A MULTI-VEHICLE FRAMEWORK FOR THE DEVELOPMENT OF ROBOTIC GAMES: THE MARCO POLO CASE

By

BRENT PERTEET

Bachelor of Science

Oklahoma State University

Stillwater, OK

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER of SCIENCE
May, 2007

A MULTI-VEHICLE FRAMEWORK FOR THE DEVELOPMENT OF ROBOTIC GAMES: THE MARCO POLO CASE

Thesis Approved:

Dr. Rafael Fierro

Thesis Adviser

Dr. Keith Teague

Dr. Sohum Sohoni

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

First and foremost, I would like to thank my thesis adviser, Dr. Rafael Fierro, for letting me be a part of the Marhes team. His support, both financial and moral, and encouragement while I have been a member of the lab has been absolutely invaluable. I feel very fortunate indeed to have had the opportunity to have this research experience.

I would also like to thank Dr. Keith Teague and Dr. Sohum Sohoni for agreeing to serve on my committee.

I would also like to thank the members of the Marhes laboratory, particularly James McClintock and Dr. Omar Orqueda, for their support and many, many helpful discussions and insights into the problems related to this work. I would particularly like to thank James McClintock who, with his can-do attitude, is always ready and willing to give 110% to discuss problems or to write code for an experiment no matter what else he might have to do.

Many thanks to Dr. Silvia Ferrari and the members of the Laboratory for Intelligent Systems and Control at Duke University for the productive collaboration which has led to many of the results presented in this thesis.

Also, many thanks to the NSF and U.S. Army Research Office who have supported this work through NSF CAREER grant #0348637 and ARO grant DAAD19-03-1-0142 (through the University of Oklahoma).

Finally, I would like to dedicate this work to my beautiful and long-suffering wife,

Jessica, who has been extraordinarily patient and has made many sacrifices to support me as I complete this research. I would also like to dedicate this work to my parents, Jerry and Tracy Perteet, who taught me to work hard and to love learning and who have encouraged me in whatever I have attempted to do.

Brent Perteet

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

In the engineering disciplines, especially in the United States, there exists a great need to address the underrepresentation of various ethnic groups. As Figure 1.1 from [1] clearly shows, ethnic groups such as Native Americans are severely underrepresented in the engineering disciplines. There may be several contributing factors among which

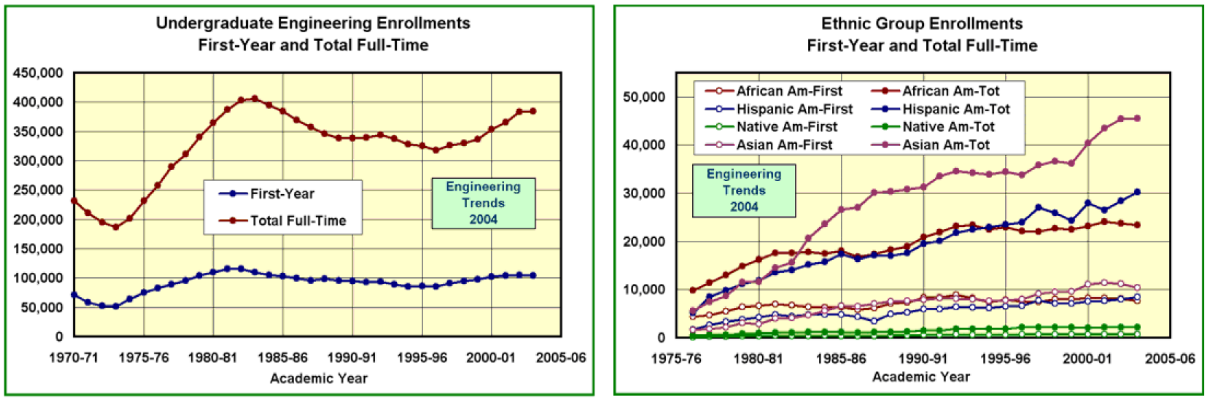


Figure 1.1: a.) Undergraduate engineering enrollment, b.) Engineering enrollment by ethnic group

may be cultural or economic. There exist many opportunities and organizations which provide financial assistance to students that are in these groups. However, many of these programs are designed to address such issues when students reach high school. What seems to be lacking are programs which introduce and motivate younger students to the science, technology, engineering and mathematics (STEM) areas in schools which may not have adequate funding or resources to provide these tools. This need serves as motivation for creating the tools for such a program.

Robotics, particularly mobile robotics, has been used effectively as a teaching tool for introducing students of all ages to science and engineering. This fact, together with the fact that children today are quite sophisticated when it comes to electronic devices and video games, motivates the development of a multi-vehicle mobile robotic framework which uses interactive games to facilitate interest and discussion toward engineering related topics. The developed framework, called *R*obotic Games, is designed to be a flexible platform which uses commercial hardware and software that may be transported to schools and presented to children. The games developed as part of Robotic Games are motivated by the various sensors, capabilities, and applications of mobile robotics in hopes that the games themselves motivate additional explanation, discussion, and discovery by young participants.

Research and development of the Robotic Games framework has itself motivated additional research in the area of pursuit-evasion games which is also addressed in this thesis. This research emerges from the children's game Marco Polo developed as part of Robotic Games which exhibits interesting interaction dynamics between a pursuer and an evader through intermittent communication.

Pursuit-evasion game theory is a well established cross-disciplinary area of research, including computer science and engineering, that has produced many important results in its half century history. This research area investigates the emergent interaction dynamics between agents or individuals who are attempting to evade capture and one or more pursuer agents who are attempting capture of evaders. These two classes of agents may be subject to varying degrees of sensing or motion constraints which produce interesting problems. This area deals with both pursuit and evasion strategies of each class of agent or player given the constraints of each. Applications of pursuit-evasion game theory include predator and prey models in nature, missile and target models in warfare, and search and rescue operations.

Marco Polo pursuit-evasion is a new problem in this area and may have application to scenarios where a team of robots may need to cooperate with one another when communications are available intermittently or when information about a target is available intermittently.

A problem motivated by Marco Polo pursuit-evasion considers the use of multiple mobile sensors to optimally cover an area such that intruders entering this area may be detected, pursued, and captured. Because of the proliferation of reliable, low-cost sensor networks and developments in autonomous vehicle technologies, advanced surveillance systems are being produced, where both the sensors and their platforms are characterized by a high degree of functionality and reconfiguration. Examples include landmine detection and identification by multiple and heterogenous sensors installed on ground vehicles [2], [3], [4], sensor networks for monitoring endangered species in their natural environment [5], robot-installed sensors to monitor urban environments, production in manufacturing plants, and civil infrastructure, high-confidence medical devices, and intruder and target detection systems. These networks are expected to operate reliably in dynamic environments with little human intervention. However, coordinating such large heterogeneous sensor networks is extremely difficult and requires the development of novel methods of communication, motion control, computation, proactive estimation and sensing, and power management.

One paradigm common to many sensing applications consists of one or more sensors installed on robotic platforms that must move through an environment to obtain measurements from multiple targets. A possible urban scenario is shown in Figure 1.2 in which heterogeneous sensors (e.g., static, mobile, ground, and aerial) work cooperatively in a dynamic target tracking mission that can be viewed as a sensing-pursuit game.

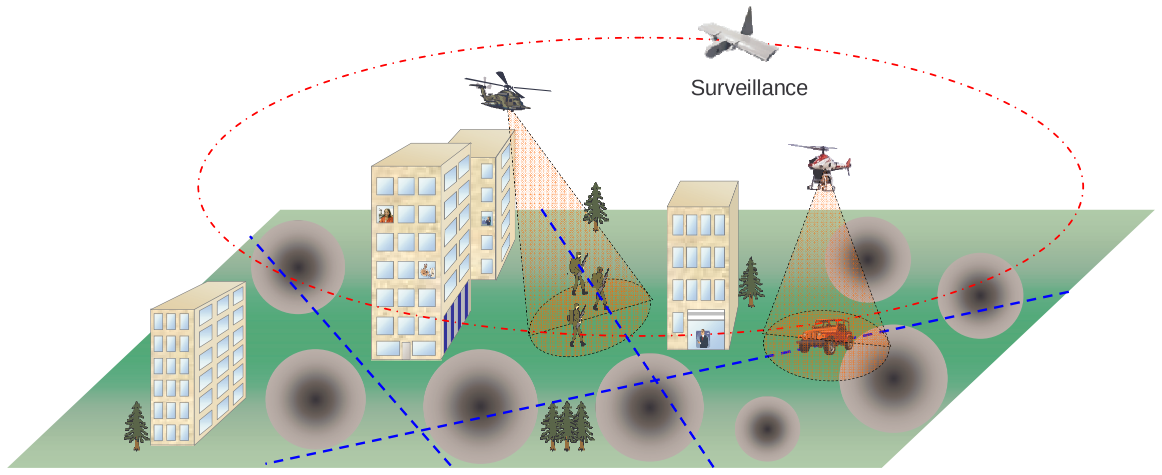


Figure 1.2: Aerial and ground sensors working together in detecting and intercepting dynamic targets (e.g., intruders, evaders) in an urban environment.

1.2 Contributions

This thesis documents a number of innovative and practical contributions to the areas of engineering education, multi-vehicle cooperation, pursuit-evasion games, and sensor networks. In the areas of engineering education and multi-vehicle cooperation, this thesis presents a framework designed to facilitate the development of educational programs and games which use multiple robots and supporting user interaction devices. Although the framework consists of a variety of off-the-shelf hardware and software components, we have integrated these components in a manner which has not previously existed. By bringing these various pieces together, the emergent framework displays remarkable flexibility. In addition to the framework, we present several multi-robot games — collectively known as "Robotic Games" — that incorporate and demonstrate various aspects of autonomous mobile robotics.

In the area of pursuit-evasion games and sensor networks, this thesis makes the contribution of introducing and formally defining a pursuit-evasion game with intermittent communication or cooperation which is based on the children's game Marco Polo. The Marco Polo problem is a new type of pursuit-evasion problem that has not been previously addressed in the literature. Because Marco Polo is part of the Robotic Games, we present an implementation of the game in addition to the formal definition. Additionally, we present a particular application of the problem in which multiple mobile sensors are coordinated for detection and pursuit. The main contribution here is the control strategy for performing the pursuit maneuver as well as the adaptation and implementation of the application in both simulation and hardware.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. We begin by discussing relevant related work in the areas of multi-vehicle platforms, robotics education, and pursuit-evasion games in Chapter 2.

In Chapter 3, we present the Robotic Games framework for educational robotics. The chapter begins with sections which describe each of the major components of the framework including the off-the-shelf hardware and both open source and commercial software libraries which are integrated. The chapter continues by describing the software architecture that is adopted to aid in the development, maintenance, and presentation of the educational program. Finally, the chapter provides a discussion of several of the games which make up the Robotic Games Library and demonstrate various aspects of robotics interaction and research.

Chapter 4 presents a case study including formal problem formulation and implementation of the children's game Marco Polo as it pertains to the pursuit-evasion and robotics problem that it motivates. The description of the implementation of this game on the platform discusses the specific algorithms used including visual simultaneous localization and mapping (vSLAM$^{\text{TM}}$) as well as the wireless communications scheme.

Chapter 5 presents a particular application of the Marco Polo problem which has motivated the development of a methodology for coordinating multiple mobile sensors to detect and pursue targets which are constrained to straight line motion. This chapter summarizes the application scenario and the multi-objective optimization methodology and pursuit strategy. The chapter concludes by presenting several simulation scenarios and hardware implementation of the methodology.

Finally, Chapter 6 discusses conclusions and suggests future work.

Chapter 2

Related Work

The area of educational and socially assistive robotics has garnered much attention in recent years. Many researchers are actively working in the area of educational robotics, which focuses on motivating and encouraging students to become more involved with and chose careers in the science, technology, engineering, and mathematics (STEM) fields. Researchers have used robotics and developed materials to introduce engineering in general to elementary and secondary students [6]. Additionally, several researchers have successfully participated in competitions such as RoboCup and RoboFlag, which involve robots playing various games including soccer and capture-the-flag [7, 8, 9].

Other researchers have developed robotic presentations such as the Robotics Roadshow Program presented in [10] as shown in Figure 2.1, which is designed to reach out to underserved, rural schools in the state of Kansas. The researchers who developed this program have created a set of simple games which involve elementary and secondary students with robots to stimulate interest in science and technology. Results from this program indicate that robots are an effective means of accomplishing the program goals of providing additional learning resources and motivating students in the direction of science and engineering. Other recent focus given to the area of educational robotics are in [11]. Although this program focuses on teaching robot design to university undergraduate students, it is yet another example of how robotics is being used in engineering education. Additionally, several groups have recently

Figure 2.1: The Robotics Roadshow "Escape the Circle" demonstration. [10]

developed multi-vehicle platforms for research. At the Marhes laboratory, we have created a rugged outdoor platform based on the Tamiya TXT-1 monster truck [12]. A similar platform, the Multiple Autonomous Robots testbed, has been developed at the University of Pennsylvania [13]. Other groups have introduced frameworks based on small hovercraft vehicles such as the Caltech multi-vehicle wireless testbed [14] and the HoTDeC [15], developed at the University of Illinois. Still others feature ground as well as unmanned aerial vehicles (UAVs) such as MIT's multi-vehicle testbed [16].

A great deal of work has also been done related to pursuit-evasion and target tracking. The authors in [17] discuss basic motion planning and control strategies for multi-robot systems. The Marco Polo game has motivated a multi-robot localization problem introduced in [18]. In [19], the authors develop a decentralized motion coordination algorithm for tracking groups of dynamic targets. Strategies to search for moving targets in a two-dimensional plane are considered in [20, 21]. In [22] the authors show that a pursuer can detect an arbitrarily fast evader using a randomized strategy. The evader can be captured by two pursuers solving a *lion and man* problem assuming that at least one pursuer is as fast as the evader. Figure 2.2 depicts the lion's strategy for capturing the man given alternating moves of unit length in the single pursuer, single evader case. A variety of optimization techniques [23] have

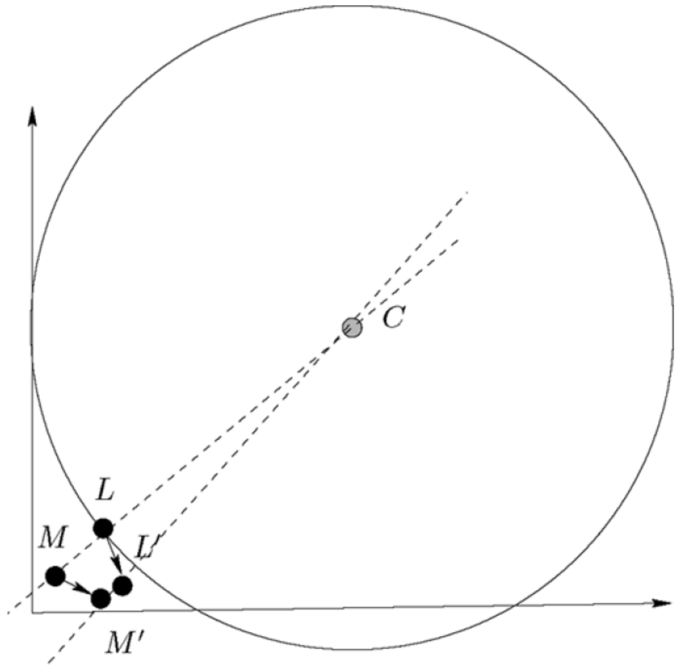


Figure 2.2: The Lion's strategy of the *lion and man* problem. [22]

been applied to the coordination of robotic networks engaged in distributed sensing tasks [24]. Optimal motion planning for multiple robots is considered in [25, 26]. Distributed motion planning approaches are discussed in [27]. In [28] the authors study optimal sensor placement for mobile sensor networks. Another relevant work is described in [29]. In this reference, the authors propose a motion control algorithm for a mobile sensor agent operating in a nonconvex polygonal environment such that the area of the region visible to the sensor is maximized. In [30], the authors discuss the use of a sensor network and motion planning for observing features of interest in an environment.

