have either $x^i \notin \Omega^i_{rep}$ or $x^i \notin \Omega^i_{att}$(with $x^i \in \Omega^i_{rep}$). The overall result is that $x^i \rightarrow \Omega^i_{free}$ as $t \rightarrow \infty$.

4.14.2 Behavior in the Free Action Region

In the state of free action, the kinematics of robot i are

$$\dot{x}^i_{free} = f^i_{free}(t)$$

where $f^i_{free}(t)$ is an arbitrary function that defines the desired trajectory for robot i. Using this arbitrary function, we can command the robot to behavior in whatever way is necessary in order to accomplish a task. If the desired trajectory causes the robot to move inside the attraction range or repulsion range of another robot, then the robot will switch to either the state of attraction or repulsion, whichever is appropriate.

4.15 Behavioral State Residency Time

Basis behaviors [18] of a robot are the simplest, lowest-level behaviors that robot performs. By coordinating them properly, these basis behaviors can be used to create complex behaviors. In our swarm, the basis behaviors are *attraction*, *repulsion* and *free action*. The switching rule for our system coordinates these behaviors in order to produce flocking behavior.

For each *basis behavior* [18], there is a corresponding *behavioral state*. The *behavioral*

state residency time (BSRT) is the total time that a given robot is in a behavior state. Thus, the *behavioral state residency time* (BSRT) is a metric for measuring the amount of time that a robot spends in performing a given basis behavior. We use the BSRT to measure the amount of time that each robot spends in the state of free action, which is a measure of how much freedom of action the robot has. The more time that a robot spends in the state of free action, the greater the behavioral state residency time will be.

4.16 Simulation Results

Using Matlab, several simulations of a four-robot (4) swarm were performed in order to demonstrate the theory. These simulations show that the robots exhibited the desired flocking behavior.

In simulation 1a, robots 1, 2, 3 and 4 were started at positions (0,10), (10,0), (0,-10) and (-10,0) respectively, and all of the robots had an initial heading of 0° (all in the inertial reference frame). The radii of influence for attraction and repulsion were $r_{att} = 25$ and $r_{rep} = 15$, and the controller constants were $K_1 = 2$, $K_2 = 10$, $K_3 = 2$, and $K_{random} = 10$. The controller constant K_{random} is used in the equations

$$x^i_{1d}(t+1) = x^i_1(t) + N_{1,sign}\, K_{random}\, N_{1,var}$$
$$x^i_{2d}(t+1) = x^i_2(t) + N_{2,sign}\, K_{random}\, N_{2,var}$$

in order to simulate random motion. In this equation, K_{random} is used to determine the *maximum size* of the random step, $N_{1,sign} \in \{+1,-1\}$ and $N_{2,sign} \in \{+1,-1\}$ are random variables that are used to determine the *sign* of the x-component and y-component of the step, and $N_{1,var} \subset [0\ldots1]$ and $N_{2,var} \subset [0\ldots1]$ are random variables that serve as *scaling factors*. Therefore, both the magnitude and the direction of the motion in both the x-axis and y-axis directions are determined randomly. This process determines the desired position for robot i, then the position tracking controller drives the robot to that desired

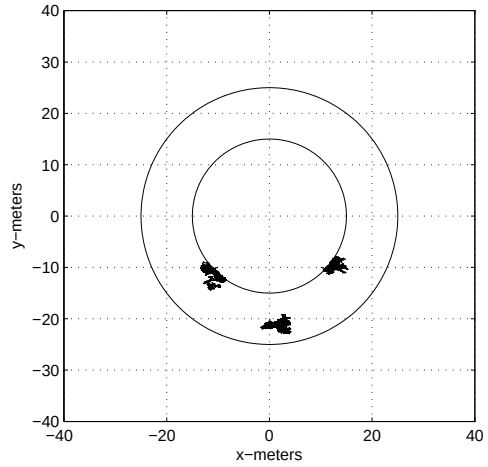

Figure 4.2: Simulation 1a: Robots 2, 3 and 4 Relative to Robot 1

position.

The results of *s*imulation 1a are presented as representative results in Figures (4.2) and (4.3). Figure (4.2) show the trajectories of robots 2, 3 and 4 relative to robot 1. This figure shows that all of the other robots stay outside of r_{rep} and inside of r_{att} for robot 1. Figure (4.3) show the trajectories of all four robots relative to the center of the swarm. This figure shows that the robots converge to within a radius of convergence that is $\frac{3}{4}r_{att}$.

In simulation 1b, robots 1, 2, 3 and 4 were started at positions (9,1), (10,0), (9,-1) and (-10,0) respectively, and all of the robots had an initial heading of 0° (all in the inertial reference frame). The radii of influence for attraction and repulsion were $r_{rep} = 25$ and $r_{att} = 15$, and the controller constants were $K_1 = 2$, $K_2 = 10$, $K_3 = 2$, and $K_{random} = 10$. In this simulation, we have the same convergence patterns as in Simulation 1a. The robots converge to the free action zone, and they arrange themselves in a more or less symmetrical pattern around the center of the swarm. Also, they converge to within $\frac{3}{4}r_{att}$ from the center of the swarm.

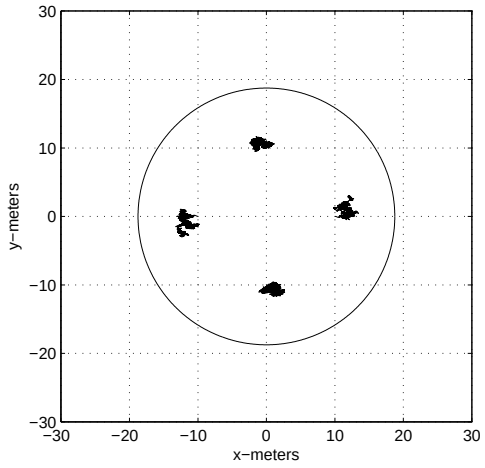

Figure 4.3: Simulation 1a: All Robots Relative to Swarm Center for All Time

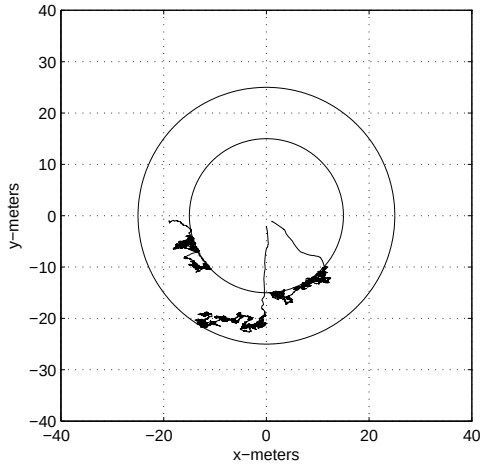

Figure 4.4: Simulation 1b: Robots 2, 3 and 4 Relative to Robot 1

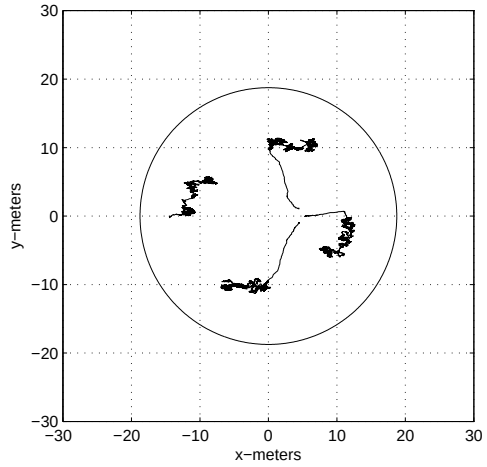

Figure 4.5: Simulation 1b: All Robots Relative to Swarm Center for All Time

In order to investigate the effect of varying the size of the free action area, a series of simulations was performed in which the radii of influence for attraction and repulsion were varied. Table 4.1 shows the different values for r_{att} and r_{rep} that were used for each simulation. The values for the controller constants and the starting positions and headings for the robots were the same as for simulation 1a. These results were found by averaging the results from ten separate simulation runs. The state residency times (in percentages of the total time of the simulation) for robot 1 for the simulations 1a-2d are given in Table 4.1.

Due to the fact that the width of the free wandering area varies from lesser to greater, one predicts that the amount of the time that the robots will spend in the free action state will gradually increase from simulation 2a to 2d. The state residency times do increase from simulation 2a to 2d, as the results in Table 4.1 show.

Table 4.2 shows the maximum and minimum values for the relative distance $r^{i,k}$ between robot 1 and the other three robots. The overall minimum or maximum is the value for the entire time of the simulation. The post-delay minimum or maximum is the value for all time after robot 1 has converged to the free action zone for the first time. These

Simulation	2a	2b	2c	2d
r_{att}	23	23.5	23.75	25
r_{rep}	17	16.5	16.25	15
Repulsion	79.54	64.35	33.19	11.11
Attraction	20.44	13.22	4.89	0.28
Free Action	0.02	22.43	61.92	88.61

Table 4.1: Behavioral State Residency Time for Robot 1

Simulation	2a	2b	2c	2d
r_{att}	23	23.5	23.75	25
r_{rep}	17	16.5	16.25	15
Overall Min	14.14	14.14	14.14	14.14
Post-Delay Min	16.56	16.08	15.84	14.54
Post-Delay Max	25.33	23.87	24.04	24.90
Overall Max	25.33	23.87	24.04	24.90

Table 4.2: Minimum and Maximum $r^{i,k}$ Values for Robot 1

results show that the robot does not always remain within the desired values for r_{att} and r_{rep}, which is rather hard to see from a figure such as Figure (4.2). As the free action zone gets larger, robot 1 is more and more able to stay within the desired values for r_{att} and r_{rep}.

Overall, the simulation results demonstrate that our method for implementing flocking in a swarm of mobile robots is successful.

4.17 Conclusion

In this article, we have presented an analysis of the behavior of the robots in a swarm that exhibits flocking behavior. We have successfully demonstrated that our method of implementing flocking behavior can guarantee collision avoidance, the maintenance of swarm cohesiveness, and freedom of action for each robot in the ideal case (a) and that we can guarantee, at a minimum, collision avoidance in the nonideal case (b). From this analysis, we conclude that our method is successful and effective.

Bibliography

[1] O. Khatib, "Real-Time Obstacle Avoidance For Manipulators and Mobile Robots," *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 500-505.

[2] J. Barraquand, B. Langlois, J.-C. Latombe, "Numerical Potential Field Techniques for Robot Path Planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, March/April 1992, pp. 224-241.

[3] J. Guldner and V. I. Utkin, "Sliding Mode Control for an Obstacle Avoidance Strategy Based On an Harmonic Potential Field," *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, December 1993.

[4] A. De Luca and G. Oriolo, "Local Incremental Planning for Nonholonomic Mobile Robots," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, vol. 1, pp. 104-110.

[5] J. C. Latombe, *Robot Motion Planning*, Boston: Kluwer Academic Publishers, 1991, pp. 295-297, 317-319.

[6] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, Cambridge, Massachusetts: The MIT Press, 2005, pp. 77-104.

[7] V. Gazi, and K. M. Passino, "Stability Analysis of Swarms," *IEEE Transactions on Automatic Control*, Vol. 48, No. 4, April 2003, pp. 692-697.

[8] T. Balch and R. C. Arkin, "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, December 1998.

[9] V. Gazi, and K. M. Passino, "A Class of Attraction/Repulsion Functions for Stable Swarm Aggregations," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, pp. 2842-2847.

[10] V. Gazi, and K. M. Passino, "Stability Analysis of Social Foraging Swarms: Combined Effects of Attractant/Repellent Profiles," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada, USA, December 2002, pp. 2848-2853.

[11] V. Gazi, and K. M. Passino, "Stability of Social Foraging Swarms," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 34, No. 1, February 2004, pp. 539-557.

[12] V. Gazi, "Swarm Aggregations Using Artificial Potentials and Sliding Mode Control," *Proceedings of the 42nd IEEE Conference on Decision and Control,* Maui, Hawaii, USA, December 2003, pp. 2041-2046.