Cooperative pursuit strategies to detect, intercept, and capture intelligent evaders in cluttered environments are described in [31]. However, the difficulty of solving pursuit-evasion games and obtaining closed-form solutions of the underlying optimization problem has motivated an intensive research aimed at developing algorithmic approaches.

The problem of finding the configuration of a network with multiple sensors that optimizes the number of tracks intercepted is considered in [32]. A Bayesian network (BN) approach is used to develop an automated computer player for CLUE in [33]. In other related work [19] the authors develop a decentralized motion coordination algorithm for tracking tasks of dynamic targets. Strategies to search for moving targets in a two-dimensional plane are considered in [20, 21]. Motion coordination strategies of multiple vehicles visiting targets generated by a stochastic process are proposed in [34]. In [35], multiple sensors are used to search an environment for a hidden target. Optimal strategies for locating the target are investigated in order to maximize the probability of detection in some amount of time. Also, in [36], the motion planning problem of coordinating a limited number of mobile sensor agents to observe a larger number of targets in an environment is investigated. This problem is viewed as an optimization problem where the quantity to minimize is the time between target observations. In [37], the authors present a method for planning the motions of a limited number of sensor resources for searching for and detecting elusive targets. Motion is planned based upon the terrain and probability estimates.

Most of the research relating sensor measurements to robot motion planning has focused on the effects that the uncertainty in the geometric models of the environment has on the motion strategies of the robot [38], [39], [40], [41], [42]. Hence, considerable progress has been made toward integrating sensor measurements in topological maps [43], and on planning strategies based only on partial or nondeterministic knowledge of the workspace [44], [45]. Another line of research has investigated the extension of motion planning techniques to the problem of sensor placement for achieving coverage of unstructured environments [46], or of a desired visibility space [47]. For instance, probabilistic roadmap planners that were originally developed for placing milestones in targeted regions of the free space [48], such as, in narrow passages [49] or near obstacles [50], have been effectively modified to place milestones near regions of interest identified by means of sensor measurements [51].

The following chapter, Chapter 3, builds upon some of the related ideas summarized here. It describes the Robotic Games framework and its role as an educational and socially assistive tool to facilitate learning and interest in science and engineering. Furthermore, Chapter 5 integrates some of the work described above and presents a methodology for using mobile sensor networks in the detection, tracking, and pursuit of intruders.

Chapter 3

Robotic Games

3.1 The Robotic Games Framework

The primary hardware components of the platform include an Evolution Robotics Scorpion Robot and a Toshiba M400 Tablet PC. The architecture also makes use of desktop computers and other common devices such as a wireless access point and a handheld computer or personal digital assistant (PDA). Figure 3.1 illustrates the major components of the framework which we describe in detail in the following sections.

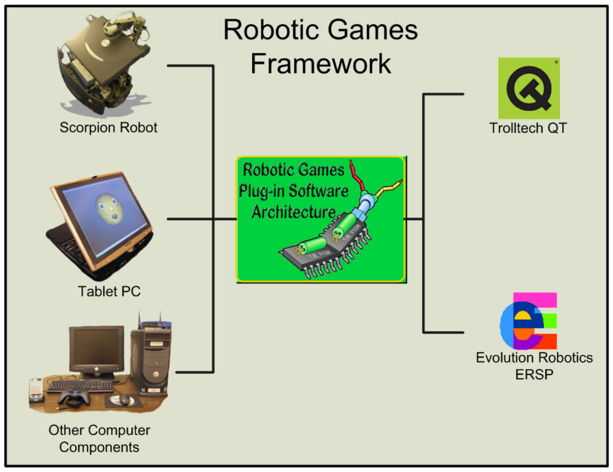
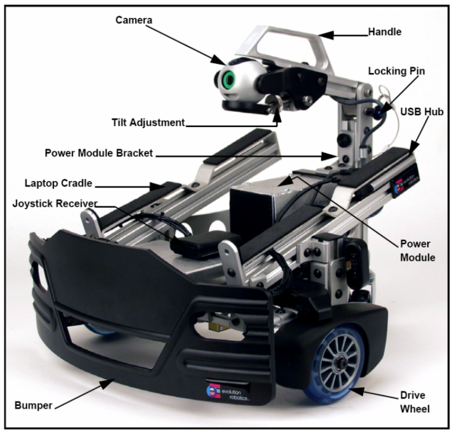


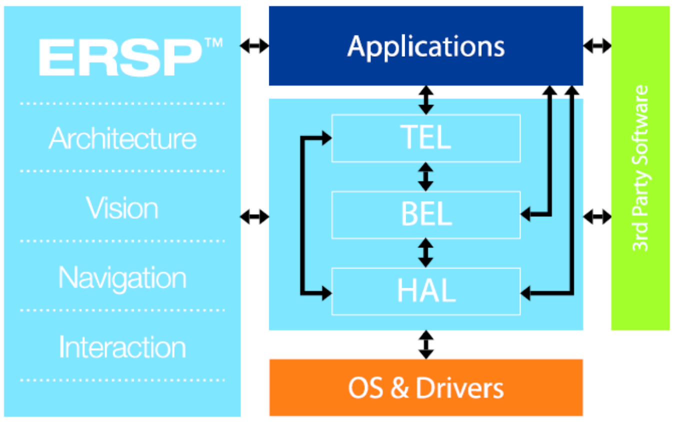
Figure 3.1: The Robotic Games Framework.

The Scorpion Robot

The robotic platform used to implement this framework is the Scorpion Robot from Evolution Robotics [52] shown in Figure 3.2(a). The robot features a variety of simple sensors for gathering information about the environment including 20 infrared range-finding sensors, a contact-sensing bumper, a camera with a wide angle lens, and high-resolution optical encoders on the two motors. All sensors except for the camera connect to a central controller called the Robot Control Module (RCM). A servo controlled differential drive system, which is also controlled using the RCM, provides locomotion. High-level control is accomplished by a Tablet PC that is mounted on the robot and connects to the RCM and camera by USB. Other USB devices such as a wireless joystick or other USB sensors may be connected to the Tablet PC as well.



(a)



(b)

Figure 3.2: a.) The Scorpion Robot and b.) The Evolution Robotics Software Platform [53].

A Tablet PC Computer

A Toshiba M400 Tablet PC is mounted on the Scorpion Robot to provide high-level processing and control functions. Though the Scorpion may be used with any suitable notebook computer, a Tablet PC has features that are particularly useful on a platform designed for education. Because the screen rotates, the display is visible while the robot is running such that a 3D face that displays emotions and reacts to the environment might be rendered on the screen as shown in Figure 3.3. This feature allows the robot to interact with humans in a more natural way, which enhances its capabilities as a educational tool. Also, the Tablet PC supports pen input, which provides another method for interacting with the robot.

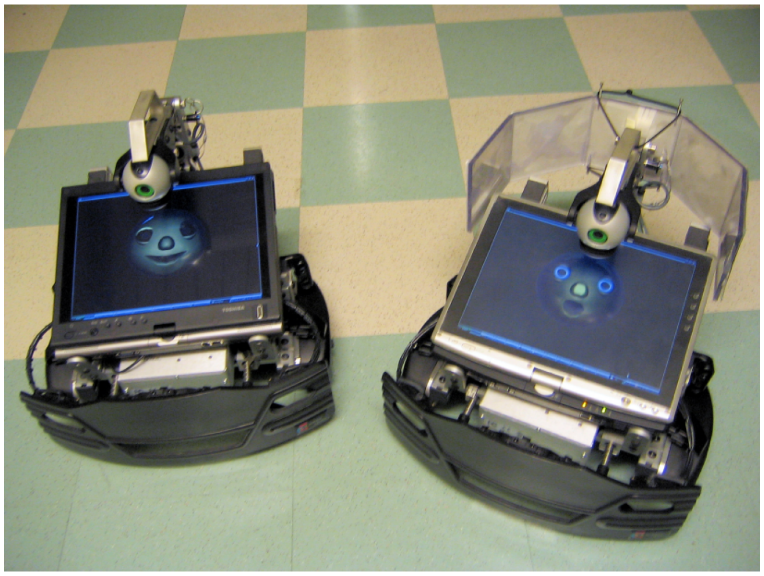


Figure 3.3: The two major hardware components of this framework – the Evolution Robotics Scorpion Robot and a Tablet PC.

Other Hardware

Many applications of this framework benefit from other common devices. Specifically, the games that we have developed use desktop computers, wireless joysticks, a wireless access point, and a PDA. The access point allows for centralized communication and enables the robots to communicate with desktops for user input. In some games, desktops are not needed but a mechanism to coordinate all of the robots (start, stop, pause the game, etc.) is required. In these instances, a PDA with wireless communication capabilities works well.

3.1.1 Software Components

This framework combines the powerful robotics processing and control algorithms available in the Evolution Robotics Software Platform (ERSP) with the Trolltech Qt application framework. These two libraries are integrated using an architecture that allows developers to easily create new games and demonstrations.

The ERSP Library

The ERSP library is an extensive software platform for developing high-level controllers while abstracting the developer from the underlying hardware. The platform consists of an advanced API for the C++ programming language. The API is available for both the Microsoft Windows and Linux platforms. We have chosen to use the Windows version for development of the Robotic Games.

The strength of the ERSP system is derived from the three layers which comprise the architecture of the system which is depicted in Figure 3.2(b). These layers are the Task Execution Layer (TEL), the Behavior Execution Layer (BEL), and the Hardware Abstraction Layer (HAL). At the bottom layer, the HAL is responsible for providing a common interface to the underlying robot hardware for the rest of the system. In other words, any robot may be used with ERSP as long as drivers that implement the required interfaces are provided. The BEL facilitates the development of simple, modular robotic abilities which are referred to as behaviors within the ERSP architecture. These behaviors may be connected together using a graphical utility to create a block diagram called a *behavior network* such as the very simple one shown in Figure 3.4. The network can then be executed and the robot performs the actions specified within the network. Finally, the Task Execution Layer (TEL) provides services for developing the highest level components in the software, known

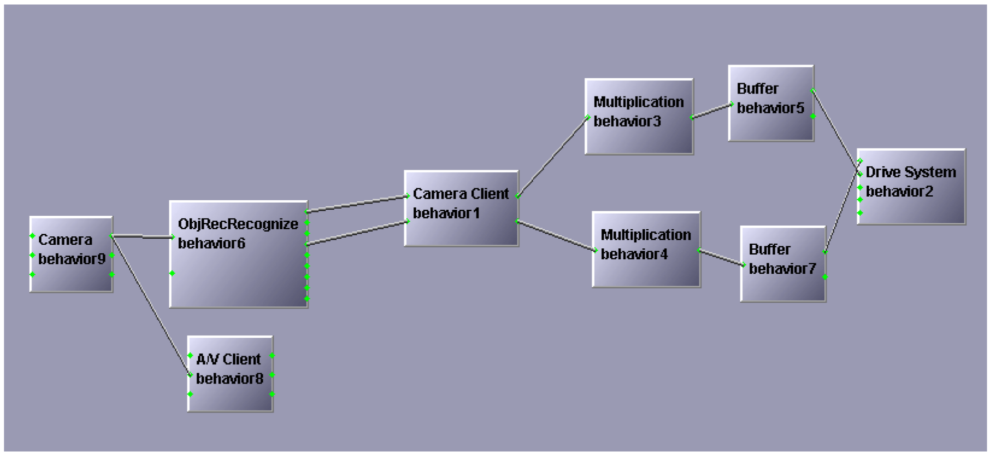


Figure 3.4: A simple ERSP behavior network.

as tasks. While behaviors are designed to have tight control loops in order to be highly reactive to situations in the environment, tasks are designed to coordinate the communication and synchronization of multiple behaviors, which may be connected either sequentially or in parallel.

The ERSP API includes a comprehensive set of advanced algorithms particularly for vision and navigation. At the center of the vision module is the ViPR$^{\mathrm{TM}}$ (Visual Pattern Recognition) algorithm. Among other things, this algorithm provides behaviors which handle object recognition, motion flow, and color segmentation which are useful for detecting objects and their pose, movement, and skin color. The most significant component of the navigation module is the implementation of the vSLAM$^{\mathrm{TM}}$ (Visual Simultaneous Localization and Mapping) algorithm [54]. This algorithm uses input from the wheel encoders and camera to accomplish the simultaneous localization and mapping function. The navigation module also contains support for path planning, obstacle avoidance, exploration, and population of an occupancy grid map.

The Qt Application Framework

Qt is an open-source, C++ application framework that is produced by Trolltech and is available under the GNU Public License for research and non-commericial use. The library supports several platforms including Windows, Linux, and MacOS and provides a rich set of classes for application development. For example, the Qt library contains object models for XML processing, multi-threading, networking, GUI design, and plug-in development. Qt also allows window based GUIs to be developed easily. Classes for common window interface controls such as menus and buttons may be combined to create advanced, event-driven interfaces.

The Robotic Games Library

The Robotic Games library consists of a set of software applications that are designed to manage games which conform to the Robotic Games software architecture. To conform to the software architecture, games must be designed using C++ with the support of the Qt and ERSP libraries. The architecture uses Qt's plug-in capabilities to manage the games. Thus, new games must also implement a plug-in interface which includes a set of common methods (start, stop, pause, etc.) The software architecture defines and manages two categories of hardware: Displays (desktop computers) and Robots (Scorpion and Tablet PC). As part of the plug-in interface, each game must specify the number of displays and robots that are required.

The Robotic Games library consists of two software components: the Game Manager and the Coordinator. A block diagram of the system architecture is shown in Figure 3.5. The Game Manager runs on robots and displays. This application is able to load and configure games that conform to the plug-in interface described above. The Coordinator application runs on a Dell Axim X50v handheld computer running

Windows Mobile 2003 and is written in C# with the .NET Compact Framework. This application is used by game referees to manage the games. Communication between the Coordinator and Game Managers is accomplished by a simple text message based protocol which is sent on top of the standard UDP/IP and TCP/IP networking protocols.

Upon execution and at the referee's request, the Coordinator queries the network using the UDP protocol for robot and display client discovery. Clients running the Game Manager listen for these discovery queries and respond by broadcasting their presence on the network also using UDP. The Coordinator receives these responses and registers each robotic node in an internal database. Using the Coordinator application, the referee may select a game for the system to run. The Coordinator then queries the game's plug-in for a list of configuration parameters and dynamically creates forms to configure the game. Finally, the referee may select a button to begin the game and the Coordinator instructs each Game Manager to configure and begin execution.