[13] V. Gazi and R. Ordonez, "Target Tracking Using Artificial Potentials and Sliding Mode Control" *Proceedings of the 2004 American Control Conference,* Boston, Massachusetts*,* June 30 - July 2, 2004, pp 5588-5593.

[14] V. Gazi, "Swarm Aggregation Using Artificial Potentials and Sliding-Mode Control," *IEEE Transactions on Robotics*, Vol. 21, No.l 6, December 2005, pp. 1208-1214.

[15] R. C. Arkin, *Behavior-based Robotics*, Cambridge, Massachusetts: The MIT Press, 1999.

[16] K. Lerman and A. Galstyan, "A General Methodology for Mathematical Analysis of Multi-Agent Systems," *USC Information Sciences Technical Report ISI-TR-529*, 2001.

[17] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, Upper Saddle River, New Jersey: Prentice Hall, 1991, pp. 42-43, 276-307.

[18] K. Lerman, "Design and Mathematical Analysis of Agent-based Systems," in *Lecture Notes in Artificial Intelligence (LNAI) 1871*, p. 220 ff, Springer-Verlag, Berlin Heidelberg, 2001.

[19] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You, and S.-R. Oh, "A Stable Target-Tracking Control for Unicycle Mobile Robots," *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1822-1827.

[20] R. Seigwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Cambridge, Massachusetts: The MIT Press, 2004, pp. 84-85.

[21] Z. Li, Y. Jia, J. Du, and S. Yuan, "Flocking for Multi-agent Systems with Switching Topology in a Noisy Environment," *2008 American Control Conference*, Westin Seattle Hotel, Seattle, Washington, USA, June 11-13, 2008, pp. 111-116.

[22] A. Regmi, R. Sandoval, R. Byrne, H. Tanner, and C. T. Abdallah, "Experimental Implementation of Flocking Algorithms in Wheeled Mobile Robots," *2005 American Control Conference*, June 8-10, 2005, Portland, OR, USA, pp. 4917-4922.

[23] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, p. 108.

[24] R. R. Murphy, *Introduction to AI Robotics*, Cambridge, Massachusetts: The MIT Press, 2000, pp. 113-122.

Chapter 5

Conclusion

5.1 Further Research

The research presented in this dissertation does not exhaust all of the possible avenues of research that are possible in regard to the flocking behavior of robot swarms (or multi-robot teams). Further research is needed in this area in order to solve various problems and to gain new knowledge and insight into these robotic systems.

Our research is based on a purely kinematic model of a robot swarm. We neglected the effects of forces, masses and other dynamic variables. Further research is needed in order to investigate how our method works when one uses a true dynamic model of the swarm that takes force, friction, mass and other such dynamic variables into account. This will require the research to identify a proper controller for the robot's velocity. Our method can be used to determine the desired velocity for the robot (that is, the desired speed and heading for it), but a dynamic model requires that we have a controller that causes the robot to track the desired velocity properly. Such velocity controllers do exist for mobile robots, and, thus, there should be little or no trouble in finding an appropriate one for this line of research.

Our research investigated the case when the free action of the robots was limited to

simple, random motion. Further research is needed in order to determine how well our method works when one has the robots of the swarm perform more complex and more useful tasks. Such tasks may include searching for an object, moving objects around in the workspace and having two or more robots cooperate in some way in order to accomplish a complex task.

We were unable to perform experiments with real robots. Further research is needed in order to see how well our method performs for systems of real robots. Such research should uncover many hidden problems with the method, since all computer simulations must necessarily use simplified models of robots that neglect certain features that may prove to be more significant that one thought at first. Moreover, the ultimate test of our method, or any other method for coordinating multi-robot systems, is the test of how well it performs on real robots.

Further research is needed in order to develop better methods for implementing both the attractive artificial potential function (APF) and the repulsive APF. The repulsive APF has the problem of causing the robot to move at excessively high speeds if the robot gets too close to an obstacle or another robot. It is possible to use gain scheduling for both functions. In gain scheduling, the controller constants (i.e., K_1, K_2, K_3, r_{att} and r_{rep}) are switched between several possible values.

Further research is needed in order to investigate the effects of heterogeneity in the swarm. Such heterogeneity can result from differences in controller constants, differences in dynamic properties of the robots (such as different masses), from different kinds of robots and from many other variable factors. It is well known that all real robot systems are heterogeneous in nature, since it is impossible to make any two real robots be exactly the same in all their properties. Thus, heterogeneity is a property of all real robot swarms for which we must eventually account.

Further research is needed in order to find better and, hopefully, more effective and efficient methods for defining the control laws that implement either attractive or repulsive

behavior. One possibility is to develop a "avoidance controller" that would define a control law for avoiding an obstacle directly instead of having to use a formula for the desired position similar to the used in our method. Such a control law would reduce the amount of computation overhead involved in our method. We have already begun some research in this area, and we expect to publish the results from that result at a later date.

Our research uses a nonlinear position tracking controller as the cornerstone of its control strategy for the robot. Further research is needed in order to investigate what "plug-and-play" characteristics our method has, if any. For example, research is needed in order to determine whether a linear controller would work with our method. It may be possible to combine a linear controller with gain scheduling (of the controller constants) in order to cause the robot to engage properly in attractive and repulsive behavior. A linear controller may have certain advantages over a nonlinear controller for this application, and further research is needed in order to determine what those advantages might be.

Further research is needed in order to determine the effect of defining the social potential function (SPF) in various ways, especially ways other than that of our method. Also, further research is needed in order to investigate the case when additive attractive behavior and additive repulsive behavior are implemented by the summation of individual components instead of by defining the robot's motion relative to the center of either the attractive or repulsive subswarm. One will need to develop a new control law for implementing the attractive or repulsive behavior in that way. Our preliminary investigation of this method suggests that it is practicable, but rather complex in theory.

5.2 Overall Conclusion

This research project was conducted at The University of Alabama in the Department of Electrical and Computer Engineering from February 2004 to April 2009. The main result of this research project is a method for the coordination of multi-robotic systems that

produces flocking behavior in a swarm of differential-drive robots. Our approach to the coordination of multi-robotic systems is effective at maintaining the cohesiveness of the swarm, at preventing collisions between individual robots and at allowing each individual robot a degree of freedom of action. We have successfully developed methods for the analysis of robot swarms, including new metrics such as behavioral state residency time (BSRT). In addition to the analytical results, we have demonstrated the practicality and effectiveness of our method in Matlab simulations. In light of these results, we deem this research project to be a complete success.