As an example, we now show how the referee uses the Coordinator on the PDA to control the system. In this scenario, the referee begins a game called "Scavenger Hunt" which requires a robot and a display for handling user interaction. The first step for the Coordinator is to locate and query nodes which have executed the Game Manager. Figure 3.6 shows a tree view of the available robot and display nodes which have responded to the Coordinator request for information as well as the games which they are able to play. The system provides an alternative view as shown in Figure 3.7 which presents the referee with the games available. By selecting the "Play game..." menu item, the referee is able to further configure the game by providing various parameters as shown in Figure 3.8. In this case, each display must be paired with

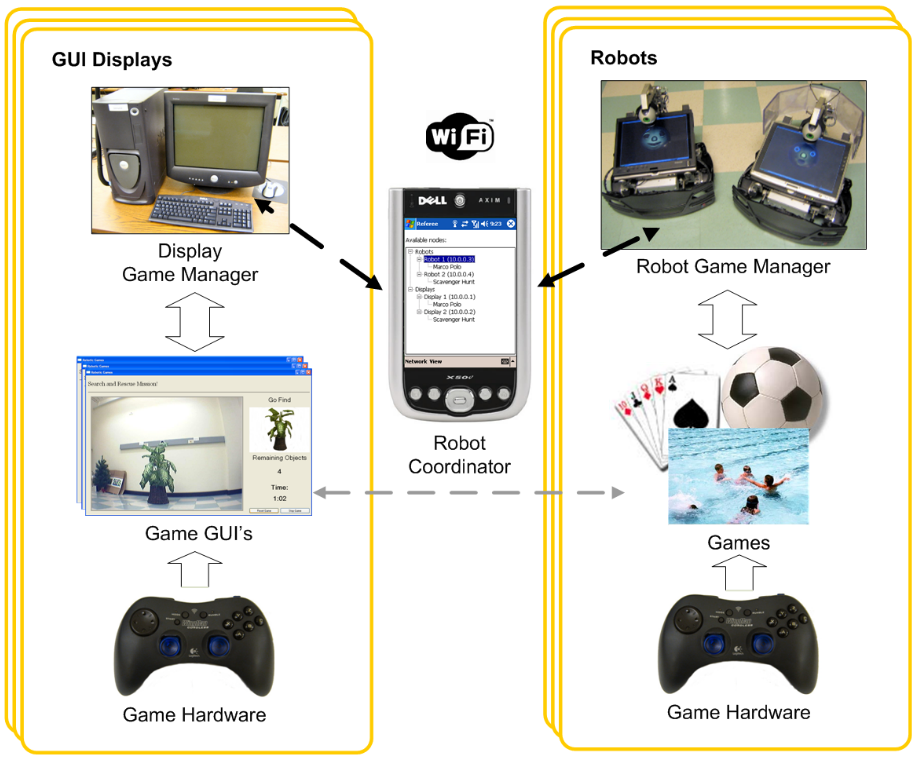
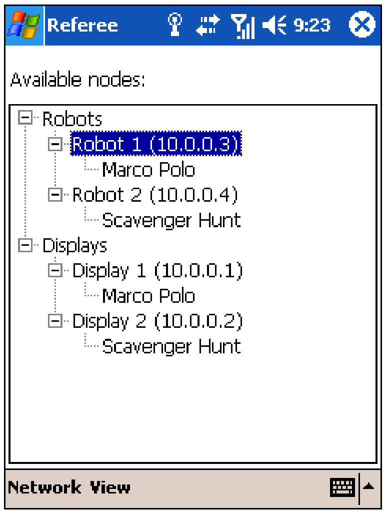


Figure 3.5: The Robotic Games System Architecture.

a robot for sending joystick commands and receiving video and other data. Upon completion of game configuration, the referee begins the game by selecting the "Start Game" button which instructs the connected nodes to load the game plug-in library and begin execution. While the game is running, the Coordinator application displays the "Game in Progress" dialog as shown in Figure 3.9. When the game is over or when the referee ends the game, the Coordinator commands participating nodes to unload the game plug-in and wait for further commands.

3.2 Game Examples

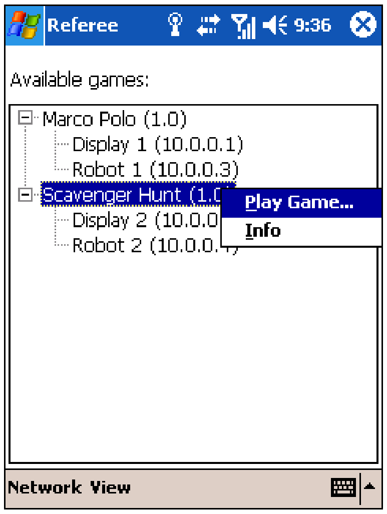
Using the framework discussed in Section 3.1, several games have been developed for the Robotic Games project. The games are designed to be educational, demonstrating various aspects of the state-of-the-art in robotics and engineering.



Available nodes:

```
Robots
  Robot 1 (10.0.0.3)
    Marco Polo
  Robot 2 (10.0.0.4)
    Scavenger Hunt
Displays
  Display 1 (10.0.0.1)
    Marco Polo
  Display 2 (10.0.0.2)
    Scavenger Hunt
```

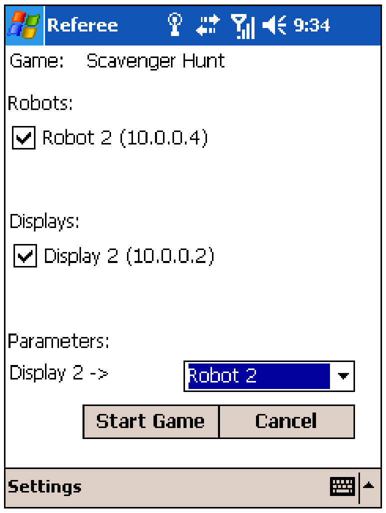
Figure 3.6: Available nodes.



Available games:

```
Marco Polo (1.0)
  Display 1 (10.0.0.1)
  Robot 1 (10.0.0.3)
Scavenger Hunt (1.0
  Display 2 (10.0.0
  Robot 2 (10.0.0.
```

Figure 3.7: Available games.



Game: Scavenger Hunt

Robots:

☑ Robot 2 (10.0.0.4)

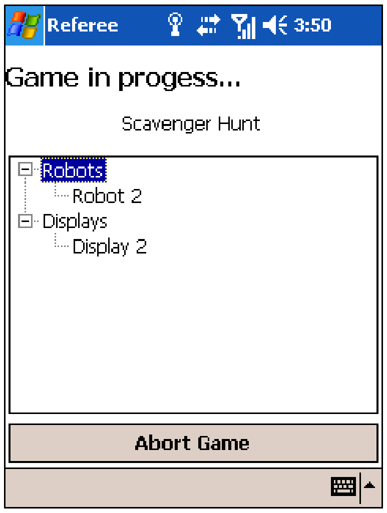
Displays:

☑ Display 2 (10.0.0.2)

Parameters:

Display 2 -> Robot 2

Figure 3.8: Game configuration.



Game in progess...

Scavenger Hunt

```
Robots
  Robot 2
Displays
  Display 2
```

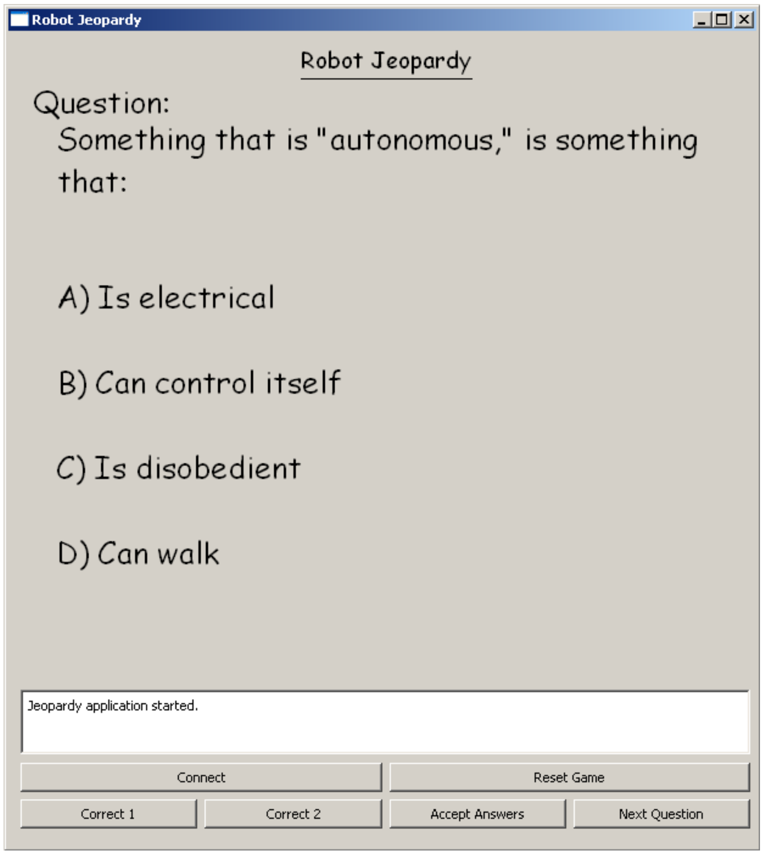
Figure 3.9: Game in progress.

3.2.1 Robot Jeopardy

When presenting Robotic Games to a group of students, we begin with an "Introduction to Robotics" presentation which is targeted to the age group of the participants. This presentation gives a history of robotics, examples of robots in everyday life, types of sensors, and entertaining videos showing current robotic research projects. The game Robot Jeopardy has been designed as a means to motivate students to absorb as much information as they can from the presentation. In this game, teams of students must answer questions based upon the presentation in order to advance their robots down a race track.

Two Scorpion robots are initially placed side by side at the starting line facing their respective team. Teams are given sets of cards which contain a pattern and a letter (A, B, C, or D) which correspond to the possible answers to the multiple choice questions asked by the game referee. When the referee announces a random question from the pool which may also be projected on a screen as in Figure 3.10, students must select the correct answer card and quickly present the card to the robot for visual recognition. The first robot which recognizes the correct answer moves toward the finish line by some distance. The process continues until one of the robots crosses the finish line and wins the game as shown in Figure 3.11.

This game uses the object recognition capabilities in the ERSP library which is based upon the ViPR algorithm. The various patterns are trained into an object database and loaded on each robot. The application which controls the game (Figure 3.10) runs on a desktop PC and establishes a connection to each robot on startup. The robot applications process camera images and forwards object recognition data to the control application. If the control application is accepting answers to a question, it commands the first robot that provides the correct answer to move forward a distance.


Robot Jeopardy

Question:
Something that is "autonomous," is something that:

A) Is electrical

B) Can control itself

C) Is disobedient

D) Can walk

Jeopardy application started.

Figure 3.10: The user interface of "Robot Jeopardy."

This game was expertly designed and implemented by James McClintock using the Robotic Games framework.

3.2.2 Obstacle Course

In the Obstacle Course game, teams of children guide a robot through a game area filled with obstacles. Obstacles might include cones to traverse, traps to avoid, or objects to locate. The goal of the game is to complete the course as quickly as possible. Each robot is given a limited amount of energy E. The game artificially limits the robot's speed v_ℓ to

$$v_\ell = E v_{\max} / E_{\max},$$

Figure 3.11: Students playing "Robot Jeopardy"

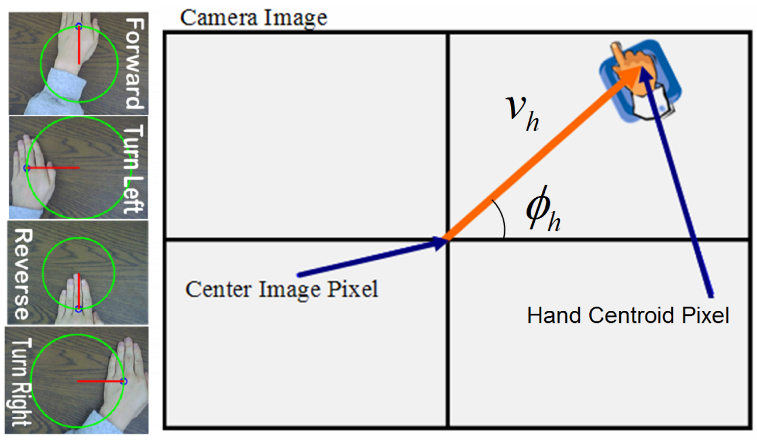


Figure 3.12: The "Obstacle Course" game uses hand movement for robot control.

where $E_{\max}$ and $v_{\max}$ are constants representing the robot's maximum energy and linear velocity respectively. As the robot moves, its energy is depleted as

$$\dot{E} = -(v/v_\ell)^2,$$

where v is the robot's current velocity [55]. Thus, driving as fast as possible, $v = v_\ell$, reduces the robot's energy the fastest. When the robot's energy is depleted, it must "rest." While resting, the robot cannot move, but its energy is restored at a constant rate. To make the game more interesting, we have integrated a vision based hand recognition system to maneuver the robot. This system uses the HandVu library

[56] to determine the pixel location of a hand's centroid within a camera frame. A vector v_h is calculated from the center of the image to this point. This vector is first normalized to the display size and then the angle ϕ_h is calculated as shown in Figure 3.12. To control the robot, values for its linear speed v and angular speed ω are taken as

$$\begin{aligned} v &= v_h \sin \phi_h, \\ \omega &= v_h \cos \phi_h. \end{aligned}$$

This mapping is designed to simulate joystick control. For example, when the hand is at the top of the image and horizontally centered, the robot moves forward. If the hand is on the left side of the image and vertically centered, the robot spins left in place.

3.2.3 Scavenger Hunt

A scavenger hunt is a game in which individuals or teams are given the task of locating a variety of objects or information. Scavenger hunts are often used in educational settings to have students learn to locate pieces of information about a given topic from a variety of sources. In this way, students learn about the subject of the hunt while also learning about the methods for gathering information.

In the case of the Robotic Games, the Scavenger Hunt game was designed to demonstrate teleoperation of a mobile robot in a potentially hostile environment for the purpose of gathering information without endangering the human operator. For the setup of the game, robots are placed in a room away from and out of the sight of the operator. The operator uses a joystick and video feedback from one of the robots within the user interface shown in Figure 3.13. When the game begins, a timer begins to track the operator's efficiency while he or she searches the environment. The game sequentially presents the player with randomly chosen objects to locate in the environment with both an image of the object as well as by a computer synthesized voice which also describes the object. As the robot moves, it uses its object recognition capabilities to let the user know when it sees one of the objects that it has knowledge of, whether it is the one being searched for or not. If the object matches the one that the operator is looking for and the object is recognized within a certain distance of the robot, the user interface clears the current object and moves on to the next. The game notifies the player when he or she has located all of the objects and then presents the player's total time.

This game may be played in a variety of ways depending on the particular audience and time constraints. One mode of play is a "relay race" in which teams of students are formed and each member locates an object and passes control to the next member. Teams may then compete against the clock or against each other. Due to the nature of the game, a number of teams equal to the number of available robots may participate. Another mode of play is that the participants compete against one another in a tournament arrangement.

Figure 3.4 depicts the the ERSP behavior network which is executed on each robot. This relatively simple network illustrates the power of the ERSP system. There is a "camera" behavior which obtains image frames from the camera resource and forwards them to the "ObjRecRecognize" object recognition behavior as well as the "A/V Client" behavior which forwards video over the network to the GUI on another computer. The "Camera Client" behavior is an example of a custom ERSP "Malleable Behavior" for sending arbitrary data over the network. In this case, the

"Camera Client" sends the results of the object recognizer and receives the joystick data. The "Multiplication" and "Buffer" behaviors scale and buffer the joystick input before passing this data to the "Drive System" behavior which ultimately control the robot's motors.

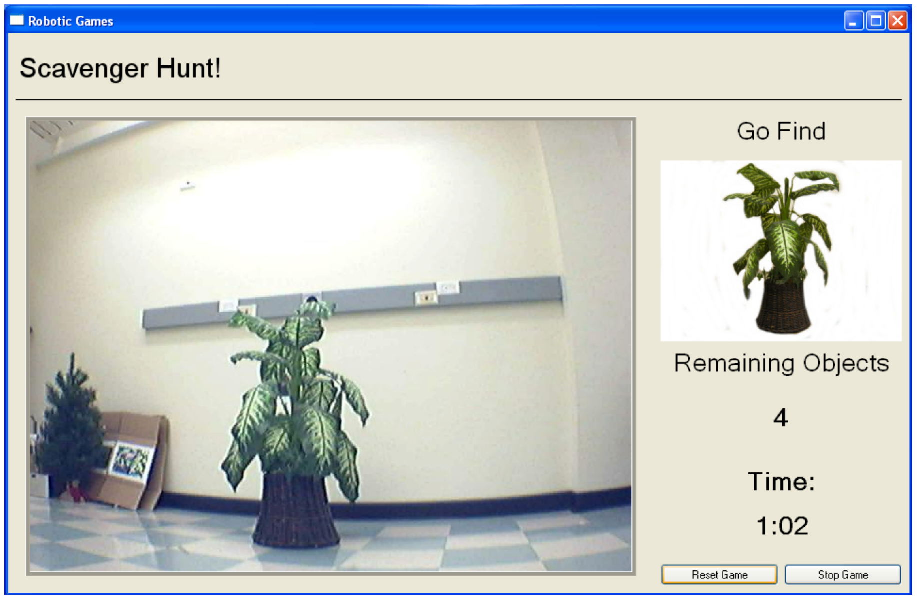


Figure 3.13: "Scavenger Hunt" user interface.

3.2.4 Marco Polo

Marco Polo was a famous Italian explorer and trader who lived during the thirteenth and fourteenth century [57]. His travels took him along the Silk Road to China and made him one of the first Westerners to bring documentation of the Chinese culture and people back to Europe. Many children today play a swimming pool game which is based upon his life and extraordinary travels. This game requires two or more players, one of which is called "Marco." Marco must keep his or her eyes closed while attempting to tag (touch) one of the other players. Other players may keep their eyes open as they move around the pool trying not to be tagged by Marco. When Marco announces “Marco?”, all other players must immediately respond with “Polo!”. This audible cue is what allows Marco to track other players. Based on Marco’s knowlege of the game area (the pool) and prior knowledge of the other players (speed, abilities, etc.), he or she forms strategies for pursuing the other playes. When a player is ultimately tagged by Marco, he or she must immediately shut his or her eyes and become the new Marco. The previous Marco may open his or her eyes to become a regular player.

Marco Polo is a pursuit-evasion game in which the pursuer receives information about the evaders’ location in random time intervals. The game uses two Scorpion robots operating in a leader/follower configuration. The pursuer robot is autonomous while the evader robot is controlled by a participant using a wireless joystick. The goal for the human player is to evade capture by the autonomous pursuer robot for as long as possible. In this game, the evader robot is defined as “caught” when the separation distance between the two robots’ centers falls below a threshold value for some length of time. At the beginning of the game, the two Scorpion robots are placed together at random locations within the game area. The player is given a wireless joystick, informed about the rules of the game, and instructed to begin moving. Once the game is running, the evader robot immediately begins responding to commands from the player’s joystick. At a constrained random interval, the pursuer robot announces “Marco?” over its laptop computer’s speakers and uses the computer network to request the position of the evader. Once the pursuer receives the evader’s location for the first time, it begins moving. Following this, the communication process described above is repeated. The human player generally attempts to maneuver around the pursuer robot in such a way as to maximize the distance between the two. The optimal strategy for a pursuer with incomplete information based on intermittent cooperation is a challenging research problem that is relevant to this game.

Chapter 4

A Marco Polo Case Study

In this chapter, a formal definition of the Marco Polo problem is presented together with an experimental implementation of the game using the framework discussed in Chapter 3.

4.1 Problem Formulation

From a game design standpoint, the goal of Marco Polo is to capture a group of intelligent evaders as quickly as possible using a team of autonomous pursuers that have intermittent knowledge of the evaders' locations. The remainder of this section presents a formal definition of this problem including the assumptions that will be made and the terminology that will be used to describe the game.

Let the game area be referred to as $\mathcal{S}$ and its boundary as $\partial\mathcal{S}$. We assume that $\mathcal{S} \subset \mathbb{R}^2$, and that it is convex and bounded. Associated with $\mathcal{S}$ is a fixed coordinate frame $\mathcal{F}_{\mathcal{S}}$. The pursuers or mobile sensor agents are nonholonomic vehicles that can be modeled using the unicycle model [58]

$$\begin{aligned}\dot{x}_p^i &= v_p^i \cos\theta_p^i,\\ \dot{y}_p^i &= v_p^i \sin\theta_p^i,\\ \dot{\theta}_p^i &= \omega_p^i,\end{aligned} \tag{4.1}$$

where $(x_p^i, y_p^i, \theta_p^i) \in SE(2)$ is the position and orientation of pursuer i with respect to $\mathcal{F}_\mathcal{S}$ and $u_p^i = [v_p^i \quad \omega_p^i]^T$ represents the input to pursuer i. In addition, pursuers are bound by certain kinematic and dynamic constraints. Specifically, we assume that the maximum linear velocity $V_{p\max}$ and angular velocity $\Omega_{p\max}$ of all pursuers is known.

The model of the evader or target agents is given by

$$\begin{aligned}\dot{x}_\tau^j &= v_\tau^j \cos\theta_\tau^j,\\ \dot{y}_\tau^j &= v_\tau^j \sin\theta_\tau^j,\end{aligned} \tag{4.2}$$

where $(x_\tau^j, y_\tau^j) \in \mathbb{R}^2$ is the position of target j, v_τ^j is uniformly distributed in $[0, V_{\tau\max}]$ and θ_τ^j is uniformly distributed in $[0, 2\pi)$. In our experimental implementation, the evader is tele-operated by a human. In this case, uniform distributions may not be the best choice. This human-robot interaction is a topic of further research.

Obstacles may be placed in the game area as long as certain conditions are met. Suppose the j^{th} obstacle obstructs a certain region in $\mathbb{R}^2$ denoted as $\mathcal{O}_j$. Valid obstacles will meet the following conditions: (i) $\mathcal{O}_j$ is a convex region in $\mathbb{R}^2$, (ii) $\mathcal{O}_j \subset \mathcal{S}$, and (iii) the minimum distance from any point $p \in \mathcal{O}_j$ to a point $q \in \mathcal{O}_i$ $(i \neq j)$ or to a point $s \in \partial\mathcal{S}$ is greater than W. Here W refers to the diameter of the smallest circle which can completely surround the largest participating robot's projection onto $\mathbb{R}^2$. This ensures that there are no regions in the game area that can be reached by some robots but not by others.

Let N be the total number of pursuers and M the total number of active targets that are participating in the game. All targets are considered active at the beginning of the game. Once a target is caught, it becomes inactive. Inactive agents must either move to a position where they represent valid obstacles and remain still or be removed from the playing field $\mathcal{S}$. Let p_i $(\tau_i) = [x^i_{p(\tau)} \quad y^i_{p(\tau)}]^T \in \mathbb{R}^2$ refer to the position of the i^{th} pursuer (target) robot with respect to $\mathcal{F}_{\mathcal{S}}$. When there is no danger of confusion, p_i (τ_i) may simply be used to refer to the robot itself as well. The notation $\mathcal{P} = \{p_1, p_2, p_3, \ldots\}$ is used to refer to the set of all pursuers and $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \ldots\}$ refers to the set of all targets. All agents in $\mathcal{P}$ and in $\mathcal{T}$ must have initial positions within $\mathcal{S}$ and cannot leave $\mathcal{S}$ while active. Let e_{ji} be the Euclidean distance from the j^{th} target position, τ_j, to the closest pursuer, i.e., $e_{ji} = d(\tau_j, p_i)$. The pursuer i is said to *capture* the target j when $e_{ji} < \varepsilon$. The threshold value ε is called the *capture threshold* for an interval Δ_c called the *capture timeframe*.

The pursuers receive information about the position of each target intermittently. Let $\sigma_i \in [\sigma_{\min} \quad \sigma_{\max}]$ be a random variable representing the time period between communications for the i^{th} target. At the instant of communication from target i, its exact position within $\mathcal{S}$ is known by the pursuer agent. Following this, the target may continue to move but the sensor agent receives no updated information until the next communication from target i. Based on the previous discussion, the problem in Marco Polo can be stated as follows:

Problem 4.1 *Given a set $\mathcal{P}$ of N pursuers and a set $\mathcal{T}$ of M target robots within a specified game area $\mathcal{S}$ and meeting all of the assumptions outlined in this section,* *choose values $u_p^i = [v_p^i \quad \omega_p^i]^T$ for all pursuers in $\mathcal{P}$ subject to the pursuers' dynamic and kinematic constraints which minimize the time t_c required to capture all targets in $\mathcal{T}$.*

4.2 Experimental Implementation

Marco Polo requires at least two robots. However, the evader robot is relatively simple using only a wireless joystick to set the robot's velocity inputs. Since the pursuer robot is autonomous, its implementation is more challenging requiring an answer to the three fundamental questions in autonomous mobile robotics: 1) Where am I? 2) Where am I going? and 3) How do I get there?

Where am I?

Section 4.1 indicates that the pursuer robot receives information about the location of the evader at a random interval. The Scorpion's encoder-based odometry alone cannot accurately provide this information over long distances. Marco Polo uses the vSLAM^{TM} algorithm that is included with ERSP. The algorithm extracts distinguishing features from camera images to correct odometry error. To ensure that robots communicate in the same reference frame, each use a common vSLAM^{TM} map. Thus, this common map of the game area is generated first. At the beginning of each game, all pursuers wander around the room attempting to determine their positions in the game area by watching for scenes in the map. After the pursuer determines its location, it begins tracking the evader. Evaders also watch for scenes in the map as game participants maneuver them. In summary, the vSLAM^{TM} module provides a method for accurately tracking each robot's location throughout the game even if they start at random locations within $\mathcal{S}$.

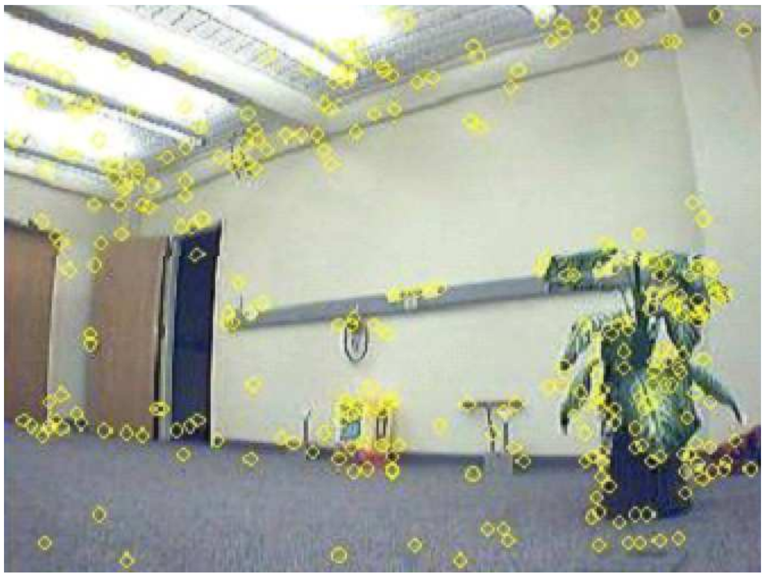


Figure 4.1: Landmark image after vSLAM^{TM} processing.

Where am I going?

This is the question that motivated most of the research related to Marco Polo. However, one key issue here relating to the experimental implementation of Marco Polo is the method used for communication between the two robots. Since both robots have Tablet PC with wireless network interfaces, using TCP/IP over a standard wireless network is simple and effective. Also, both ERSP and Qt provide networking modules that make communication relatively easy. We use the ERSP networking in our implementation.

How do I get there?

Answering this question involves designing a low-level controller that is capable of generating values of v and ω that will maneuver the robot from its current location to a goal location within the game area. Since the goal location depends on the low-level controller that is chosen, a high-level control strategy must take into account the low-level controller that is used or supply the needed control values rather than

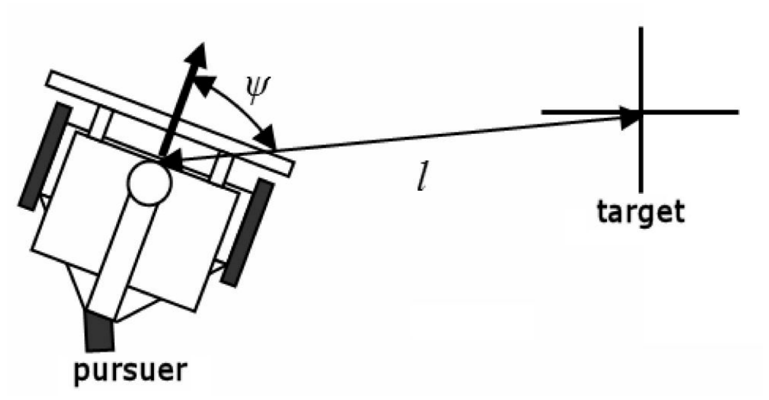


Figure 4.2: Marco Polo controller definitions.

the goal location. The pursuit strategy presented above is designed for use with an attractive potential field controller [59]. Suppose that pursuer i is located at (x_p^i, y_p^i) and must travel to the goal location $r_i = (x_r^i, y_r^i) \in \mathcal{S}$. We define the distance error $\ell_i = d(p_i, r_i)$ as the Euclidean distance between pursuer i and its target as depicted in Figure 4.2. Let $\phi_i = \arctan 2(y_r^i - y_p^i, x_r^i - x_p^i)$ represent the angle of a vector connecting the pursuer to its target. Then, we define the angular error as $\psi_i = \theta_p^i - \phi_i$ where $\psi_i \in (-\pi \quad \pi]$. Based on this notation, the proportional control law is given by

$$u_p^i = K_p \zeta_i, \tag{4.3}$$

where $\zeta_i = [\ell_i \quad \psi_i]^T$ is the error vector and $K_p = \mathrm{diag}(k_v, k_\omega)$ is a diagonal matrix of control constants.