Richard Patrick Samples

Tuscaloosa, Alabama

Thursday, April 23, 2009

Appendix A

Matlab M-Files for Article 1

A.1 Article 1 Simulator M-File

```
%-------------------------------------------------------------------------
% Richard Patrick Samples, Ph.D.(Cand.)
% PhD Dissertation Research
%
% PhD_One_Sim.m
% Simulator M-File for the "PhD-1" Article
%
% Last Updated: August 20, 2008
% Previous Updates:
%     August 19, 2008
%     July 25, 2008
%     July 23, 2008
% Electrical and Computer Engineering
% The University of Alabama, Tuscaloosa, Alabama
%
% This M-File is a simulation of a two-robot swarm of robots with
% a nonlinear tracking controller.
%
% Assumptions:
% (1) Synchronous - the robots move at the same time.
% (2) The robots are the same in all attributes (e.g., mass, speed,
% type, etc.).  That is, the swarm is homogeneous.
% (3) The robots use the same nonlinear tracking controller
%
% dx = v*cos(theta)
% dy = v*sin(theta)
% dtheta = omega
%
%
%
% Figure 1: Trajectory of the Swarm Center
% Figure 2: Robots 1 and 2 - Trajectory Relative to Swarm Center
% Figure 3: Robot 1 - Trajectory Relative to Robot 2
% Figure 4: Robot 2 - Trajectory Relative to Robot 1
% Figure 5: Behavioral State Residency Times
%

%
%-------------------------------------------------------------------------
% Clear all variables and figures and the screen to prepare.
clear
clc
close ALL
%
%-------------------------------------------------------------------------
```

```
% Initialize all variables and define constants.
T=0.01;          % time step
tmax=60;         % max time
nmax=tmax/T;     % calculates number of steps in simulation
t(1) = 0;
%-------------------------------------------------------------------------
% Initialize controller constants
K1 = 20;
K2 = 20;

% Set the desired separation and precision
zDesired = 50;
epsilonPrecision = 5;

% Constant that determines delta_xI and delta_yI in random wandering mode
Krandom = 5;

%-------------------------------------------------------------------------
% Initial Conditions (i.e., starting coordinates)
% Robot 1
xI(1,1)=20;
yI(1,1)=0;
thetaI(1,1) = 0;
state(1,1) = 2;

% Robot 2
xI(1,2)=0;
yI(1,2)=20;
thetaI(1,2) = 0;
state(1,2) = 2;

%-------------------------------------------------------------------------
% waitbar: Get handle and set up
execHandle = waitbar(0,'Executing Simulation');
%-------------------------------------------------------------------------



t(1) = 0;
% [Begin Main Simulation Loop]
for k = 1:nmax;
    t(k+1)=k*T;

    % Desired Position for Convergence for Robot 1
    xZ1 = xI(k,1)-xI(k,2);
    yZ1 = yI(k,1)-yI(k,2);
    psi_z1 = atan2(yZ1,xZ1);
    xIdConv1 = xI(k,2) + zDesired*cos(psi_z1);
    yIdConv1 = yI(k,2) + zDesired*sin(psi_z1);

    % Desired Position for Convergence for Robot 2
    xZ2 = xI(k,2)-xI(k,1);
    yZ2 = yI(k,2)-yI(k,1);
    psi_z2 = atan2(yZ2,xZ2);
    xIdConv2 = xI(k,1) + zDesired*cos(psi_z2);
    yIdConv2 = yI(k,1) + zDesired*sin(psi_z2);

    % Find Separation for Robot 1
    xnos1 = xI(k,1)-xIdConv1;
    ynos1 = yI(k,1)-yIdConv1;
    separation1 = sqrt( xnos1^2 + ynos1^2 );

    % Find Separation for Robot 2
    xnos2 = xI(k,2)-xIdConv2;
    ynos2 = yI(k,2)-yIdConv2;
    separation2 = sqrt( xnos2^2 + ynos2^2 );

    % State Selection Code For Robot 1
```

```
if (separation1 > epsilonPrecision)
    % Make the desired position the desired position for convergence
    xId1 = xIdConv1;
    yId1 = yIdConv1;

     % Update State of Robot 1
    state(k+1,1) = 1;

else  % Free Action; In this case, random motion
    % Determine a (pseudo-)random desired position by adding
    % a randomly selected delta value to the current position
    % in the Inertial Frame

    % rand returns a number between 0 and 1 (pseudo-random)
    % Determine Magnitude
    delta_xI = Krandom*rand;
    delta_yI = Krandom*rand;

    % Determine the sign (positive or negative); Default
    % value is positive
    if (rand < 0.5)
       delta_xI = -delta_xI;
    end

    if (rand < 0.5)
       delta_yI = -delta_yI;
    end

    % Determine the Free Action Goal Position
    xId1 = xI(k,1) + delta_xI;
    yId1 = yI(k,1) + delta_yI;

    % Update State of Robot 1
    state(k+1,1) = 2;

end % State Selection Code for Robot 1


% State Selection Code For Robot 2
if (separation2 > epsilonPrecision)
    % Make the desired position the desired position for convergence
    xId2 = xIdConv2;
    yId2 = yIdConv2;

     % Update State of Robot 2
    state(k+1,2) = 1;

else  % Free Action; In this case, random motion
    % Determine a (pseudo-)random desired position by adding
    % a randomly selected delta value to the current position
    % in the Inertial Frame

    % rand returns a number between 0 and 1 (pseudo-random)
    % Determine Magnitude
    delta_xI = Krandom*rand;
    delta_yI = Krandom*rand;

    % Determine the sign (positive or negative); Default
    % value is positive
    if (rand < 0.5)
       delta_xI = -delta_xI;
    end

    if (rand < 0.5)
       delta_yI = -delta_yI;
    end

    % Determine the Free Action Goal Position
```

```
    xId2 = xI(k,2) + delta_xI;
    yId2 = yI(k,2) + delta_yI;

    % Update State of Robot 2
    state(k+1,2) = 2;

end % State Selection Code for Robot 2

% Simulation Step Code for Robot 1

    % Calculate position of Robot 1 relative to the desired position
    xGrobot1 = xI(k,1) - xId1;
    yGrobot1 = yI(k,1) - yId1;

    % Calculate the tracking coordinates for the controller

    r_squared1 = xGrobot1^2 + yGrobot1^2;
    r1 = sqrt(r_squared1);

    psi_1 = atan2(yGrobot1, xGrobot1);
    phi_1 = pi + thetaI(k,1) - psi_1;

    % Control Law
    v1 = K1*r1*cos(phi_1);
    omega1 = -K1*sin(phi_1)*cos(phi_1)-K2*phi_1;

    % Robot Kinematics

    dx1 = v1*cos(thetaI(k,1));
    dy1 = v1*sin(thetaI(k,1));
    dtheta1 = omega1;

    % Euler's Method (Step by one time interval)
    xI(k+1,1) = xI(k,1) + dx1*T;
    yI(k+1,1) = yI(k,1) + dy1*T;
    thetaI(k+1,1) = thetaI(k,1) + dtheta1*T;

% End of Step Code for Robot 1

% Simulation Step Code for Robot 2

    % Calculate position of Robot 2 relative to the desired position
    xGrobot2 = xI(k,2) - xId2;
    yGrobot2 = yI(k,2) - yId2;

    % Calculate the tracking coordinates for the controller

    r_squared2 = xGrobot2^2 + yGrobot2^2;
    r2 = sqrt(r_squared2);

    psi_2 = atan2(yGrobot2, xGrobot2);
    phi_2 = pi + thetaI(k,2) - psi_2;

    % Control Law
    v2 = K1*r2*cos(phi_2);
    omega2 = -K1*sin(phi_2)*cos(phi_2)-K2*phi_2;

    % Robot Kinematics

    dx2 = v2*cos(thetaI(k,2));
    dy2 = v2*sin(thetaI(k,2));
    dtheta2 = omega2;

    % Euler's Method (Step by one time interval)
    xI(k+1,2) = xI(k,2) + dx2*T;
    yI(k+1,2) = yI(k,2) + dy2*T;
    thetaI(k+1,2) = thetaI(k,2) + dtheta2*T;
```

```
    % End of Step Code for Robot 2

    % waitbar: update waitbar with current time fraction
    waitbar(t(k)/tmax, execHandle);

end  % [Main Simulation Loop]


%-------------------------------------------------------------------------
% Plot the Results
%-------------------------------------------------------------------------
% Calculate the Swarm Center: xC, yC, using the mean.
for k = 1:nmax
    xC(k) = (xI(k,1) + xI(k,2))/2;
    yC(k) = (yI(k,1) + yI(k,2))/2;
end

% Calculate rMax and rMin for Analysis
rMax = zDesired + epsilonPrecision;
rMin = zDesired - epsilonPrecision;

%-------------------------------------------------------------------------

% Plot the trajectory of the Swarm Center
plot(xC,yC,'k')
%title('Trajectory of the Swarm Center')
axis square
xlabel('x-meters')
ylabel('y-meters')
grid on
%-------------------------------------------------------------------------

% Adjust Robot 1 trajectory to be relative to swarm center
for k = 1:nmax
    xIRC(k,1) = xI(k,1) - xC(k);
    yIRC(k,1) = yI(k,1) - yC(k);
end

% Adjust Robot 2 trajectory to be relative to swarm center
for k = 1:nmax
    xIRC(k,2) = xI(k,2) - xC(k);
    yIRC(k,2) = yI(k,2) - yC(k);
end

%-------------------------------------------------------------------------
% Robot 1 - Draw Circles for attraction and repulsion radii

% Radius of the hyperball is 1/2 the size of the atraction radius
% for this case.
figure
radius = (1/2)*rMax;
theta = linspace(0,2*pi,100);     % create vector theta
x = radius*cos(theta);            % generate x-coordinate
y = radius*sin(theta);            % generate y-coordinate
plot(x,y,'k-');                   % plot circle
hold on
% Plot second figure
plot(xIRC(:,1),yIRC(:,1),'k')
hold on
plot(xIRC(:,2),yIRC(:,2),'k')
%title('Robots 1 and 2 - Trajectory Relative to Swarm Center')
axis square
xlabel('x-meters')
ylabel('y-meters')
grid on


%-------------------------------------------------------------------------
```

```
% Create new figure
figure
% Adjust Robot 1 trajectory to be relative that of Robot 2
for k = 1:nmax
    xIROR(k,1) = xI(k,1) - xI(k,2);
    yIROR(k,1) = yI(k,1) - yI(k,2);
end
% Adjust Robot 2 trajectory to be relative to that of Robot 1
for k = 1:nmax
    xIROR(k,2) = xI(k,2) - xI(k,1);
    yIROR(k,2) = yI(k,2) - yI(k,1);
end
%-------------------------------------------------------------------------
% Robot 1 - Draw Circles for attraction and repulsion radii
radius = rMin;
theta = linspace(0,2*pi,100);     % create vector theta
x = radius*cos(theta);            % generate x-coordinate
y = radius*sin(theta);            % generate y-coordinate
plot(x,y,'k-');                   % plot circle
hold on
radius = rMax;
theta = linspace(0,2*pi,100);     % create vector theta
x = radius*cos(theta);            % generate x-coordinate
y = radius*sin(theta);            % generate y-coordinate
plot(x,y,'k-');                   % plot circle
hold on
% Plot second figure
plot(xIROR(:,1),yIROR(:,1),'k')
%title('Robot 1 - Trajectory Relative to Robot 2')
axis square
xlabel('x-meters')
ylabel('y-meters')
grid

%-------------------------------------------------------------------------
% Robot 2 - Draw Circles for attraction and repulsion radii
figure
radius = rMin;
theta = linspace(0,2*pi,100);     % create vector theta
x = radius*cos(theta);            % generate x-coordinate
y = radius*sin(theta);            % generate y-coordinate
plot(x,y,'k-');                   % plot circle
hold on
radius = rMax;
theta = linspace(0,2*pi,100);     % create vector theta
x = radius*cos(theta);            % generate x-coordinate
y = radius*sin(theta);            % generate y-coordinate
plot(x,y,'k-');                   % plot circle
hold on
% Plot second figure
plot(xIROR(:,2),yIROR(:,2),'k')
%title('Robot 2 - Trajectory Relative to Robot 1')
axis square
xlabel('x-meters')
ylabel('y-meters')
grid
hold on
%-------------------------------------------------------------------------
% Robot 1: Perform Behavioral State Residency Time Analysis
state_count = [0; 0];
for k=1:nmax
    if (state(k,1) == 1)
        state_count(1) = state_count(1) + 1;    % convergence
    else
        state_count(2) = state_count(2) + 1;    % free action
    end
end
state_total = state_count(1) + state_count(2);
```

```
state_1_R1 = num2str(100*state_count(1)/state_total);
state_2_R1 = num2str(100*state_count(2)/state_total);
stateC_1_R1 = num2str(state_count(1));
stateC_2_R1 = num2str(state_count(2));

%-------------------------------------------------------------------------
% Robot 2: Perform Behavioral State Residency Time Analysis
state_count = [0; 0];
for k=1:nmax
    if (state(k,2) == 1)
        state_count(1) = state_count(1) + 1;    % convergence
    else
        state_count(2) = state_count(2) + 1;    % free action
    end
end
state_total = state_count(1) + state_count(2);
state_1_R2 = num2str(100*state_count(1)/state_total);
state_2_R2 = num2str(100*state_count(2)/state_total);
stateC_1_R2 = num2str(state_count(1));
stateC_2_R2 = num2str(state_count(2));

figure
plot(0,0)
title('Robot 1 and 2 - Behavioral State Residency Times')

text(-0.8,0.6,['Robot 1 Residency Times   : '] )
text(-0.8,0.4,['state 1 (convergence) : ' stateC_1_R1 '   (' state_1_R1 ' %)'])
text(-0.8,0.2,['state 2 (free action) : ' stateC_2_R1 '   (' state_2_R1 ' %)'])

text(-0.8,-0.2,['Robot 2 Residency Times   : '] )
text(-0.8,-0.4,['state 1 (convergence) : ' stateC_1_R2 '   (' state_1_R2 ' %)'])
text(-0.8,-0.6,['state 2 (free action) : ' stateC_2_R2 '   (' state_2_R2 ' %)'])

%-------------------------------------------------------------------------
% Close the waitbar() window (clean-up)
close(execHandle);

%-------------------------------------------------------------------------
% End of Code
%-------------------------------------------------------------------------
```

A.2 Article 1 Simulation Driver M-File

```
%-------------------------------------------------------------------------
% Richard Patrick Samples, Ph.D.(Cand.)
% PhD Dissertation Research
%
% Sim_Driver.m
% This is a wrapper M-file that runs a series of simulations
% using the PhD_One_Sim.m M-File for the PhD-1 Article.
% Simuluation Series
%
%
% Last Updated: August 20, 2008
%
% Electrical and Computer Engineering
% The University of Alabama, Tuscaloosa, Alabama
%-------------------------------------------------------------------------


clc

numSims = 10;
Convergence(1:numSims) = 0;
FreeAction(1:numSims) = 0;
```

```
% Open file to append data
fid1 = fopen('PhD_One_Convergence.m','a');
fid2 = fopen('PhD_One_FreeAction.m','a');

fprintf(fid1,'1e (Convergence):');
fprintf(fid1,'\n');

fprintf(fid2,'1e (FreeAction):');
fprintf(fid2,'\n');


for i_sim = 1:numSims
    PhD_One_Sim

    fprintf(fid1,state_1_R1);
    fprintf(fid1,'\n');
    fprintf(fid2,state_2_R1);
    fprintf(fid2,'\n');
    %Convergence(i_sim,1) = ' ';
    %Convergence(i_sim,1) = state_1_R1(1,1)
    %FreeAction(i_sim,1) =  state_2_R1(1,1);

    close ALL
    close(execHandle)

end
```

A.3 Article 1 Social Potential Function Graph M-File

```
%-------------------------------------------------------------------------
% Richard Patrick Samples, PhD(Cand)
% PhD Dissertation Research
% PhD One Article
%
% Last Updated: August 20, 2008
%
% Electrical and Computer Engineering
% The University of Alabama, Tuscaloosa, Alabama
%
% This M-File graphs the social potential function (SPF) used in the PhD-1
% article.
%
%
%-------------------------------------------------------------------------

clc

Lr = 30;
La = 70;
Mag = 50;

x1 = [0 Lr];
x2 = [Lr La];
x3 = [La 100];
x4 = [Lr Lr];
x5 = [La La];
x6 = [Lr Lr];
x7 = [La La];
x8 = [0 100];

y1 = [Mag Mag];
y2 = [0   0];
y3 = [-Mag -Mag];
y4 = [Mag 0];
y5 = [0 -Mag];
```

```
y6 = [-125 125];
y7 = [-125 125];
y8 = [0 0];

%plot(x1,y1,'k')
%hold on
plot(x2,y2,'k')
hold on
%plot(x3,y3,'k')
hold on
plot(x4,y4,'k')
hold on
plot(x5,y5,'k')
hold on
plot(x6,y6,'k:')
hold on
plot(x7,y7,'k:')
hold on
plot(x8,y8,'k:')
hold on
axis([0 100 -125 125])
xlabel('Distance of Separation')
ylabel('G(*)')

text(10,40,'Repulsion')
text(40,40,'Free Action')
text(80,40,'Attraction')
text(31,-80,'Lr = 30')
text(58,-80,'La = 70')
text(45,-100,'K1 = 2.5')

% Left Half of Non-Zero Function
K1 = 2.5;
rLeft = linspace(0,30,1000);
drLeft = -K1*(rLeft-50);
plot(rLeft,drLeft,'k')
hold on

% Left Half of Non-Zero Function
K1 = 2.5;
rRight = linspace(70,100,1000);
drRight = -K1*(rRight-50);
plot(rRight,drRight,'k')
hold on
```

Appendix B

Matlab M-Files for Article 2

B.1 Attractive Function M-File

```
%-------------------------------------------------------------------------
% Richard Patrick Samples, Ph.D.(Cand.)
% PhD Dissertation Research
%
% PhD_Two_ATT_Sim.m
% Simulator M-File for the "PhD-2" Article
% Simulation for the Quadratic Attractive Function
%
% Last Updated: November 26, 2008
%
% Electrical and Computer Engineering
% The University of Alabama, Tuscalooa, Alabama
%
% This M-File is a simulation of a robot using the
% nonlinear tracking controller.  The plot is a "star diagram"
% that shows actual vs. ideal behavior.
%
%
% dx = v*cos(theta)
% dy = v*sin(theta)
% dtheta = omega
%
% Figure 1: "Star Diagram" of Robot Behavior
%
%
%
%-------------------------------------------------------------------------
% Clear all variables and figures and the screen to prepare.
clear
clc
close ALL

% Set up Initial Conditions vectors

xIC(1) = 1000;
yIC(1) = 0;

xIC(2) = 1000*cos(45*(pi/180));
yIC(2) = 1000*sin(45*(pi/180));

xIC(3) = 0;
yIC(3) = 1000;

xIC(4) = 1000*cos(135*(pi/180));
yIC(4) = 1000*sin(135*(pi/180));
```

```
xIC(5) = -1000;
yIC(5) = 0;

xIC(6) = 1000*cos(225*(pi/180));
yIC(6) = 1000*sin(225*(pi/180));

xIC(7) = 0;
yIC(7) = -1000;

xIC(8) = 1000*cos(315*(pi/180));
yIC(8) = 1000*sin(315*(pi/180));

xSave(1)= 0;
ySave(1) = 0;

for Wrap_k = 1:8;

%-------------------------------------------------------------------------
% Initialize all variables and define constants.
T=0.001;          % time step
tmax=5;         % max time
nmax=tmax/T;     % calculates number of steps in simulation
t(1) = 0;
%-------------------------------------------------------------------------
% Initialize controller constants
K1 = 1;
K2 = 5;

% Set the desired separation and precision

epsilonPrecision = 1;


%-------------------------------------------------------------------------
% Initial Conditions (i.e., starting coordinates)
% Robot 1: Ideal Robot
xI(1,1) = xIC(Wrap_k);
yI(1,1) = yIC(Wrap_k);
thetaI(1,1) = 0;
state(1,1) = 2;

% Robot 2: Real Robot (Using Tracking Controller)
xI(1,2) = xIC(Wrap_k);
yI(1,2) = yIC(Wrap_k);
thetaI(1,2) = 0;
state(1,2) = 2;

% Set the Desired Position
xIdConvBOTH = 0;
yIdConvBOTH = 0;

t(1) = 0;
% [Begin Main Simulation Loop]
for k = 1:nmax;
    t(k+1)=k*T;

    % Desired Position for Convergence for Robot 1
    xIdConv1 = xIdConvBOTH;
    yIdConv1 = yIdConvBOTH;

    % Desired Position for Convergence for Robot 2
    xIdConv2 = xIdConvBOTH;
    yIdConv2 = yIdConvBOTH;

    % Find Separation for Robot 1
    xnos1 = xI(k,1)-xIdConv1;
    ynos1 = yI(k,1)-yIdConv1;
```

```
separation1 = sqrt( xnos1^2 + ynos1^2 );

% Find Separation for Robot 2
xnos2 = xI(k,2)-xIdConv2;
ynos2 = yI(k,2)-yIdConv2;
separation2 = sqrt( xnos2^2 + ynos2^2 );

% State Selection Code For Robot 1
if (separation1 > epsilonPrecision)
    % Make the desired position the desired position for convergence
    xId1 = xIdConv1;
    yId1 = yIdConv1;

     % Update State of Robot 1
    state(k+1,1) = 1;

else  % No Motion; The robot will not move.

    % Determine the Free Action Goal Position
    xId1 = xI(k,1);
    yId1 = yI(k,1);

    % Update State of Robot 1
    state(k+1,1) = 2;

end % State Selection Code for Robot 1


% State Selection Code For Robot 2
if (separation2 > epsilonPrecision)
    % Make the desired position the desired position for convergence
    xId2 = xIdConv2;
    yId2 = yIdConv2;

     % Update State of Robot 2
    state(k+1,2) = 1;

else  % No Motion

    % Determine the Free Action Goal Position
    % The goal is the same as the actual current position
    xId2 = xI(k,2);
    yId2 = yI(k,2);

    % Update State of Robot 2
    state(k+1,2) = 2;

end % State Selection Code for Robot 2

% Simulation Step Code for Robot 1

    % vd = -K1(x-xd) => vd = -K1 r1

    % Calculate position of Robot 1 relative to the desired position
    xGrobot1 = xI(k,1) - xId1;
    yGrobot1 = yI(k,1) - yId1;

    % Calculate the tracking coordinates for the controller

    r_squared1 = xGrobot1^2 + yGrobot1^2;
    r1 = sqrt(r_squared1);

    thetaI_k = atan2(yGrobot1, xGrobot1);


    % Control Law
    v1 = -K1*r1;
```

```
        % Robot Kinematics

        dx1 = v1*cos(thetaI_k);
        dy1 = v1*sin(thetaI_k);


        % Euler's Method (Step by one time interval)
        xI(k+1,1) = xI(k,1) + dx1*T;
        yI(k+1,1) = yI(k,1) + dy1*T;
        thetaI(k+1,1) = thetaI_k;

    % End of Step Code for Robot 1

    % Simulation Step Code for Robot 2

        % Calculate position of Robot 2 relative to the desired position
        xGrobot2 = xI(k,2) - xId2;
        yGrobot2 = yI(k,2) - yId2;

        % Calculate the tracking coordinates for the controller

        r_squared2 = xGrobot2^2 + yGrobot2^2;
        r2 = sqrt(r_squared2);

        psi_2 = atan2(yGrobot2, xGrobot2);
        phi_2 = pi + thetaI(k,2) - psi_2;

        % Control Law
        v2 = K1*r2*cos(phi_2);
        omega2 = -K1*sin(phi_2)*cos(phi_2)-K2*phi_2;

        % Robot Kinematics

        dx2 = v2*cos(thetaI(k,2));
        dy2 = v2*sin(thetaI(k,2));
        dtheta2 = omega2;

        % Euler's Method (Step by one time interval)
        xI(k+1,2) = xI(k,2) + dx2*T;
        yI(k+1,2) = yI(k,2) + dy2*T;
        thetaI(k+1,2) = thetaI(k,2) + dtheta2*T;




end  % [Main Simulation Loop]

% Save Simulation Results for each Simulation Run
x1Save(:,Wrap_k) = xI(:,1);
y1Save(:,Wrap_k) = yI(:,1);

x2Save(:,Wrap_k) = xI(:,2);
y2Save(:,Wrap_k) = yI(:,2);

end % Wrap_k (Outer Wrapping Loop)




%-------------------------------------------------------------------------
% Plot the Results
%-------------------------------------------------------------------------
% Plot Robot 1 Trajectory
```

```
for Plot1_k = 1:8
    plot( x1Save(:,Plot1_k),y1Save(:,Plot1_k),'k--' );
    hold on
end


xlabel('X')
ylabel('Y')
%title('Ideal(Dotted Line) vs Actual (Solid Line) Robot Trajectory')


% Plot Robot 2 Trajectory

for Plot1_k = 1:8
    plot( x2Save(:,Plot1_k),y2Save(:,Plot1_k),'k' );
    hold on
end
```

```
%-------------------------------------------------------------------------
% End of Code
%-------------------------------------------------------------------------
```