Software Integration

To complete the game, the design techniques discussed above are integrated into a Qt plug-in DLL that may be configured and managed using the Robotic Games Coordinator and Game Manager. The plug-in is designed using four modules including: a vSLAM$^{\text{TM}}$ module, a networking module, a joystick control module, and an autonomous control module. The game referee uses the Coordinator to specify which of the two robots will be the pursuer and which will be the evader. Both robots load the vSLAM^{TM} and networking modules together with one of the control modules. All three modules run in separate threads and communicate using a thread-safe event system that is built into ERSP. The autonomous control design is best described us-

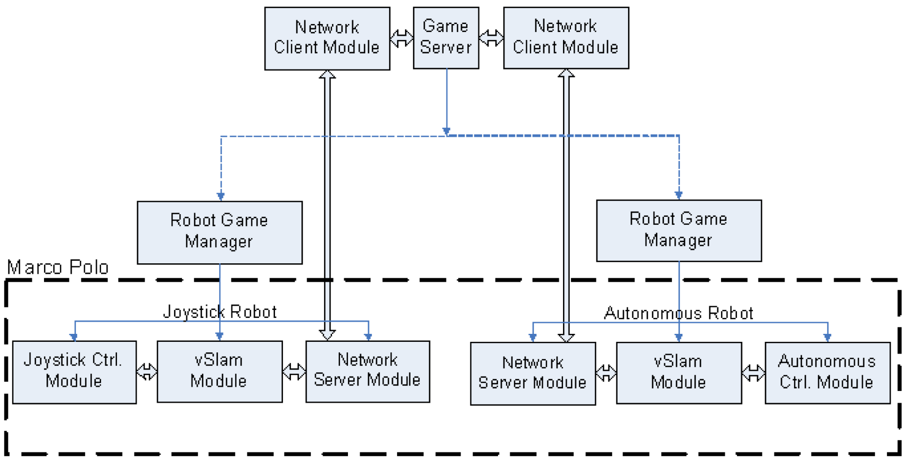


Figure 4.3: Block diagram of the Marco Polo game.

ing the notation of a hierarchical hybrid system [17]. An automaton representation of this system is shown in Figure 4.4. This design consists of two outer states that switch based on the vSLAM^{TM} module's confidence in its location estimation. If this value is below a threshold, the *Localization* state is active. Otherwise, the robot switches to the *Gameplay* state. Each outer state contains three inner states including obstacle avoidance, boundary detection, and goal seeking. The goal in the *Localization* state is simply wandering. This state is implemented using an ERSP behavior and is used to enhance the vSLAM^{TM} module's ability to localize. In the *Gameplay* state, the goal is to approach a goal point that is calculated based on the pursuit strategy as discussed above. When one of the obstacle avoidance or boundary avoidance states is active, the current goal is ignored and v and ω are adjusted to prevent leaving the boundary or colliding with an obstacle.

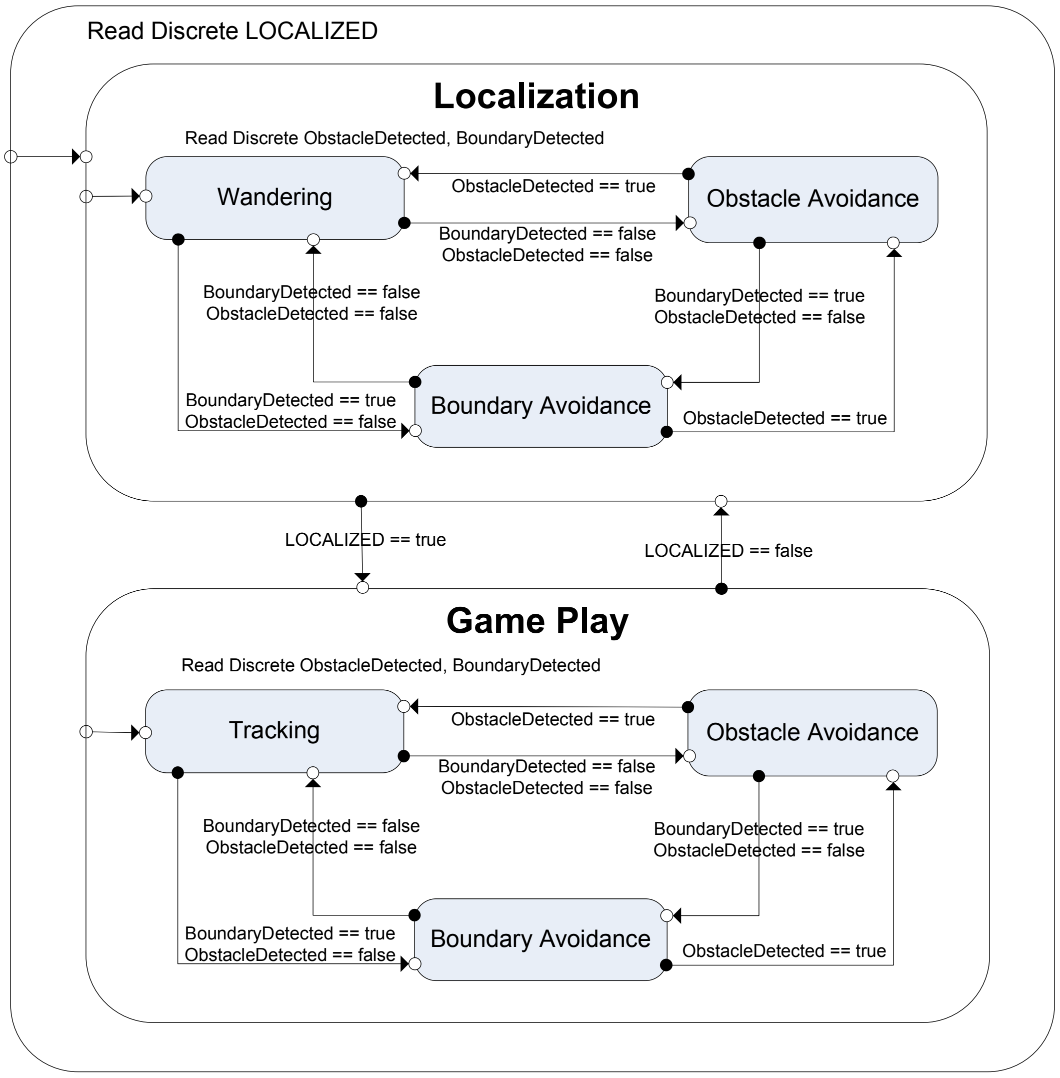

Read Discrete LOCALIZED
Localization
Read Discrete ObstacleDetected, BoundaryDetected
Wandering
Obstacle Avoidance
ObstacleDetected == true
BoundaryDetected == false
ObstacleDetected == false
BoundaryDetected == false
ObstacleDetected == false
BoundaryDetected == true
ObstacleDetected == false
Boundary Avoidance
BoundaryDetected == true
ObstacleDetected == false
ObstacleDetected == true
LOCALIZED == true
LOCALIZED == false
Game Play
Read Discrete ObstacleDetected, BoundaryDetected
Tracking
Obstacle Avoidance
ObstacleDetected == true
BoundaryDetected == false
ObstacleDetected == false
BoundaryDetected == false
ObstacleDetected == false
BoundaryDetected == true
ObstacleDetected == false
Boundary Avoidance
BoundaryDetected == true
ObstacleDetected == false
ObstacleDetected == true

Figure 4.4: A hybrid automaton representing the autonomous control module.

Chapter 5

Geometric Optimization for Detecting and Intercepting Dynamic Targets

In this chapter, we consider a modified version of the Marco Polo problem that is relevant to military, security, and surveillance applications for both ground based and aerial vehicles [60, 61, 62]. The major modification to the general Marco Polo problem is that no communication between targets and pursuers takes place, but the pursuers may obtain only position information about the targets when they enter their field of view.

5.1 Methodology Summary

This problem has been formalized and a solution methodology has been developed in a collaborative effort between the Laboratory of Intelligent Systems and Controls at Duke University and the MARHES Laboratory at Oklahoma State University which includes the author. Details of the methodology may be found in [63]. This chapter summarizes the problem assumptions and methodology that has been presented in these works and describes the differences from the Marco Polo problem presented in the previous chapter.

In this problem, multiple mobile targets enter a rectangular environment containing obstacles at random locations on the environment border and with random heading. Targets are assumed to be constrained to straight line motion and travel with constant velocity as they move through the environment. Multiple sensors are placed in the environment in order to maximize the probability of detecting targets by k different sensors such that the targets track may be hypothesized. This type of detection and tracking scheme using spatially distributed sensors is known as a *track-before-detect* approach [64] and is typical when using proximity sensors which have limited range and are subject to frequent false alarms. By considering detections at different times from k independent sensors, a higher-level controller may be able to declare a target track to be positively identified in order to deploy a sensor to pursue and capture the target.

Target tracks may be classified by the number of detections into three different classes. These classes are given by the following definitions.

Definition 5.1 *An unobserved track is the path of a target j for which there are no detections at the present time, t.*

Definition 5.2 *A partially-observed track is the path of a target that is estimated from $0 < l < k$ individual sensor detections obtained up to the present time, t.*

Definition 5.3 *A fully-observed track is the path of a target that is estimated from at least k individual sensor detections obtained up to the present time, t.*

The user designs the value of the parameter k based upon the reliability and deployment costs of the sensors. In [64] it was found that from a geometric point of view

$k = 3$ is a convenient number of detections for estimating a track in the absence of false alarms. However, in certain surveillance applications the cost associated with capturing a target is very high and, therefore, a higher number of detections may be required.

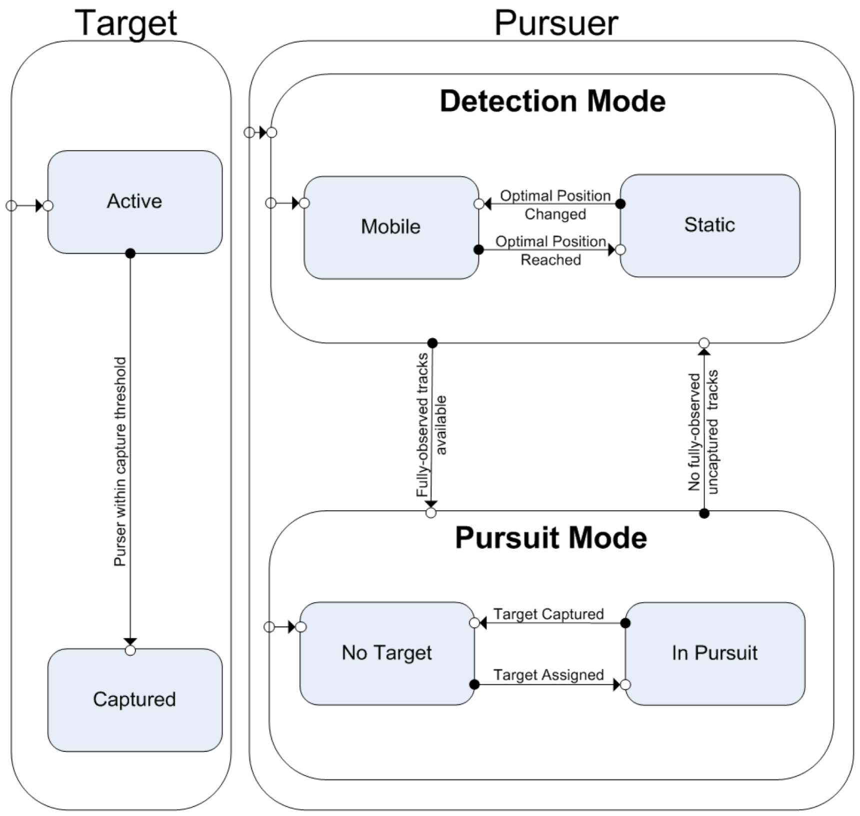


Figure 5.1: A hybrid automaton which models the pursuer and target as hierarchical hybrid systems with discrete states of operation.

The major difference between this problem and that of Marco Polo is how sensors or pursuers switch between *detection* mode and *pursuit* mode as shown in Figure 5.1. In detection mode, a pursuer has three objectives (i) Avoid obstacles; (ii) maximize the probability of detecting unobserved tracks; and (iii) maximize the probability of detecting partially-observed tracks. The objective of a pursuer in pursuit mode is to minimize the time to capture t_c a target based upon its fully-observed track.

The methodology seeks to coordinate the network of sensors in detection mode to address the objectives of these sensors as characterized by its treatment of three areas: information-driven motion planning, probability of detection for partially-observed tracks, and search area coverage.

Information-driven motion planning deals with the issue of moving sensors in the environment for the purpose of obtaining information from multiple targets. The methodology addresses this problem by first performing convex polygonal decomposition of the environment which is a task common in classical motion planning [65]. This decomposition divides the environment into convex regions or cells which contain obstacles (C-obstacles), those which are free of obstacles or C_{free} cells, and obstacle free observation cells which have a non-zero probability of containing a partially-observed target (C-targets). Obstacle free cells which share a boundary are said to be adjacent. From the adjacency of cells, an undirected graph is constructed in which the nodes of the graph represent C_{free} cells that are connected by graph edges to reflect cell adjacency.

The sensor planning objectives are then expressed in terms of a reward function that represents the expected profit of the measurements that would be obtained by moving from a configuration in one cell, $q_i \in \kappa_l$, to a configuration in an adjacent cell, $q_i \in \kappa_\imath$,

$$R(\kappa_l, \kappa_\imath) = B(\kappa_l, \kappa_\imath) - J(\kappa_l, \kappa_\imath), \tag{5.1}$$

where B is the benefit gained by moving between the two cells, and J is the cost. The value of the reward function between adjacent cells becomes the value on the arcs of the connectivity graph. The reward function is defined as

$$R(\kappa_l, \kappa_\imath) = w_1 P_{\mathcal{R}}(\kappa_\imath) + w_2 \Delta P_{\mathcal{S}}^k(\kappa_l, \kappa_\imath) - w_3 d(\kappa_l, \kappa_\imath), \tag{5.2}$$

where $P_{\mathcal{R}}$ is the probability of detecting a target with a partially-observed track, $\Delta P_{\mathcal{S}}^{k}$ is the gain in the probability of detecting unobserved tracks, and $d(\kappa_l, \kappa_\imath)$ is the Euclidean distance between adjacent cells.

Then, the optimal sequence of cells or *channel* that maximizes the total reward of a pursuer in detection mode is

$$\mu^* \equiv \{\kappa_0, \ldots, \kappa_f\}^* = \arg\max_{\mu} \sum_{(\kappa_l, \kappa_\imath) \in \mu} R(\kappa_l, \kappa_\imath). \tag{5.3}$$

Once the connectivity graph has been obtained, the optimal channel μ^* is computed using a graph searching algorithm, such as the A* [65]. This optimal set of cells are then mapped into a set of waypoints which the sensor uses to navigate. Figure 5.2 illustrates the decomposed game area with obstacles, a sensor, and a partially detected target. Figure 5.3 is an example connectivity graph that might be obtained from such an environment.

5.2 Pursuit Strategy

In this section we propose two very simple approaches to pursue a target in the Marco Polo formulation or a fully observed target as described in the previous section. These solutions include (i) moving to the last known target location and (ii) assuming the target is moving in a straight line with constant velocity and intercepting along that line.

The first solution simply instructs the pursuer to move to the last known point of the target. This strategy of capturing the target may be most useful when the pursuer is far from the target and the communication interval is small enough. That is, this strategy would be employed to move the pursuer close enough to the evader

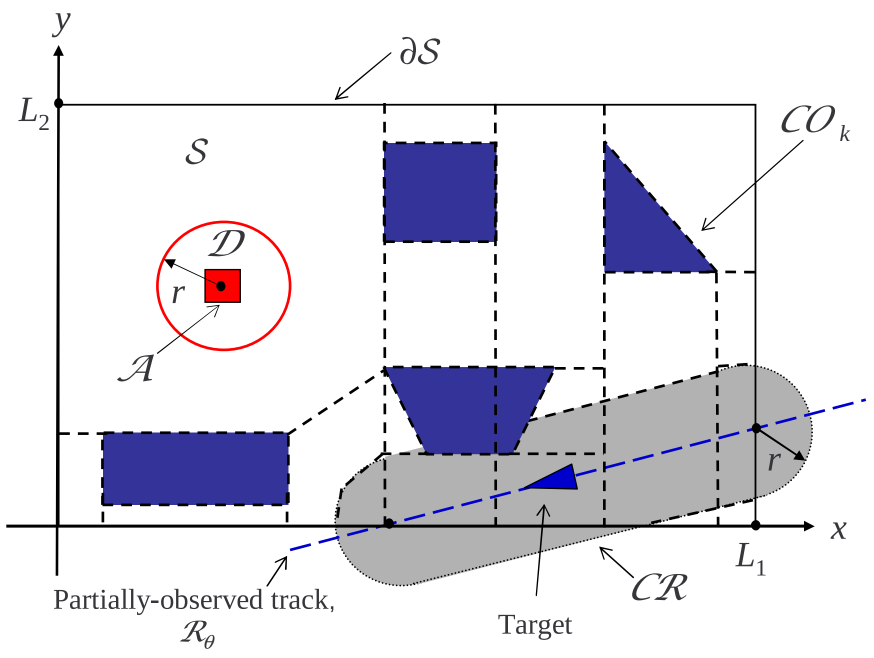


Figure 5.2: Example of cell decomposition (dashed lines) for a workspace with four C-obstacles (dark gray) and one C-target $\mathcal{CR}$ (light gray) corresponding to $2 < k$ detections. One sensor with range r and field of view $\mathcal{D}$ is installed on a robot with a square platform geometry $\mathcal{A}$.