B.2 Repulsive Function M-File

```
%-------------------------------------------------------------------------
% Richard Patrick Samples, Ph.D.(Cand.)
% PhD Dissertation Research
%
% PhD_Two_EAR_Sim.m
% Simulator M-File for the "PhD-2" Article
% Simulation for the Effective Attractive Function for Repulsion (EAR)
%
% Last Updated: November 26, 2008
%
%
% The University of Alabama, Tuscalooa, Alabama
%
% This M-File is a simulation of a robot using the
% nonlinear tracking controller.  The plot is a "star diagram"
% that shows actual vs. ideal behavior.
%
%
% dx = v*cos(theta)
% dy = v*sin(theta)
% dtheta = omega
%
% Figure 1: "Star Diagram" of Robot Behavior
%
%
%
%-------------------------------------------------------------------------
% Clear all variables and figures and the screen to prepare.
clear
clc
close ALL

% Set up Initial Conditions vectors

Rinitial = 1;
```

```
xIC(1) = Rinitial;
yIC(1) = 0;

xIC(2) = Rinitial*cos(45*(pi/180));
yIC(2) = Rinitial*sin(45*(pi/180));

xIC(3) = 0;
yIC(3) = Rinitial;

xIC(4) = Rinitial*cos(135*(pi/180));
yIC(4) = Rinitial*sin(135*(pi/180));

xIC(5) = -Rinitial;
yIC(5) = 0;

xIC(6) = Rinitial*cos(225*(pi/180));
yIC(6) = Rinitial*sin(225*(pi/180));

xIC(7) = 0;
yIC(7) = -Rinitial;

xIC(8) = Rinitial*cos(315*(pi/180));
yIC(8) = Rinitial*sin(315*(pi/180));

% Set up Goal Position Vector

Rgoal = 10;

xGPV(1) = Rgoal;
yGPV(1) = 0;

xGPV(2) = Rgoal*cos(45*(pi/180));
yGPV(2) = Rgoal*sin(45*(pi/180));

xGPV(3) = 0;
yGPV(3) = Rgoal;

xGPV(4) = Rgoal*cos(135*(pi/180));
yGPV(4) = Rgoal*sin(135*(pi/180));

xGPV(5) = -Rgoal;
yGPV(5) = 0;

xGPV(6) = Rgoal*cos(225*(pi/180));
yGPV(6) = Rgoal*sin(225*(pi/180));

xGPV(7) = 0;
yGPV(7) = -Rgoal;

xGPV(8) = Rgoal*cos(315*(pi/180));
yGPV(8) = Rgoal*sin(315*(pi/180));


for Wrap_k = 1:8;

    %-------------------------------------------------------------------------
    % Initialize all variables and define constants.
    T=0.01;          % time step
    tmax=5*20;         % max time
    nmax=tmax/T;     % calculates number of steps in simulation
    t(1) = 0;
    %-------------------------------------------------------------------------
    % Initialize controller constants
    K1 = 2;
    K2 = 10;
    K3 = 2;

    % Set the desired separation and precision
```

```
rRep = 5;


%-------------------------------------------------------------------------
% Initial Conditions (i.e., starting coordinates)
% Robot 1: Ideal Robot
xI(1,1) = xIC(Wrap_k);
yI(1,1) = yIC(Wrap_k);
thetaI(1,1) = 0;


% Robot 2: Real Robot (Using Tracking Controller)
xI(1,2) = xIC(Wrap_k);
yI(1,2) = yIC(Wrap_k);
thetaI(1,2) = 0;



t(1) = 0;
% [Begin Main Simulation Loop]
for k = 1:nmax;
    t(k+1)=k*T;

    % Desired Position is the Origin (0,0)
    xId = 0;
    yId = 0;


    % Find Separation for Robot 1
    xnos1 = xI(k,1)-xId;
    ynos1 = yI(k,1)-yId;
    separation1 = sqrt( xnos1^2 + ynos1^2 );

    % Find Separation for Robot 2
    xnos2 = xI(k,2)-xId;
    ynos2 = yI(k,2)-yId;
    separation2 = sqrt( xnos2^2 + ynos2^2 );

    % State Selection Code For Robot 1
    if (separation1 < rRep)


        % Direct simulation of Repulsive Function
        % vd = -K3*f3*r1

        % Calculate position of Robot 1 relative to the target position
        xGrobot1 = xI(k,1) - xId;
        yGrobot1 = yI(k,1) - yId;

        % Calculate the tracking coordinates for the controller

        r_squared1 = xGrobot1^2 + yGrobot1^2;
        r1 = sqrt(r_squared1);

        % Point the robot away from the target position
        thetaI_k = atan2(yGrobot1, xGrobot1);


        % Control Law
        f3 = ( (1/rRep) - (1/r1) )*(1/(r1^3));
        v1 = -K3*f3*r1;


        % Robot Kinematics
        dx1 = v1*cos(thetaI_k);
        dy1 = v1*sin(thetaI_k);
```

```
else  % No Motion; The robot will not move.

    % Set velocity to ZERO
    dx1 = 0;
    dy1 = 0;

    thetaI_k = 0;

end % State Selection Code for Robot 1

%-------------------------------------------------------------------%
% State Selection Code For Robot 2
%-------------------------------------------------------------------%
if (separation2 < rRep)


    % Effective Position for Repulsion
    % f3 = (1/rRep - 1/r)(1/r^3)
    % x_d_eff = x - (K3/K1)f3(x-xd) ==>
    %
    %     xGrobot2 = x - (K3/K1)f3(x - xd)
    %     yGrobot2 = y - (K3/K1)f3(y - yd)

    % r2 = Actual separation (x-xd)
    % r2eff = effective separation (x-xd_eff)

    rX2 = xI(k,2) - xId;
    rY2 = yI(k,2) - yId;

    r_squared2 = rX2^2 + rY2^2;
    r2 = sqrt(r_squared2);

    f3 = ( (1/rRep) - (1/r2) )*(1/r2^3);

    % Calculate distance relative to the goal position
    % xGrobot2eff = x - xdeff;
    % yGrobot2eff = y - ydeff;
    % xdeff = x - (K3/K1)f3(x-xd);

    xGrobot2eff = xI(k,2) - ( xI(k,2) - (K3/K1)*f3*(xI(k,2) - xId) );
    yGrobot2eff = yI(k,2) - ( yI(k,2) - (K3/K1)*f3*(yI(k,2) - yId) );

    xdeff(k,Wrap_k) =  -(K3/K1)*f3*(xI(k,2) - xId);
    ydeff(k,Wrap_k) =  -(K3/K1)*f3*(yI(k,2) - yId);

    % Calculate position of Robot 2 relative to the desired position
    % xGrobot2 = xI(k,2) - xId2;
    % yGrobot2 = yI(k,2) - yId2;

    % Calculate the tracking coordinates for the controller

    r_squared2eff = xGrobot2eff^2 + yGrobot2eff^2;
    r2eff = sqrt(r_squared2eff);

    psi_2 = atan2(yGrobot2eff, xGrobot2eff);
    phi_2 = pi + thetaI(k,2) - psi_2;

    % Control Law
    v2 = K1*r2eff*cos(phi_2);
    omega2 = -K1*sin(phi_2)*cos(phi_2)-K2*phi_2;

    % Robot Kinematics

    dx2 = v2*cos(thetaI(k,2));
    dy2 = v2*sin(thetaI(k,2));
    dtheta2 = omega2;
```

```
        else  % No Motion

            % Set the velocity to ZERO

            dx2 = 0;
            dy2 = 0;
            dtheta2 = 0;

        end % State Selection Code for Robot 2

        % Simulation Step Code for Robot 1

            % Euler's Method (Step by one time interval)
            xI(k+1,1) = xI(k,1) + dx1*T;
            yI(k+1,1) = yI(k,1) + dy1*T;
            thetaI(k+1,1) = thetaI_k;

        % End of Step Code for Robot 1

        % Simulation Step Code for Robot 2

            % Euler's Method (Step by one time interval)
            xI(k+1,2) = xI(k,2) + dx2*T;
            yI(k+1,2) = yI(k,2) + dy2*T;
            thetaI(k+1,2) = thetaI(k,2) + dtheta2*T;

        % End of Step Code for Robot 2


    end  % [Main Simulation Loop]

    % Save Simulation Results for each Simulation Run
    x1Save(:,Wrap_k) = xI(:,1);
    y1Save(:,Wrap_k) = yI(:,1);

    x2Save(:,Wrap_k) = xI(:,2);
    y2Save(:,Wrap_k) = yI(:,2);

end % Wrap_k (Outer Wrapping Loop)


%-------------------------------------------------------------------------
% Plot the Results
%-------------------------------------------------------------------------
% Plot Robot 1 Trajectory

for Plot1_k = 1:8
    plot( x1Save(:,Plot1_k),y1Save(:,Plot1_k),'k--' );

    hold on
end


xlabel('X')
ylabel('Y')
%title('Ideal(Dotted Line) vs Actual (Solid Line) Robot Trajectory')


% Plot Robot 2 Trajectory

for Plot1_k = 1:8
    plot( x2Save(:,Plot1_k),y2Save(:,Plot1_k),'k' );

    hold on
end

%axis([-3 3 -3 3])
```

```
%--------------------------------------------------------------------------
% End of Code
%--------------------------------------------------------------------------
```

B.3 Repulsive Function Graph M-File

```
%--------------------------------------------------------------------------
% Richard Patrick Samples, Ph.D.(Cand.)
% PhD Dissertation Research
%
% PhD-2 Article
% RepulsiveFcnPlot.m
%
% Last Updated: November 26, 2008
%
% Electrical and Computer Engineering
% The University of Alabama, Tuscalooa, Alabama
%--------------------------------------------------------------------------

clc
close ALL
clear
r = 1;
dr = 0.01;
k = 1;
r0 = 5;  % r0 is rRep
K3 = 2;  % K3 is nu in the standard function
while r <= r0

    Urep(k) = (1/2)*K3*( (1/r) - (1/r0) )^2;
    dUrep(k) = -K3*( (1/r0) - (1/r) )*(1/r^2);
    f3(k) = -K3*( (1/r0) - (1/r) )*(1/r^3);

    rV(k) = r;
    r = r + dr;
    k = k + 1;
end

subplot(2,1,1),plot(rV, dUrep,'k')
title('Plot of dUrep and Urep')
xlabel('r')
ylabel('dUrep')
grid on
axis([0 6 -0.1 2])

subplot(2,1,2),plot(rV,Urep,'k')
grid on
xlabel('r')
ylabel('Urep')
axis([0 6 -0.1 0.7])

figure
plot(rV,Urep,'k')
grid on
%title('Plot of F_3')
xlabel('r')
ylabel('f3')
axis([0 6 -0.1 0.7])
```

Appendix C

Matlab M-Files for Article 3

C.1 Article 3 Simulator Kernel M-File

```
%-------------------------------------------------------------------------%
% Richard Patrick Samples, Ph.D.(Cand.)                                   %
% Electrical and Computer Engineering                                     %
% The University of Alabama, Tuscalooa, Alabama                           %
%                                                                         %
% Copyright (C) 2009 Richard Patrick Samples                              %
%-------------------------------------------------------------------------%
%
% PhD Dissertation Research: Spring Semester 2009
%
% PhD_Three_Sim_Kernel.m
%
% Simulator M-File for the PhD-3 Article
%
% Simulation of a swarm of four (4) robots that use the coordination
% strategy described in the PhD-3 article.
%
% Last Updated:      30 March 2009
%
%
% Kinematics:
%        dx = v*cos(theta)
%        dy = v*sin(theta)
%        dtheta = omega
%
%
% Metrics:
%   Graphical Metric: Graph
%   Numerical Metric: Tables, other numerical data
%
% We seek to demonstrate three properties in the robot swarm:
%   1. Collision Avoidance
%   2. Maintenance of Cohesiveness of the Swarm
%   3. Freedom of Action for the Individual Robot
%
% Corresponding Metic(s):
%
%   Overall Behavior: Graph of Whole Swarm
%
%   1.  Concentric Circle Graph for Robot 1 (inner circle)
%       Table of minimum r_ik values for each Robot i
%
%   2.  Concentric Circle Graph for Robot 1 (outer circle)
%       Table of maximum r_ik values for each Robot i
%       Concentric Circle Graph for all Robots Relative to Center of Swarm
```

```
%
%   3.  Behavioral State Residency Time (BSRT) Analysis
%
%
%
% Figure 1: Graph of Whole Swarm
% Figure 2: Concentric Circle Graph (Robot 1 ONLY)
% Figure 3: Concentric Circle Grraph (All Robots Relative to Center)
% Figure 4: Table of Maximum and Minimum r_ik for each Robot i (note 1)
% Figure 5: BSRT Analysis for each Robot i
%
% Note 1: The Table will have both the overall min/max r_ik values plus
% the post-delay time values, which are the values for the robot after
% it has converged to the free action region for the first time in the
% simulation.
%
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%
%-------------------------------------------------------------------------
% Initialize all variables and define constants.
% Note: T (=dt or delta_t) must be very small for accuracy here.
% ==> T=0.002 is used as a benchmark.


T=0.01;         % time step
tmax=1*30;       % max time
nmax=tmax/T;     % calculates number of steps in simulation
t(1) = 0;

Tsampling = 0.01;       % Period for Sampling (Sampling Period)



%-------------------------------------------------------------------------
% Initialize controller constants
K1 = 2;
K2 = 10;
K3 = 2;
Krandom = 10;
rAttraction = 25;
rConvergence = (3/4)*rAttraction;
rRepulsion = 15;

% Initialize robot number variables
Robot_i = 1;
Robot_iMax = 4;

%-------------------------------------------------------------------------
% Initial Conditions (i.e., starting coordinates)
% Robot 1 (North)
xI(1,1)=9;
yI(1,1)=1;
thetaI(1,1) = 0;
v(1,1) = 0;
omega(1,1) = 0;

% Robot 2  (East)
xI(1,2)=10;
yI(1,2)=0;
thetaI(1,2) = 0;
v(1,2) = 0;
omega(1,2) = 0;

% Robot 3 (South)
xI(1,3)=9;
yI(1,3)=-1;
thetaI(1,3) = 0;
```

```
v(1,3) = 0;
omega(1,3) = 0;

% Robot 4 (West)
xI(1,4)=-10;
yI(1,4)=0;
thetaI(1,4) = 0;
v(1,4) = 0;
omega(1,4) = 0;


%-------------------------------------------------------------------------
% Repulsion and Attraction radii
% Robots 1 through 4
v_rep_save(1,Robot_i) = 0;

for Robot_i=1:Robot_iMax
    state(1,Robot_i) = 3;   % Initialize Behavioral state to
                            % 3=free wandering (2=attraction, 1=repulsion)
    %v_rep_save(1,Robot_i) = 0;
    %v_att_save(1,Robot_i) = 0;
    %v_free_save(1,Robot_i) = 0;
    %v_save(1,Robot_i) = 0;

end

for Robot_i = 1:Robot_iMax
    for Robot_k = 1:Robot_iMax
        r_ik_save(1,Robot_i,Robot_k) = 0;
    end
end



%-------------------------------------------------------------------------
% BEGIN --- MAIN SIMULATION LOOP
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% waitbar: Get handle and set up
execHandle = waitbar(0,'Executing Simulation');
%-------------------------------------------------------------------------

% t(k+1) with k=0,1,2,3,...

t(1) = 0;
for k = 1:nmax
    %
    % NOTE: The index k is the index to the matrices, while the time
    % vector t is for the time.  The index k must be an integer, while
    % the vector t's contents are real numbers (i.e., non-integers).
    %
    % Calulate the time for the index k
    t(k+1)=k*T;

    %-------------------------------------------------------------------------
    % BEGIN --- Inner Robot_i  Loop
    %-------------------------------------------------------------------------
    for Robot_i = 1:Robot_iMax

        %-------------------------------------------------------------
        % CALCULATE --- Centers of Repulsion and Attraction
        %               Numbers of Robots in Centers (Mrep,Matt)
        %-------------------------------------------------------------

        M_Repulsion_i(Robot_i) = 0;
        M_Attraction_i(Robot_i) = 0;
```

```
xICenterRepulsion(Robot_i) = xI(k,Robot_i);
yICenterRepulsion(Robot_i) = yI(k,Robot_i);

xICenterAttraction(Robot_i) = xI(k,Robot_i);
yICenterAttraction(Robot_i) = yI(k,Robot_i);


for Robot_k = 1:Robot_iMax
    if Robot_k ~= Robot_i

        % Calculate the separation between Robots i and k
        xnos = xI(k,Robot_k)-xI(k,Robot_i);
        ynos = yI(k,Robot_k)-yI(k,Robot_i);
        r_ik = sqrt( xnos^2 + ynos^2 );

        r_ik_save(k,Robot_i,Robot_k) = r_ik;

        if (r_ik <= rRepulsion)
            xICenterRepulsion(Robot_i) = xICenterRepulsion(Robot_i) + xI(k,Robot_k
                );
            yICenterRepulsion(Robot_i) = yICenterRepulsion(Robot_i) + yI(k,Robot_k
                );
            M_Repulsion_i(Robot_i) = M_Repulsion_i(Robot_i) + 1;
        end

        if (r_ik >= rAttraction)
            xICenterAttraction(Robot_i) = xICenterAttraction(Robot_i) + xI(k,
                Robot_k);
            yICenterAttraction(Robot_i) = yICenterAttraction(Robot_i) + yI(k,
                Robot_k);
            M_Attraction_i(Robot_i) = M_Attraction_i(Robot_i) + 1;
        end

    end
end

% Final Caclulation of Center of Repulsion
if (M_Repulsion_i(Robot_i) > 0)
    xICenterRepulsion(Robot_i) = xICenterRepulsion(Robot_i)/M_Repulsion_i(Robot_i)
        ;
    yICenterRepulsion(Robot_i) = yICenterRepulsion(Robot_i)/M_Repulsion_i(Robot_i)
        ;
end

% Final Calculation of Center of Attraction
if (M_Attraction_i(Robot_i) > 0)
    xICenterAttraction(Robot_i) = xICenterAttraction(Robot_i)/M_Attraction_i(
        Robot_i);
    yICenterAttraction(Robot_i) = yICenterAttraction(Robot_i)/M_Attraction_i(
        Robot_i);
end


%-------------------------------------------------------------------
% SAMPLE AND HOLD --- Only sample for data ever period Tsampling.
% Then, use the data to calculate new v,omega, theta and state.
%-------------------------------------------------------------------
if ( mod( t(k),Tsampling) == 0 )
    %---------------------------------------------------------------
    % SAMPLE --- Calculate new v, omega and state
    %---------------------------------------------------------------

    %---------------------------------------------------------------
    % STATE SELECTION (i.e., Select v and omega)
    %---------------------------------------------------------------
    if (M_Repulsion_i(Robot_i) > 0)
            % REPULSION
```

```
        % Calculate the separation between Robot i and
        % the center of repulsion (i.e., r_z)
        %   r_z = || x - xCenterRepulsion ||
        xnos_z = xI(k,Robot_i) - xICenterRepulsion(Robot_i);
        ynos_z = yI(k,Robot_i) - yICenterRepulsion(Robot_i);
        r_z(Robot_i) = sqrt(xnos_z^2 + ynos_z^2);



        % xdeff = x + (K3/K1)*(x-xCenterRepulsion)

        x_d_rep(k,Robot_i) = xI(k,Robot_i) + (K3/K1)*(xI(k,Robot_i) -
            xICenterRepulsion(Robot_i));
        y_d_rep(k,Robot_i) = yI(k,Robot_i) + (K3/K1)*(yI(k,Robot_i) -
            yICenterRepulsion(Robot_i));

        % Calculate the Tracking Coordinate for the Controller
        %   r_i, psi, phi
        %   r_i = || x - x_d_ref||
        xnos_i = xI(k,Robot_i) - x_d_rep(k,Robot_i);
        ynos_i = yI(k,Robot_i) - y_d_rep(k,Robot_i);
        r_i(Robot_i) = sqrt(xnos_i^2 + ynos_i^2);

        psi(Robot_i) = atan2(ynos_i, xnos_i);
        phi(Robot_i) = pi + thetaI(k,Robot_i) - psi(Robot_i);

        v(k+1,Robot_i) = K1*r_i(Robot_i)*cos(phi(Robot_i));
        omega(k+1,Robot_i) = -K1*sin(phi(Robot_i))*cos(phi(Robot_i))-K2*phi(
            Robot_i);

        % Set state to Repulsion
        state(k+1,Robot_i) = 1;

    elseif (M_Attraction_i(Robot_i) > 0)
        % ATTRACTION

        % r_i = || x - xCenterAttraction||
        xnos_i = xI(k,Robot_i) - xICenterAttraction(Robot_i);
        ynos_i = yI(k,Robot_i) - yICenterAttraction(Robot_i);
        r_i(Robot_i) = sqrt(xnos_i^2 + ynos_i^2);

        psi(Robot_i) = atan2(ynos_i, xnos_i);
        phi(Robot_i) = pi + thetaI(k,Robot_i) - psi(Robot_i);


        v(k+1,Robot_i) = K1*r_i(Robot_i)*cos(phi(Robot_i));
        omega(k+1,Robot_i) = -K1*sin(phi(Robot_i))*cos(phi(Robot_i))-K2*phi(
            Robot_i);


        % Set state to Attraction
        state(k+1,Robot_i) = 2;

    else
        % FREE ACTION
        % Determine a (pseudo-)random desired position by adding
        % a randomly selected delta value to the current position
        % in the Inertial Frame

        % rand returns a number between 0 and 1 (pseudo-random)
        % Determine Magnitude
        delta_xI = Krandom*rand;
        delta_yI = Krandom*rand;

        % Determine the sign (positive or negative); Default
        % value is positive
        if (rand < 0.5)
            delta_xI = -delta_xI;
```

```
                    end

                    if (rand < 0.5)
                        delta_yI = -delta_yI;
                    end

                    xnos_i = delta_xI;
                    ynos_i = delta_yI;
                    r_i(Robot_i) = sqrt(xnos_i^2 + ynos_i^2);

                    psi(Robot_i) = atan2(ynos_i, xnos_i);
                    phi(Robot_i) = pi + thetaI(k,Robot_i) - psi(Robot_i);

                    % Control Law
                    v(k+1,Robot_i) = K1*r_i(Robot_i)*cos(phi(Robot_i));
                    omega(k+1,Robot_i) = -K1*sin(phi(Robot_i))*cos(phi(Robot_i))-K2*phi(
                        Robot_i);


                    % Set State to Free Action
                    state(k+1,Robot_i) = 3;
                end
            else
                %-------------------------------------------------------------------
                % HOLD -- Hold values of v, omega and state for k+1
                % (That is, the (k+1) values are the same as the k values.
                %-------------------------------------------------------------------
                v(k+1,Robot_i) = v(k,Robot_i);
                omega(k+1,Robot_i) = omega(k,Robot_i);
                state(k+1,Robot_i) = state(k,Robot_i) ;
            end
            % END of Sample and Hold

            %-------------------------------------------------------------------
            % CALCULATE -- Robot Kinematics
            %
            % PERFORM -- One Time Step Using Euler's Method)
            %
            % NOTE: Step thetaI first, because xI and yI depend on it.
            %-------------------------------------------------------------------
            dtheta(Robot_i) = omega(k+1,Robot_i);
            thetaI(k+1,Robot_i) = thetaI(k,Robot_i) + dtheta(Robot_i)*T;

            dx(Robot_i) = v(k+1,Robot_i)*cos(thetaI(k+1,Robot_i));
            dy(Robot_i) = v(k+1,Robot_i)*sin(thetaI(k+1,Robot_i));

            xI(k+1,Robot_i) = xI(k,Robot_i) + dx(Robot_i)*T;
            yI(k+1,Robot_i) = yI(k,Robot_i) + dy(Robot_i)*T;


            %-------------------------------------------------------------------
            % END -- Inner Robot_i  Loop
            %-------------------------------------------------------------------
        end

        % waitbar: update waitbar with current time fraction
        waitbar(t(k)/tmax, execHandle);

    end
    %-----------------------------------------------------------------------
    % END -- MAIN SIMULATION LOOP
    %-----------------------------------------------------------------------



    %-----------------------------------------------------------------------
    % BEGIN -- DATA PROCESSING CODE
    %-----------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% Adjust each robot's trajectory to make it the center of a plot so
% that we can see if the other robots remain within attraction and
% repulsion radii of the the given robot.
%-------------------------------------------------------------------------
for Robot_i = 1:Robot_iMax
   for nOtherRobot = 1:Robot_iMax
      if (nOtherRobot ~= Robot_i)
        for k = 1:nmax
            xIRCa(k,Robot_i,nOtherRobot) = xI(k,nOtherRobot) - xI(k,Robot_i);
            yIRCa(k,Robot_i,nOtherRobot) = yI(k,nOtherRobot) - yI(k,Robot_i);
        end
      end
   end
end


%-------------------------------------------------------------------------
% For each Robot - Draw Circles for attraction and repulsion radii
% then plot the trajectories of the other robots relative to that robot
%-------------------------------------------------------------------------
for Robot_i=1:Robot_iMax
    figure
    radius = rRepulsion;
    theta = linspace(0,2*pi,1000);     % create vector theta
    x = radius*cos(theta);             % generate x-coordinate
    y = radius*sin(theta);             % generate y-coordinate
    plot(x,y,'k-');                    % plot circle
    hold on
    radius = rAttraction;
    theta = linspace(0,2*pi,1000);     % create vector theta
    x = radius*cos(theta);             % generate x-coordinate
    y = radius*sin(theta);             % generate y-coordinate
    plot(x,y,'k-');                    % plot circle
    hold on
    % Plot Robot Trajectories on the figure
    nRobotStr = num2str(Robot_i);
    %title(['Other Robot Trajectores Relative to Robot ' nRobotStr])
    axis square
    xlabel('x-meters')
    ylabel('y-meters')
    grid on
    for nOtherRobot = 1:Robot_iMax
        if (nOtherRobot ~= Robot_i)
            % Color Code the Plots for the Four Robots:
            % Robot 1: Red 'r'
            % Robot 2: Blue 'b'
            % Robot 3: Green 'g'
            % Robot 4: Cyan 'c'
            % For IEEE Journals: All Robots i: Black 'r'
            if (nOtherRobot == 1)
                color = 'k';
            elseif (nOtherRobot == 2)
                color = 'k';
            elseif (nOtherRobot == 3)
                color = 'k';
            elseif (nOtherRobot == 4)
                color = 'k';
            else
                color = 'k';
            end
            plot(xIRCa(:,Robot_i,nOtherRobot),yIRCa(:,Robot_i,nOtherRobot),color)
        end
    end


    axis([-40 40 -40 40])
```

```
    axis square

end
%-------------------------------------------------------------------------
% Plot Trajectories of all robots in Inertial Reference Frame.
figure
for Robot_i=1:Robot_iMax

    % Color Code the Plots for the Four Robots:
    % Robot 1: red
    % Robot 2: blue
    % Robot 3: green
    % Robot 4: cyan
    if (Robot_i == 1)
        color = 'r';
    elseif (Robot_i == 2)
        color = 'b';
    elseif (Robot_i == 3)
        color = 'g';
    elseif (Robot_i == 4)
        color = 'c';
    else
        color = 'k';
    end

    plot(xI(:,Robot_i),yI(:,Robot_i),color);
    hold on
end
%title('All Robots in Inertial Reference Frame')
xlabel('x-meters')
ylabel('y-meters')
grid on
axis([-30 30 -30 30])
axis square
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% PLOT -- All Robots Relative to Swarm Center
%
% Plot the trajectories of the robots relative to the center of the
% swarm to show convergence within a hyperball (here, a circle).
%-------------------------------------------------------------------------

% Calculate the Center of the Swarm for each step k
% (For eack step k, find the mean for x and y)
for k = 1:nmax

    % Initialize to zero to prepare for summing.
    xICenter(k) = 0;
    yICenter(k) = 0;

    for Robot_i = 1:Robot_iMax
        % Sum up the x and y coordinates of each Robot i
        xICenter(k) = xICenter(k) + xI(k,Robot_i);
        yICenter(k) = yICenter(k) + yI(k,Robot_i);
    end

    % Find final mean, which is the coordinate of the center
    xICenter(k) = xICenter(k)/Robot_iMax;
    yICenter(k) = yICenter(k)/Robot_iMax;

end


% Adjust each robot's trajectory to make the Center of the Swarm the
% center of the graph.  This will show convergence.
for Robot_i = 1:Robot_iMax
    for k = 1:nmax
```

```
        xIRC(k,Robot_i) = xI(k,Robot_i) - xICenter(k);
        yIRC(k,Robot_i) = yI(k,Robot_i) - yICenter(k);
    end
end

% Plot the trajectories of all of the robots relative to the center.
% Then, plot the circle of convergence to show convergence.
figure

% Plot the circle of convergence; Its radius is 1/2 Ratt
radius = rConvergence;
theta = linspace(0,2*pi,1000);    % create vector theta
x = radius*cos(theta);            % generate x-coordinate
y = radius*sin(theta);            % generate y-coordinate
plot(x,y,'k-');                   % plot circle
hold on

for Robot_i=1:Robot_iMax

    % Color Code the Plots for the Four Robots:
    % Robot 1: red 'r'
    % Robot 2: blue 'b'
    % Robot 3: green 'g'
    % Robot 4: cyan 'c'
    % For IEEE Journals: All Robots i: Black 'k'
    if (Robot_i == 1)
        color = 'k';
    elseif (Robot_i == 2)
        color = 'k';
    elseif (Robot_i == 3)
        color = 'k';
    elseif (Robot_i == 4)
        color = 'k';
    else
        color = 'k';
    end

    plot(xIRC(:,Robot_i),yIRC(:,Robot_i),color);
    hold on
end
%title('All Robots Relative to the Center of the Swarm')
xlabel('x-meters')
ylabel('y-meters')
grid on
axis([-30 30 -30 30])
axis square


%-------------------------------------------------------------------------
% Robot i: Perform State Residency Time Analysis
% The State Residency Time is the time the robot spends in a
% given behavioral state (1=repulsion,2=attraction,3=free wandering)
%
% The longer a robot is in the state 3=free wandering, the more
% freedom of action it has.
%-------------------------------------------------------------------------

for Robot_i=1:Robot_iMax
    state_count = [0; 0; 0];
    for k=1:nmax
        if (state(k,Robot_i) == 1)
            state_count(1) = state_count(1) + 1;    % repulsion
        elseif (state(k,Robot_i) == 2)
            state_count(2) = state_count(2) + 1;    % attraction
        else
            state_count(3) = state_count(3) + 1;    % free wandering
        end
    end
```

```
    state_total = state_count(1) + state_count(2) + state_count(3);
    state_1 = num2str(100*state_count(1)/state_total);
    state_2 = num2str(100*state_count(2)/state_total);
    state_3 = num2str(100*state_count(3)/state_total);

    state_ik(Robot_i,1) = 100*state_count(1)/state_total;
    state_ik(Robot_i,2) = 100*state_count(2)/state_total;
    state_ik(Robot_i,3) = 100*state_count(3)/state_total;

    state_ikC(Robot_i,1) = state_count(1);
    state_ikC(Robot_i,2) = state_count(2);
    state_ikC(Robot_i,3) = state_count(3);


    stateC_1 = num2str(state_count(1));
    stateC_2 = num2str(state_count(2));
    stateC_3 = num2str(state_count(3));
    nRobotStr = num2str(Robot_i);


end

disp(' ')
disp('State Residency Times (%): 1--2--3')
for Robot_i=1:Robot_iMax
    Robot_i_str = num2str(Robot_i);
    state_1_str = num2str( state_ik(Robot_i,1) );
    state_2_str = num2str( state_ik(Robot_i,2) );
    state_3_str = num2str( state_ik(Robot_i,3) );

    disp(['Robot ' Robot_i_str ':   ' state_1_str '   ' state_2_str '   ' state_3_str]);

end

disp(' ')
disp('State Residency Counts(#): 1--2--3')
state_total_str = num2str( state_total);
disp(['State Total = ' state_total_str])
for Robot_i=1:Robot_iMax
    Robot_i_str = num2str(Robot_i);
    state_1C_str = num2str( state_ikC(Robot_i,1) );
    state_2C_str = num2str( state_ikC(Robot_i,2) );
    state_3C_str = num2str( state_ikC(Robot_i,3) );

    disp(['Robot ' Robot_i_str ':   ' state_1C_str '   ' state_2C_str '   ' state_3C_str])
        ;

end



%-------------------------------------------------------------------------
% Find the value of k (index) for the First Convergence
% state(k, Roobot_i)
%-------------------------------------------------------------------------

first_convergence_k(Robot_i) = 1;
for Robot_i=1:Robot_iMax

    not_yet_found = 1;

    % Initiatlize k = 2 because state(1,Robot_i) is always initialized
    % to "3" by the program.
    k = 2;
    while ( not_yet_found && (k <= length(state(:,Robot_i))) )
        if (state(k,Robot_i) == 3)
            first_convergence_k(Robot_i) = k;
            not_yet_found = 0;
```

```
        end
        k = k + 1;
    end

end

%-------------------------------------------------------------------------
% Calculate Max/Min Chart for r_ik from r_ik_save matrix.
% This is a way to verify that each robot has avoided colliding with
% other robots.
%-------------------------------------------------------------------------

for Robot_i=1:Robot_iMax
    for Robot_k=1:Robot_iMax
        r_ik_max_chart(Robot_i,Robot_k) = max( r_ik_save(:,Robot_i,Robot_k) );
        r_ik_min_chart(Robot_i,Robot_k) = min( r_ik_save(:,Robot_i,Robot_k ));
    end
end

% Find the minumum r_ik
for Robot_i=1:Robot_iMax

    for Robot_x=1:Robot_iMax
        % Replace the zero ("0") value with the non-zero max value
        if (Robot_x == Robot_i)
            r_ik_min_chart(Robot_i,Robot_x) = max( r_ik_min_chart(Robot_i,:) );
        end
    end

    % Find the min r_ik
    r_ik_overall_min(Robot_i) = min( r_ik_min_chart(Robot_i,:) );

    % Find the max r_ik
    r_ik_overall_max(Robot_i) = max( r_ik_max_chart(Robot_i,:) );
end


%-------------------------------------------------------------------------
% Find the post-initial-convergence values for max/min r_ik
%-------------------------------------------------------------------------

% Calculate a chart of the max and min r_ik for each Robot i.  This
% covers all states k for each pair of Robots i and k.

% Set variables for the values of k for first convergence to free action
% regiona and for the length of the matrix (max k value).
k_begin = first_convergence_k(Robot_i);
k_end = length( r_ik_save(:,1,1) );

% Calulate the chart
for Robot_i=1:Robot_iMax
    for Robot_k=1:Robot_iMax
        r_ik_max_delay_chart(Robot_i,Robot_k) = max( r_ik_save(k_begin:k_end,Robot_i,
            Robot_k) );
        r_ik_min_delay_chart(Robot_i,Robot_k) = min( r_ik_save(k_begin:k_end,Robot_i,
            Robot_k ));
    end
end

% Find the min and max r_ik values.
for Robot_i=1:Robot_iMax

    for Robot_x=1:Robot_iMax
        % Replace the zero ("0") value with the non-zero max value
        if (Robot_x == Robot_i)
            r_ik_min_delay_chart(Robot_i,Robot_x) = max( r_ik_min_delay_chart(Robot_i,:) )
                ;
        end
```

```
    end

    % "first_convergence_k(Robot_i)" gives the value of k when the Robot i
    % first converges to the free action region.

    % Find the min r_ik
    r_ik_delay_min(Robot_i) = min( r_ik_min_delay_chart(Robot_i,:) );

    % Find the max r_ik
    r_ik_delay_max(Robot_i) = max( r_ik_max_delay_chart(Robot_i,:) );
end

%-------------------------------------------------------------------------
% Display the Table of max/min r_ik values
%-------------------------------------------------------------------------


disp(' ')
disp('Min/Max r_ik Values:')
Ratt_str = num2str(rAttraction);
Rrep_str = num2str(rRepulsion);
disp(['Ratt = ' Ratt_str])
disp(['Rrep = ' Rrep_str])
for Robot_i=1:Robot_iMax
   Robot_i_str = num2str(Robot_i);
   outer_min = num2str(r_ik_overall_min(Robot_i));
   outer_max = num2str(r_ik_overall_max(Robot_i));
   inner_min = num2str(r_ik_delay_min(Robot_i));
   inner_max = num2str(r_ik_delay_max(Robot_i));
   disp(['Robot ' Robot_i_str ':   ' outer_min '   ' inner_min '   ' inner_max '   '
       outer_max])
end


%-------------------------------------------------------------------------
% Close the waitbar() window (clean-up)
%close(execHandle);


%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% END OF PhD_Four_Sim.m
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
```