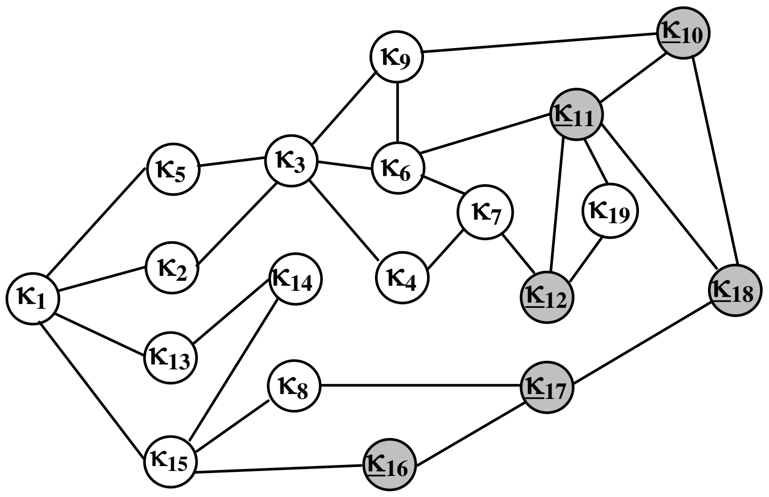

κ_1
κ_2
κ_3
κ_4
κ_5
κ_6
κ_7
κ_8
κ_9
$\underline{\kappa}_{10}$
$\underline{\kappa}_{11}$
$\underline{\kappa}_{12}$
κ_{13}
κ_{14}
κ_{15}
$\underline{\kappa}_{16}$
$\underline{\kappa}_{17}$
$\underline{\kappa}_{18}$
κ_{19}

Figure 5.3: Connectivity graph obtained from the cell decomposition in Figure 5.2, where the cells in the decomposition are numbered from left to right and from top to bottom, and the observation cells are shown in grey.

to select a more sophisticated strategy.

The second approach shown in Figure 5.4 is a simple solution based upon the geometry of the problem taking into account the kinematic constraints of the pursuer. This approach is strongly dependent on the controller used to guide the nonholonomic vehicle to a target waypoint. In this case, the assumed controller is based on the potential field controller (PFC) presented in [59]. The strategy also assumes that the target is moving with both constant velocity and heading. The strategy attempts to intercept the target at a point δ which is a function of the interception time t_c which is the time for the pursuer and target to reach that point. The initial states of the pursuer and target are $p_0 = (x_p, y_p, \theta_p)$ and $\tau_0 = (x_\tau, y_\tau, \theta_\tau)$, respectively. The interception point is defined as

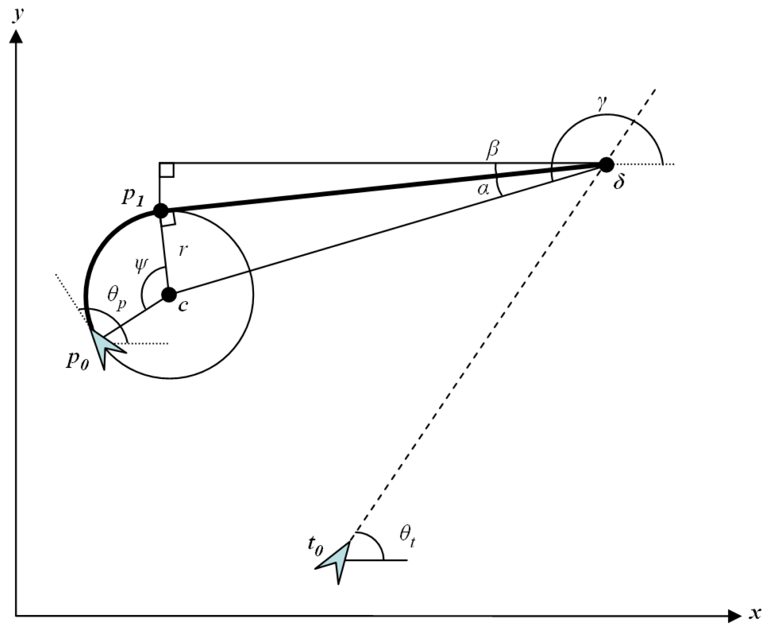


Figure 5.4: Pursuit strategy to capture a target.

$$\delta(t_c) = \begin{bmatrix} x_t + t_c v_t \cos\theta_t \\ y_t + t_c v_t \sin\theta_t \end{bmatrix}.$$

The time to interception is

$$t_c = \frac{r\psi + \|c - \delta(t_c)\| \cos\alpha}{v_p}, \tag{5.4}$$

where the distance traveled by the pursuer is the distance along the arc p_0p_1 plus the straight line distance between p_1 and δ. The arc radius is the same as the turn radius of the pursuer and is defined as $r = \frac{v_p}{\omega_p}$, where v_p is the maximum linear velocity and ω_p is the maximum angular velocity of the pursuer. There are two possible circles corresponding to a right or a left turn of the pursuer. The center points, c_R and c_L, of the circles defined by the turn radius are calculated as

$$c_R = \begin{bmatrix} p_{0x} + r\cos(\theta_p - \frac{\pi}{2}) \\ p_{0y} + r\sin(\theta_p - \frac{\pi}{2}) \end{bmatrix},$$

$$c_L = \begin{bmatrix} p_{0x} + r\cos(\theta_p + \frac{\pi}{2}) \\ p_{0y} + r\sin(\theta_p + \frac{\pi}{2}) \end{bmatrix}.$$

The center point lying closest to the interception point is chosen as

$$c = \begin{cases} c_R, & \text{if } \|c_R - \delta\| \leq \|c_L - \delta\|, \\ c_L, & \text{if } \|c_R - \delta\| > \|c_L - \delta\|. \end{cases}$$

The other parameters for calculating the interception time are calculated as

$$\alpha = \arcsin\left(\frac{r}{\|c - \delta\|}\right),$$

$$\gamma = \arctan(c_y - \delta_y, c_x - \delta_x),$$

$$\beta = \begin{cases} \gamma - \alpha, & \text{if } c = c_R \\ \gamma + \alpha, & \text{if } c = c_L \end{cases},$$

$$p_1 = \begin{bmatrix} \delta_x + \|c - \delta\| \cos\alpha \cos\beta \\ \delta_y + \|c - \delta\| \cos\alpha \sin\beta \end{bmatrix},$$

$$\psi = |\arctan(p_{1y} - c_y, p_{1x} - c_x) - \arctan(p_{0y} - c_y, p_{0x} - c_x)|.$$

The interception point δ in (5.4) which is passed to the PFC may be solved numerically using an iterative algorithm such as that presented in [66] or a Newton method.

5.3 Simulation Results

In order to validate the methodology and pursuit strategy described in Section 5.1, a Matlab simulator has been developed. We integrate the motion planning and control methodologies described in previous sections into simple simulation scenarios.

5.3.1 Pursuit Strategy

The Matlab simulator has been developed for the single pursuer, single target case and is able to use a joystick input such that an intelligent driver may control the target agent either to specify waypoints for later following by the target or to actually steer the target as the pursuer attempts to capture it.

To facilitate comparison of the two pursuit strategies, several basic maneuvers were performed using the joystick and a simple waypoint extraction algorithm [67]. These maneuvers, which are performed by the target in each simulation, include (i) a "figure 8", (ii) a spiral, and (iii) random walk motion. For the simulations, the pursuer and target agents are given maximum linear and angular velocities with the pursuer's greater than the target's. Also, to guarantee that the simulation ends in finite time, the velocity of the target is reduced with time as $v_t = v_{t0}e^{-\frac{t}{\tau}}$ where τ is the time constant and v_{t0} is the target's initial maximum linear velocity. The communication interval of the target's position to the pursuer is a random variable σ whose distribution is uniform in $[\sigma_{\min}, \sigma_{\max}]$. For the simulations, the initial state of the pursuer is $p_0 = [-5 \quad 8 \quad \frac{\pi}{2}]^T$ and the initial state of the evader is $\tau_0 = [0 \quad 0 \quad 0]^T$. Other parameter values used for the simulations are summarized in Table 5.1.

Parameter	Symbol	Value
Capture threshold	Δ_c	0.5 m
Angular velocity	ω_p, ω_t	1 rad/s
Linear Velocity	v_t	0.5 m/s
	v_p	0.55 m/s
Communication interval	$\sigma_{\min}$	1 s
	$\sigma_{\max}$	5 s
Time constant	τ	180 s

Table 5.1: Simulation values.

A total of 100 simulations were performed for each strategy/maneuver pair. Time to capture is the metric used for comparison in each case. Table 5.2 summarizes the results of these simulations. The line interception strategy clearly outperforms

	Figure 8		Spiral		Random	
	Line	Point	Line	Point	Line	Point
Minimum (s)	29.25	39.7	23.95	42.4	32.6	37.75
Maximum (s)	43.25	111.3	68	117.35	91.6	113.95
Mean (s)	31.88	78.64	37.11	74.68	44.13	78
Standard Dev.	1.81	14.76	12.47	14.36	13.61	12.68

Table 5.2: Comparison of simulation results for line interception and point interception strategies.

the more naive approach of simply moving to the last known location. These simulations only present results for a few specific cases. However, they indicate that, in general, the line interception approach is a better solution for target maneuvers containing straight line motion. The full solution to the general problem may be a composition of these and other strategies currently under investigation including a circular interception strategy and an application of the “lion and man” problem [68]. The controller could perhaps be modeled as a hybrid system for switching among the various strategies. The continuation of this research has been left as future work.

This strategy has also been implemented in hardware using the Evolution Robotics Scorpion robotic platform. Figure 5.5 shows these experimental results. The target robot follows a straight line and is quickly captured by its pursuer which moves with a constant linear velocity and a potential field controller which steers it toward the calculated interception point using the method in Section 5.3.1.

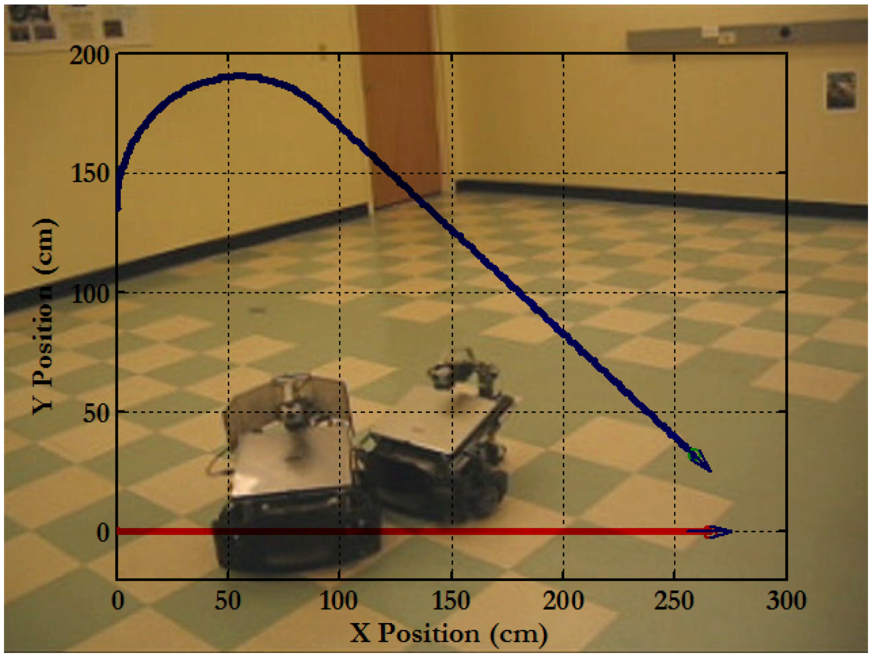
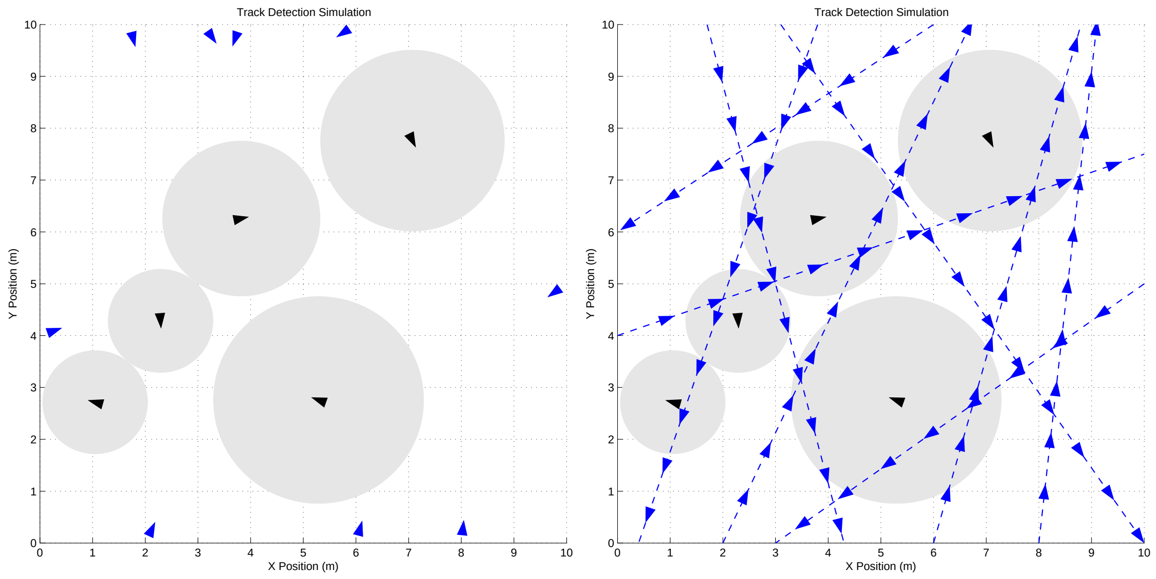


Figure 5.5: Experimental setup: A target is captured by its pursuer.

5.3.2 Scenario 1: Track Detection and Simultaneous Pursuit

We now present three simulation scenarios based upon the methodolgy summarized in Section 5.1. Figure 5.6(a) depicts the search area of the first scenario. The simulated region is 10 m by 10 m. A total of five sensors with various sensing radii (1 m, 1 m, 1.5 m, 1.75 m, and 2 m, respectively) are optimally placed in the search area to maximize the track detection probability and remain fixed. In this case, the number of sensor detections required for a completely detected track is $k = 3$. Targets randomly appear on the border $\partial\mathcal{S}$ of the search area $\mathcal{S}$ with random heading. For the simulation, all targets begin moving at the same time with constant velocity and heading.