C.2 Article 3 Simulator Shell M-File

```
%-------------------------------------------------------------------------%
% Richard Patrick Samples, Ph.D.(Cand.)                                   %
% PhD Dissertation Research                                               %
%                                                                         %
% Copyright (C) 2009 Richard Patrick Samples                              %
%-------------------------------------------------------------------------%
%
% PhD_Four_Sim_Shell.m
%
% This is a wrapper (or shell) M-file that runs a series of simulations
% using the PhD_One_Sim.m M-File for the PhD-1 Article.
% Simuluation Series
%
% Use with: PhD_Four_Sim_Kernel.m
%
%
% Last Updated: 12 March 2009
%
% Electrical and Computer Engineering
```

```
% The University of Alabama, Tuscaloosa, Alabama
%--------------------------------------------------------------------------


clc
clear all
close all

numSims = 10;    % Set the number of simulation runs to perform.
pauseControl = 0;

% Select Robot for which to record simulation data


%--------------------------------------------------------------------------
% BEGIN -- Main Loop of Shell
%--------------------------------------------------------------------------
for i_sim = 1:numSims
    tic
    PhD_Three_Sim_Kernel_Corrected


    % Store Simulation Data for Later Use

    Robot_x = 1;
    % Store the Behavioral State Residencey Time (%) data
    state_ik_per_sim(i_sim,:) = [state_ik(Robot_x,1) state_ik(Robot_x,2) state_ik(Robot_x
        ,3)];
    state_ikC_per_sim(i_sim,:) = [state_ikC(Robot_x,1) state_ikC(Robot_x,2) state_ikC(
        Robot_x,3)];


    % Store the Behavioral State Residency Cout (#) data
    % Store min/max r_ik (separation) data
    r_ik_per_sim(i_sim,:) = [r_ik_overall_min(Robot_x) r_ik_delay_min(Robot_x)
        r_ik_delay_max(Robot_x) r_ik_overall_max(Robot_x) ];



    Time_for_Simulation = toc;
    Time_str = num2str(Time_for_Simulation);
    disp(['Simulation Time:  ' Time_str '  (sec)'])

    if (pauseControl == 1)
        pause
    end

    close ALL
    close(execHandle)



end
%--------------------------------------------------------------------------
% END -- Main Loop of Shell
%--------------------------------------------------------------------------



%--------------------------------------------------------------------------
% Write Data to the Output file
%--------------------------------------------------------------------------

fid = fopen('PhD_Three_Sim_Output.m','a');

% Write a Header for the simulation to the Output File
fprintf(fid,'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx');
fprintf(fid,'\n');
```

```
fprintf(fid,'Simulation: 2d:  Ratt=25 Rrep = 15');
fprintf(fid,'\n');
fprintf(fid,'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx');
fprintf(fid,'\n');
fprintf(fid,'\n');

% Write Behavioral State Residency Time Data to Output File
fprintf(fid,'State Time: (Repulsion—Attraction—FreeAction):');
fprintf(fid,'\n');

for i_sim = 1:numSims
    fprintf(fid,'%10.4f, %10.4f, %10.4f,',state_ik_per_sim(i_sim,:));
    fprintf(fid,'\n');

end

% Calculate and Print Average of Time
% Add up each state time for each sim, then divide by numSims

avg_state_ik_per_sim(i_sim,:) = [0 0 0];
for k_state =1:3
    sumTime = 0;
    for i_sim = 1:numSims
        sumTime = sumTime + state_ik_per_sim(i_sim,k_state);
    end
    avg_state_ik_per_sim(i_sim,k_state) = sumTime/numSims;
end

fprintf(fid,'Average State Time:');
fprintf(fid,'\n');
fprintf(fid,'%10.4f, %10.4f, %10.4f,',avg_state_ik_per_sim(i_sim,:));
fprintf(fid,'\n');
fprintf(fid,'\n');



% Write Behavioral State Residency Count Data to Output File
fprintf(fid,'State Count: (Repulsion—Attraction—FreeAction):');
fprintf(fid,'\n');

for i_sim = 1:numSims
    fprintf(fid,'%10.4f, %10.4f, %10.4f,',state_ikC_per_sim(i_sim,:));
    fprintf(fid,'\n');
end

% Calculate and Print Averge of Count
avg_state_ikC_per_sim(i_sim,:) = [0 0 0];
for k_state =1:3
    sumCount = 0;
    for i_sim = 1:numSims
        sumCount = sumCount + state_ikC_per_sim(i_sim,k_state);
    end
    avg_state_ikC_per_sim(i_sim,k_state) = sumCount/numSims;
end

fprintf(fid,'Average State Count:');
fprintf(fid,'\n');
fprintf(fid,'%10.4f, %10.4f, %10.4f,',avg_state_ikC_per_sim(i_sim,:));
fprintf(fid,'\n');
fprintf(fid,'\n');


fprintf(fid,'\n');

% Write r_ik Min/Max Data to Output File
fprintf(fid,'r_ik (min delay_min delay_max max):');
fprintf(fid,'\n');
fprintf(fid,'Ratt = %6.2f', rAttraction);
```

```
fprintf(fid,'\n');
fprintf(fid,'Rrep = %6.2f', rRepulsion);
fprintf(fid,'\n');



for i_sim = 1:numSims
    fprintf(fid,'%10.4f, %10.4f, %10.4f,',r_ik_per_sim(i_sim,:));
    fprintf(fid,'\n');
end

% Calulate Average and Print of r_ik Min/Max data

avg_r_ik_per_sim(i_sim,:) = [0 0 0];
for k_state =1:4
    sumR_ik = 0;
    for i_sim = 1:numSims
        sumR_ik  = sumR_ik  + r_ik_per_sim(i_sim,k_state);
    end
    avg_r_ik_per_sim(i_sim,k_state) = sumR_ik /numSims;
end

fprintf(fid,'Average r_ik:');
fprintf(fid,'\n');
fprintf(fid,'%10.4f, %10.4f, %10.4f,',avg_r_ik_per_sim(i_sim,:));
fprintf(fid,'\n');
fprintf(fid,'\n');

fid = fclose(fid);

%-------------------------------------------------------------------------
% END --- PhD_Four_Sim_Shel.m
%-------------------------------------------------------------------------
```