(a) (b)

Figure 5.6: a.) The initial target locations and headings and initial sensor placement for the optimal coverage based on the number of sensors, and the sensing radius of each. b.) All target tracks as they move through the search area. Some tracks are completely detected those passing through 3 different sensing regions, others are partially detected less than 3 sensing regions, and others go completely undetected.

As targets move through the environment and through the sensing regions, sensors are assumed to be able to distinguish among different targets as they are detected (Figures 5.6(b) and 5.7(a)).

Once several tracks have been fully-observed, the pursuers switch from *detection* mode to *pursuit* mode and attempt to capture targets on the fully-observed tracks. Figure 5.7(b) shows four fully-observed tracks $\mathcal{R}_i$ and four targets τ_i to be captured.

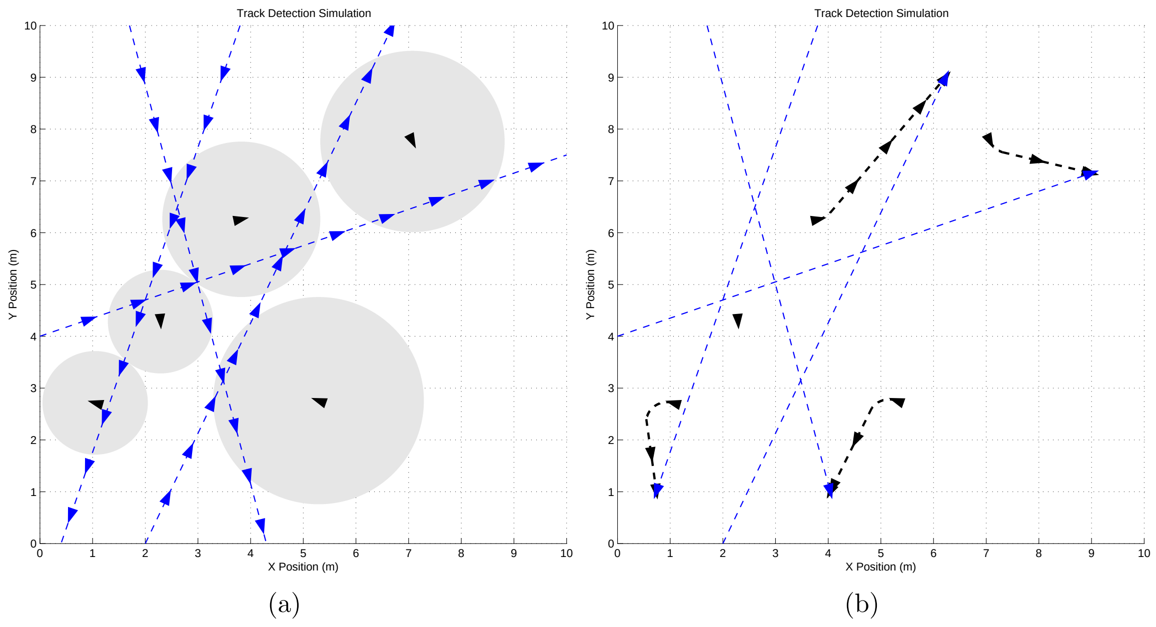


Figure 5.7: a.) Fully observed tracks. b.) Targets are intercepted as the sensors switch from *detection* mode to *pursuit* mode.

Also, five pursuers p_j are available to pursue the targets τ_i. To assign a pursuer p_j to a target track $\mathcal{R}_i$, the Euclidean distance from that target to all unassigned pursuers is calculated and the closest pursuer is chosen. This pursuer p_i is then removed from the set of available pursuers $\mathcal{P}$. These steps are carried out until all target tracks in $\mathcal{R}$ have pursuers assigned or $\mathcal{P} = \emptyset$ indicating that all pursuers are chasing targets. When a pursuer captures a target track, that track is removed from $\mathcal{R}$. If $\mathcal{R} \neq \emptyset$, the set $\mathcal{P}$ is reset to include all pursuers and this algorithm is repeated completely. New target tracks are assigned to all pursuers, including the one that just captured its target.

5.3.3 Scenario 2: Multiple Static Sensors and One Pursuer

In many surveillance applications static sensor networks can be used with a few motion-enabled sensors. Static sensors are placed to optimally cover a given area [32]. If a target (evader) is detected, then a mobile sensor can be sent to investigate or capture the target. This scenario is a special case of the pursuit-evasion problem addressed in this work. The simulation results are depicted in Figure 5.8. This scenario extends the first scenario by including obstacles and the use of the reward function (5.2). The obstacle free path of the single pursuer is calculated from the reward function with $w_1 = 0$, $w_2 = 0$, and $w_3 = 1$ which finds the minimal distance path.

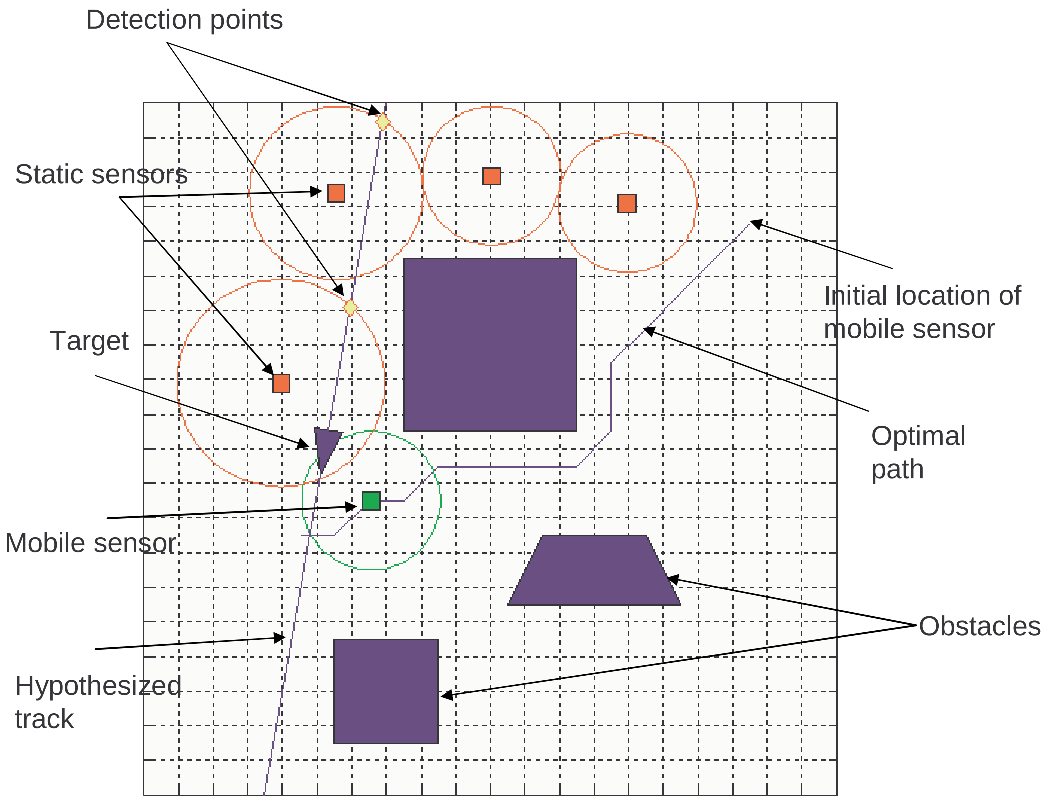
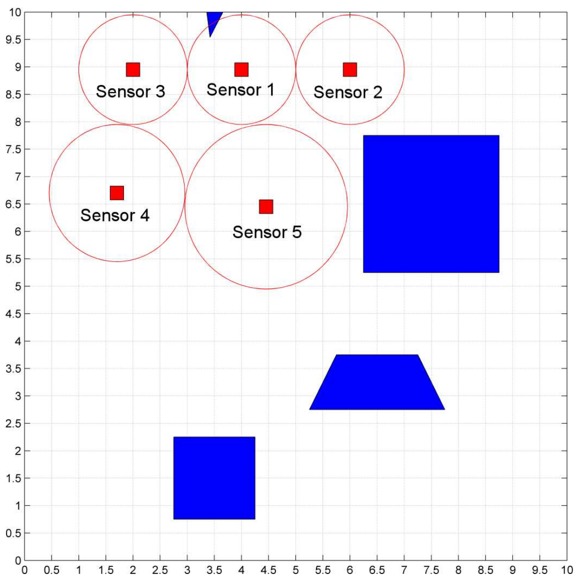


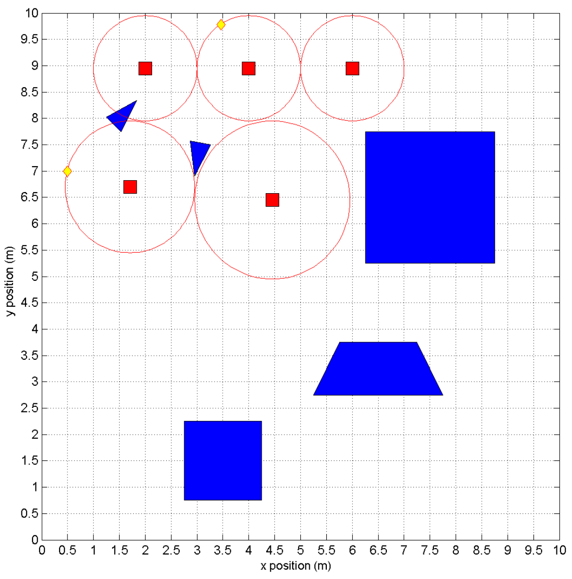
Figure 5.8: Static multiple detectors and one pursuer.

5.3.4 Scenario 3: Multiple Mobile Sensors and Targets

The third simulation scenario extends the second by considering the same environment but with multiple targets and non-zero values for the reward function weights w_1 and w_2 (the detection probability terms) which are $w_1 = 1$ and $w_2 = 1$ in the simulation. With these particular weights, it is possible that sensors closer to targets are not



(a)



(b)

Figure 5.9: a.) Initial sensor placement. b.) Two targets each detected once.

selected to obtain additional measurements. Before the simulation scenario begins, five sensors with platforms measuring 0.25 m square are placed in the 10 m by 10 m environment to maximize the probability of detecting tracks with $k = 2$ since we require this number of detections to form a partially observed track. Obstacle and coverage maps are generated for each sensor corresponding to the placement in each cell. Figure 5.9(a) shows the initial environment and the five sensors - one with sensing radius 1.5 m, one with sensing radius 1.25 m, and three with sensing radii of 1 m. Initially, all sensors are in *detection* mode and each is a candidate to switch to the *pursuit* mode when target tracks become fully observed.

In this scenario, two targets enter the environment at different locations and headings and with different velocities. As they move along their trajectories, they are detected by the sensors (Figure 5.10(b)). The sensors in the network remain motionless since each target has been detected only once. After the second detection of a target, the network hypothesizes the target track based on previous detections and

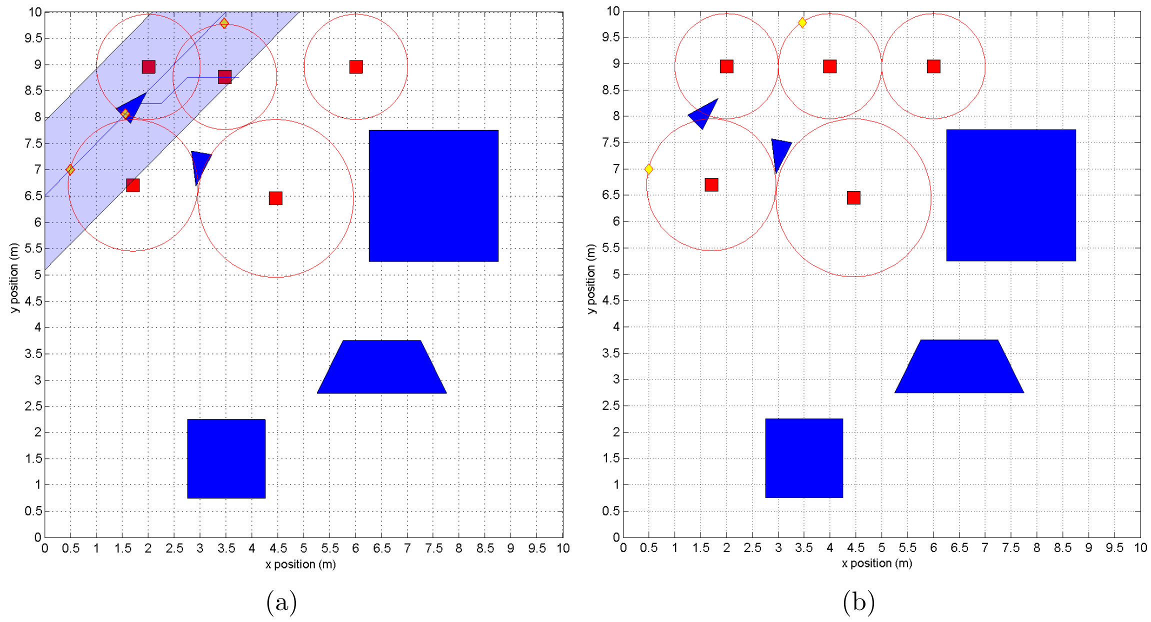
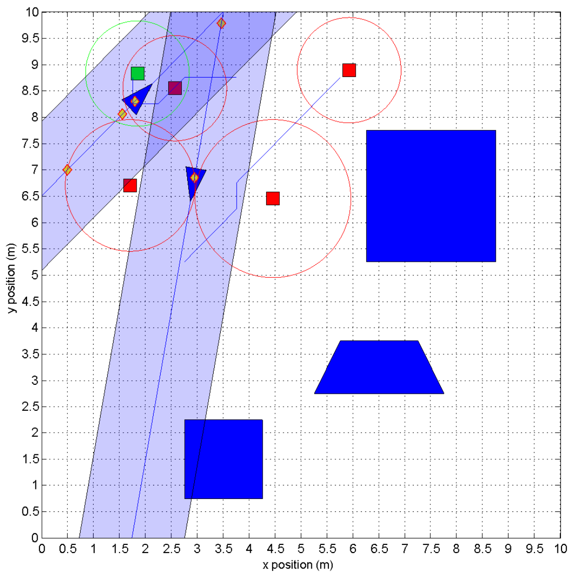


Figure 5.10: a.) Target 1 is partially observed with its hypothesized track, and Sensor 1 is deployed to obtain additional observations. b.) Two targets each detected once.

deploys the sensor which receives the highest reward (or lowest cost) as obtained by the A* graph search algorithm [65] to move to obtain an additional detection of the target (Figure 5.10(a)). When the second target becomes partially detected, the same track hypothesis and sensor deployment occurs. At the point that the first target's track becomes fully observed, the network again evaluates the reward (distance) and deploys the best sensor, denoted by its green color in Figure 5.11(a), to pursue the target. The same pursuit is performed when the second target is fully observed as shown in Figure 5.11(b). The state of the network following capture of all known targets is depicted in Figure 5.12. The network is rearranged to maximize area coverage at the next recalculation interval. Table 5.3 summarizes the chronology of the main events which occur during the simulation. Algorithm 1 illustrates how the simulation scenario has been implemented.