C.3 Article 3 Output File

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Simulation: 2a:  Ratt=23 Rrep = 17
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

State Time: (Repulsion---Attraction---FreeAction):
   80.2500,     19.7333,      0.0167,
   80.0500,     19.9333,      0.0167,
   78.7333,     21.2500,      0.0167,
   79.5667,     20.4167,      0.0167,
   78.9667,     21.0167,      0.0167,
   79.3500,     20.6333,      0.0167,
   78.8000,     21.1833,      0.0167,
   80.0833,     19.9000,      0.0167,
   79.1667,     20.8167,      0.0167,
   80.4333,     19.5500,      0.0167,
Average State Time:
   79.5400,     20.4433,      0.0167,

State Count: (Repulsion---Attraction---FreeAction):
 4815.0000,  1184.0000,      1.0000,
 4803.0000,  1196.0000,      1.0000,
 4724.0000,  1275.0000,      1.0000,
 4774.0000,  1225.0000,      1.0000,
 4738.0000,  1261.0000,      1.0000,
 4761.0000,  1238.0000,      1.0000,
 4728.0000,  1271.0000,      1.0000,
 4805.0000,  1194.0000,      1.0000,
 4750.0000,  1249.0000,      1.0000,
 4826.0000,  1173.0000,      1.0000,
Average State Count:
```

```
 4772.4000,  1226.6000,     1.0000,


r_ik (min delay_min delay_max max):
Ratt =  23.00
Rrep =  17.00
   14.1421,    16.5502,    25.3236,   25.3236,
   14.1421,    16.5636,    25.4658,   25.4658,
   14.1421,    16.5685,    25.3115,   25.3115,
   14.1421,    16.5458,    25.2994,   25.2994,
   14.1421,    16.5675,    25.2840,   25.2840,
   14.1421,    16.5632,    25.2787,   25.2787,
   14.1421,    16.5590,    25.3736,   25.3736,
   14.1421,    16.5513,    25.3750,   25.3750,
   14.1421,    16.5577,    25.3655,   25.3655,
   14.1421,    16.5673,    25.2514,   25.2514,
Average r_ik:
   14.1421,    16.5594,    25.3329,   25.3329,

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Simulation: 2b:  Ratt=23.5 Rrep = 16.5
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

State Time: (Repulsion—Attraction—FreeAction):
   63.6833,    13.2667,    23.0500,
   64.2333,    13.0500,    22.7167,
   65.1667,    13.2000,    21.6333,
   62.9667,    13.7000,    23.3333,
   66.3333,    12.0000,    21.6667,
   64.2667,    13.4500,    22.2833,
   64.4000,    13.4333,    22.1667,
   63.4500,    12.9167,    23.6333,
   65.2500,    13.4500,    21.3000,
   63.7667,    13.7000,    22.5333,
Average State Time:
   64.3517,    13.2167,    22.4317,

State Count: (Repulsion—Attraction—FreeAction):
 3821.0000,   796.0000,  1383.0000,
 3854.0000,   783.0000,  1363.0000,
 3910.0000,   792.0000,  1298.0000,
 3778.0000,   822.0000,  1400.0000,
 3980.0000,   720.0000,  1300.0000,
 3856.0000,   807.0000,  1337.0000,
 3864.0000,   806.0000,  1330.0000,
 3807.0000,   775.0000,  1418.0000,
 3915.0000,   807.0000,  1278.0000,
 3826.0000,   822.0000,  1352.0000,
Average State Count:
 3861.1000,   793.0000,  1345.9000,


r_ik (min delay_min delay_max max):
Ratt =  23.50
Rrep =  16.50
   14.1421,    16.0794,    23.8517,   23.8517,
   14.1421,    16.0793,    23.8140,   23.8140,
   14.1421,    16.0343,    23.8994,   23.8994,
   14.1421,    16.1021,    23.9250,   23.9250,
   14.1421,    16.0800,    23.8832,   23.8832,
   14.1421,    16.0976,    23.8289,   23.8289,
   14.1421,    16.0921,    23.8570,   23.8570,
   14.1421,    16.0743,    23.9435,   23.9435,
   14.1421,    16.0805,    23.8909,   23.8909,
   14.1421,    16.0730,    23.8404,   23.8404,
Average r_ik:
   14.1421,    16.0793,    23.8734,   23.8734,
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Simulation: 2c:  Ratt=23.75 Rrep = 16.25
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

State Time: (Repulsion—Attraction—FreeAction):
   35.0000,     5.0000,    60.0000,
   28.2167,     5.0000,    66.7833,
   36.2167,     4.2333,    59.5500,
   35.3833,     4.6333,    59.9833,
   35.6667,     5.4833,    58.8500,
   35.5333,     4.1667,    60.3000,
   33.0667,     4.8500,    62.0833,
   28.9000,     4.4000,    66.7000,
   31.4500,     5.8333,    62.7167,
   32.5000,     5.3000,    62.2000,
Average State Time:
   33.1933,     4.8900,    61.9167,

State Count: (Repulsion—Attraction—FreeAction):
 2100.0000,   300.0000,  3600.0000,
 1693.0000,   300.0000,  4007.0000,
 2173.0000,   254.0000,  3573.0000,
 2123.0000,   278.0000,  3599.0000,
 2140.0000,   329.0000,  3531.0000,
 2132.0000,   250.0000,  3618.0000,
 1984.0000,   291.0000,  3725.0000,
 1734.0000,   264.0000,  4002.0000,
 1887.0000,   350.0000,  3763.0000,
 1950.0000,   318.0000,  3732.0000,
Average State Count:
 1991.6000,   293.4000,  3715.0000,


r_ik (min delay_min delay_max max):
Ratt =  23.75
Rrep =  16.25
   14.1421,    15.8135,    24.0165,   24.0165,
   14.1421,    15.8265,    24.1963,   24.1963,
   14.1421,    15.8293,    23.9974,   23.9974,
   14.1421,    15.7995,    24.0018,   24.0018,
   14.1421,    15.8393,    24.0324,   24.0324,
   14.1421,    15.8741,    24.0339,   24.0339,
   14.1421,    15.8266,    24.0421,   24.0421,
   14.1421,    15.8579,    23.9943,   23.9943,
   14.1421,    15.8750,    24.0932,   24.0932,
   14.1421,    15.8545,    24.0101,   24.0101,
Average r_ik:
   14.1421,    15.8396,    24.0418,   24.0418,

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Simulation: 2d:  Ratt=25 Rrep = 15
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

State Time: (Repulsion—Attraction—FreeAction):
    9.0667,     0.0500,    90.8833,
   13.7333,     0.9000,    85.3667,
   15.7000,     0.0000,    84.3000,
    8.2500,     0.0000,    91.7500,
    8.6000,     0.1833,    91.2167,
    9.8667,     0.2667,    89.8667,
   12.4500,     0.6667,    86.8833,
    7.5667,     0.4833,    91.9500,
    8.3833,     0.2500,    91.3667,
   17.4500,     0.0000,    82.5500,
Average State Time:
   11.1067,     0.2800,    88.6133,

State Count: (Repulsion—Attraction—FreeAction):
```

```
  544.0000,     3.0000,  5453.0000,
  824.0000,    54.0000,  5122.0000,
  942.0000,     0.0000,  5058.0000,
  495.0000,     0.0000,  5505.0000,
  516.0000,    11.0000,  5473.0000,
  592.0000,    16.0000,  5392.0000,
  747.0000,    40.0000,  5213.0000,
  454.0000,    29.0000,  5517.0000,
  503.0000,    15.0000,  5482.0000,
 1047.0000,     0.0000,  4953.0000,
Average State Count:
  666.4000,    16.8000,  5316.8000,


r_ik (min delay_min delay_max max):
Ratt =  25.00
Rrep =  15.00
   14.1421,    14.5881,    25.0414,   25.0414,
   14.1421,    14.5594,    25.2494,   25.2494,
   14.1421,    14.1515,    24.9134,   24.9134,
   14.1421,    14.6572,    24.0796,   24.0796,
   14.1421,    14.6252,    25.1280,   25.1280,
   14.1421,    14.6419,    25.2115,   25.2115,
   14.1421,    14.6031,    25.3363,   25.3363,
   14.1421,    14.3671,    25.1150,   25.1150,
   14.1421,    14.6241,    25.0932,   25.0932,
   14.1421,    14.6277,    23.8403,   23.8403,
Average r_ik:
   14.1421,    14.5445,    24.9008,   24.9008,
```