(a)

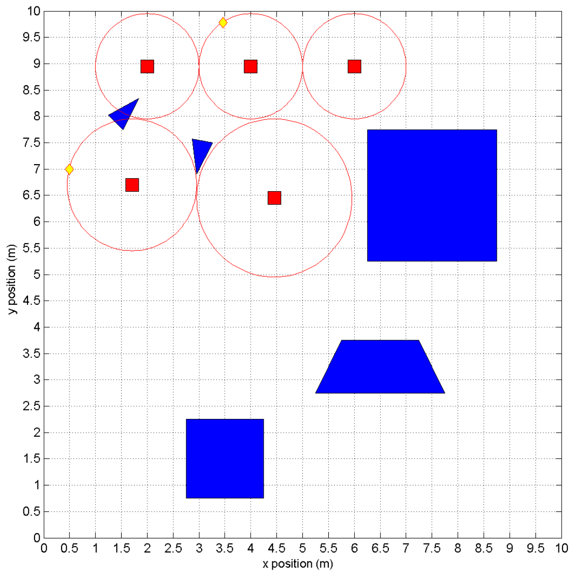

(b)

Figure 5.11: a.) Target 1 is fully observed and Sensor 3 is deployed to pursue it while Target 2 becomes partially observed, and Sensor 1 is deployed to obtain an additional observation. b.) Target 1 has been captured. Target 2 is fully observed and is pursued by Sensor 1.

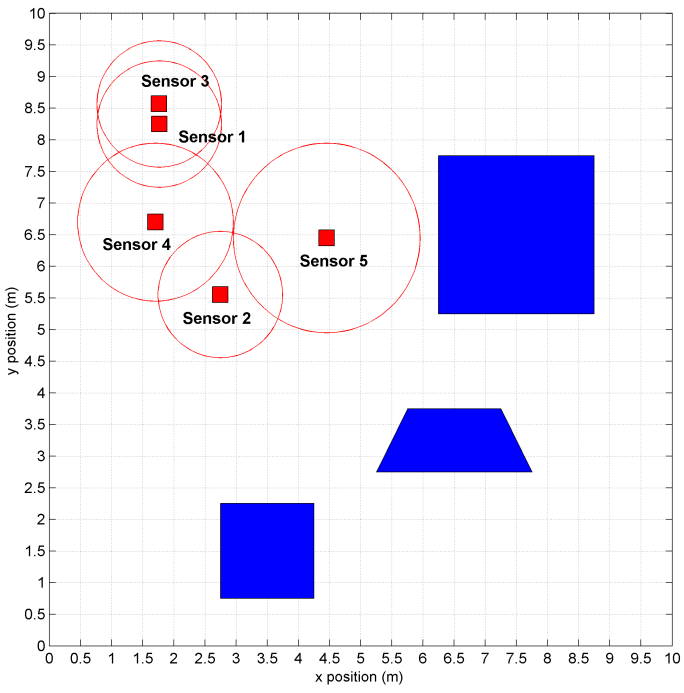


Figure 5.12: Final sensor arrangement after both targets are captured.

Event	Time (s)	Position (m)	Sensor	Target
Detect	0.40	(3.46,9.78)	1	2
Detect	1.70	(0.49,6.99)	4	1
Detect	5.45	(1.56,8.06)	3	1
Deploy	5.45	(1.75,8.25)	1	1
Detect	6.30	(1.80,8.30)	1	1
Pursue	6.30	-	3	1
Detect	6.35	(2.94,6.85)	4	2
Deploy	6.35	(2.75,5.25)	2	2
Capture	6.60	(1.76,8.57)	3	1
Detect	8.55	(2.75,5.77)	2	2
Pursue	8.55	-	2	2
Capture	9.40	(2.75,5.55)	2	2

Table 5.3: Simulation events of Scenario 3.

Algorithm 1 Scenario 3 Algorithm

```
1: Perform initial optimal sensor placement
2: Decompose environment into $\mathcal{C}_{free}$ and $\mathcal{C}_{obstacle}$ cells
3: for all Sensors do
4:    Calculate obstacle map
5:    Calculate coverage map
6: end for
7: while Game not over do
8:    for all Sensors in pursuit do
9:       if Pursued target beneath capture threshold then
10:         Remove target
11:         End pursuit
12:      end if
13:   end for
14:   if Detection then
15:      if Target detections = 2 then
16:         Hypothesize target track
17:         Calculate observation cells
18:         for all Sensors that have not detected this target do
19:            Calculate path and reward to investigate target
20:         end for
21:         Deploy the sensor with the greatest reward
22:      else if Target detections = 3 then
23:         for all Sensors not in pursuit do
24:            Calculate path and reward to pursue target
25:         end for
26:         Deploy the sensor with the greatest reward
27:      end if
28:   end if
29:   if Sensor update interval then
30:      for all Sensors do
31:         Calculate coverage map
32:      end for
33:      Deploy next sensor to maximize coverage
34:   end if
35: end while
```

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have presented a framework for robotics education and research and a new pursuit-evasion game which has been implemented on the platform and in simulation. The Robotic Games framework has been designed as a tool for implementing educational games using multiple robots that may be used for outreach to stimulate the interest of children towards science, technology, engineering, and mathematics. In addition to the framework itself, a number of robotic games have been developed which demonstrate specific robotic behaviors and associated problems. "Robot Jeopardy" reinforces robotics concepts learned and demonstrates an application of computer vision. "Scavenger Hunt" demonstrates teleoperation of a robot in a hostile environment for the purpose of exploration or search and rescue. "Obstacle Course" also demonstrates teleoperation but using a mechanism that may be more intuitive for humans to eventually command teams of robots. Finally, "Marco Polo" demonstrates formation control of an autonomous robot which is able to use computer vision and other sensor to localize, map, and navigate in an environment.

During the development of the framework and games, a number of student groups – including the Cub Scouts, 4-H, and the Native American Sequoyah High School in Tahlequah, Oklahoma – participated in hands-on demonstrations and tests of the games. Most of the students were between the ages of 8 and 12 years and were visibly interested in the demonstrations. Feedback from the students in the form of comments and questions indicated that the robotic games had indeed stimulated their interest and imaginations. In addition to this qualitative feedback, we have received positive quantitative feedback from one the largest groups that participated. This feedback is presented in the Appendix and clearly shows that the students found the demonstrations both educational and enjoyable.

In addition to its educational benefits, one game in particular, Marco Polo, has become the basis of new research in the area of pursuit-evasion games which involve intermittent communication or knowledge about an evader's position. We have presented the formal definition of this problem as well as implementations in hardware and simulation. The methodology developed from the definition of the problem has been summarized and additional simulations have been carried out which serve to validate this methodology. Simulations have been presented in a progessive manner to illustrate the various scenarios that might be encountered. Also, a simple geometric derivation has been presented which allows a nonholonomic vehicle using an attractive potential field controller to intercept a target constrained to straight line motion. The strategy has been implemented on hardware to illustrate its usage in a real scenario.

6.2 Future Work

The Robotic Games framework has been developed using specific, commercial hardware and software platforms. In order to expand the development and usage of the framework, integration with other cost-free or open source projects in the future would allow for additional features to be added. For instance, Player/Gazebo [69] is an open source project which provides a lightweight robot server which runs on the Linux operating system and supports are variety of additional sensors and robotic platforms and includes a 3D simulation environment with realistic physics modeling. Microsoft Robotics Studio [70] is another possible robotic development environment which, like Gazebo, supports a 3D simulation environment and physics engine. Additionally, both Player and Microsoft Robotics Studio provide drivers for interfacing with additional sensors and robotic platforms such as the iRobot Roomba and Create. Usage of these iRobot platforms would allow games to be developed on a low cost, readily available platform.

This thesis has presented the formal definition of the pursuit-evasion problem Marco Polo and has introduced some initial research to address the creation of pursuit strategies in the presence of intermittent communication. However, much additional theoretical development is required to more fully address a solution to the general problem. The methodology developed based on this problem has promising initial results but would benefit from additional research and development. For instance, more consideration should be given to the estimation of partially observed targets. That is, a more sophisticated probability function should be developed rather than the simple binary distribution that is currently used. Also, coverage maps that are calculated for each sensor are dependent on the positions of the other sensors in the network. Because of this, as sensors are deployed, coverage maps are quickly outdated and require recalculation periodically. More analysis and work is needed to find a method which may more intelligently recalculate the coverage function taking into account the strong interdependence of sensor positions.

Appendix A

Supporting Documentation

This appendix includes documents that support the value of the Robotic Games system as an educational tool. We presented a robotics presentation and hands-on demonstration on July 22, 2006 to a group of approximately 45 students between the ages of 9 and 12 years as part of a science camp organized by the 4-H Youth Development Program at Oklahoma State University. As shown in the following pages, our presentation was overwhelmingly favored over the other portions of the camp, especially in terms of its educational value.

OSU
OKLAHOMA COOPERATIVE EXTENSION SERVICE

Oklahoma Cooperative Extension Service
Division of Agricultural Sciences and Natural Resources
Oklahoma State University

4-H Youth Development Program
205 4-H Youth Development Bldg., Stillwater, Oklahoma 74078-6063, (405) 744-5390, Fax (405) 744-6522

Brent Perteet
405 Engineering South
OSU CAMPUS

Dear Brent:

Thank you for presenting an Introduction to Robotics program for the 2006 4-H Science Camp. I believe your program was the highlight of the day. The Robotics program was overwhelming rated as the "Most Educational" program.

The kids really enjoyed learning about all the opportunities in robotics, but most of all I believe they enjoyed the stations and being able to get their hands on the robots and see how they worked. Thank you for putting that all together.

I am attaching a copy of the evaluation for your records. I would also like a copy of the powerpoint you presented if it is available. I would like to use it with our small Lego robots. Also if you have any good photos we could use in promotional activities I would like them as well. I have photo releases on all the kids, so we can use the pictures in brochures, flyers or on the web. Just let us know if you choose to print them.

Thanks again for your time and effort. Please let us know if we can help you in anyway. I am excited you are teaching youth about robotics and trying to reach out past the University.

Sincerely,

[illegible]

Jeff Sallee
Assistant Extension Specialist, 4-H Youth Development
Science and Technology
405.744.8885
jeff.sallee@okstate.edu

Attachment: Evaluation Summary

cc: Dr. Rafeal Fierro

Thanks! I hope
we can work together
again sometime soon.
J

Science Camp
2006 Evaluation Form Summary

Ratings:

Program	1	2	3	4	5	Don't' Know
Agweather	2	4	9	17	13	
GPS	2	6	12	9	16	
W/Q/Seechi Discs	5	2	22	7	11	
Robotics		1	3	10	31	
Overall			4	14	25	1

Which activity was the most educational?

Agweather _7_
GPS _4_
W/Q Seechi Discs _1_
Robotics _35_

One think I will do as a result of what I learned.

Look up websites
Robotics (3)
Don't' Know (3)
Go out and study more
Pay attention to the weather maps
Go to weather sites
Look at weather on computer
Check on weather to see if it is bad
Share what I learned with my friends and family
Nothing
Always use a GPS when hunting
Read more about GPS and Robotics
Engineering
Building a robot
Check on the weather more often and be able to understand what I'm looking at.
Make Robots
Might take Robotics in college
Agweather/Mesonet
Go outside more
Watch the news
Measure water depths
See more Robotics involved things
Go Geocaching
Test my pond's water (2)
Take it to my club and share with them
How you use a GPS
Go to Agweather on the computer
To make robots for fun
Go to OSU and be an engineer…..well I wanted to be one before this. ☺
Look at the weather and build robots
Share my knowledge with my friends and beloved family
Put it in my recordbook
Not to be freaky of Ag Weather
Have more workshops
Buy robots

Which activity did you enjoy most?
Agweather (11)
GPS (7)
Water Quality/Seechi Discs
Robotics (27)
How can Science Camp be improved?
Doing Chemistry
Don't stay outside for 2 hours (2)
Don't walk so much (2)
More computer games
Take out water quality workshop (2)
Be more exciting (2)
Better lunch (2)
More hands on activities (4)
Tell more about livestock
Less walking
Way less waiting
Don't know (3)
I think it's fine
They teach kids stuff
By having fun!
Would like it not to be a busy time of year for Eskimo Joe's & Hideaway
Start 18er!!!
Robotics
If you learn more about the weather and stuff
More clues on the GPS
Visit the veterinarian
Pizza for lunch
None (3)
Too long
2 times a year
More inside stuff
Different things to eat
Less time in classes
More activities outside
By doing more programs and activities and making each one shorter
It can't
Harder hiding spots for the GPS
Write one thing you learned at Science Camp.
Types of Robots
How to make Seechi Discs (2)
How to use GPS (6)
If you keep on a trying you can succeed.
That Robots are very, very, very smart
How to build robots (2)
Get in Robotics
Oklahoma has the most weather towers (2)
Robots use computers as brains
To know to be scared or not
I learned that Robots are used for a bunch of stuff
How to read satellites and radars
Agweather has graphs
Robots (2)

Robotics is hard work
Weather Mesonet
How robots work
The directions on the GPS
I learned to get on the internet and check the weather
It is fun
That OU and OSU actually work together
How to use GPS
How to git on the computer
How to make Seechi Discs
More robots are based on nature
How to read radar
About Robots and GPS
About Robots
I learned about the different kinds of stuff in the water that pollutes it.
Sediment is the highest water pollution
How to make robots
How a robot works
That it is hard for Robots to walk
How to look at the weather
Maybe
How to find objects using GPS
Too much
The way to monitor robots

Would you like to attend Science Camp again?

Yes___33___ No___1___ Maybe __9__

Other Comments:

No GPS
Better lunch next time (2)
Not so long on each class
Choose something besides sandwiches
It was fun (2)
Be more exciting and not so strict
Be more fun (2)
Fun, exciting
No more waiting on anything
The Robotics was the best!!!
Robotics rocks!!
Thank you
I really liked this camp, but it needs more hands-on activities
I would like to thank your helpers!
I know it sounds impossible, but could you let it be overnite next time?
None (4)
Science Camp was so fun! Thank you.
It was awesome
I like Science Camp!!
Thank you, I liked it
The weather thing was fun and so was Robotics
Thank you it was really fun
Great, fantastic!!
Too much walking, need transportation
You people need pencils with with erasers. You need transportation.
I had a great time here.

Bibliography

[1] "Undergraduate engineering gender and ethnicity enrollment trends - some good news, but mostly bad news," Engineering Trends, Houghton, MI, Newsletter 1004B, October 2004.

[2] J. Paik, "Image processing-based mine detection techniques using multiple sensors: A review," *Subsurface Sensing Technologies and Applications, An International Journal*, vol. 3, pp. 203–252, 2002.

[3] L. R. Pasion, S. D. Billings, and D. W. Oldenburg, "Evaluating the effects of magnetic soils on TEM measurements for UXO detection," *Proceedings of the 72nd Annual Meeting of the Society of Exploration Geophysicists*, pp. 1428–1431, 2002.

[4] R. V. Dam, "Soil effects on thermal signatures of buried nonmetallic landmines," *Detection and Remediation Technologies for Mines and Minelike Targets VIII, Proceedings of the SPIE*, vol. 5089, pp. 1210–1218, 2003.

[5] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," *Proceedings 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, 2002.

[6] M. Mataric, “Robotics education for all ages,” in *Proceedings, AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, Palo Alto, CA, March 2004.

[7] B. Argall, Y. Gu, B. Browning, and M. Veloso, “The first Segway soccer experience: Towards peer-to-peer human-robot teams,” Carnegie Mellon University, Tech. Rep. CMU-CS-05-161, 2005.

[8] B. Browning, J. Searock, P. E. Rybski, and M. Veloso, “Turning segways into soccer robots,” *Industrial Robot*, vol. 32, no. 2, pp. 149–156, 2005.

[9] R. D’Andrea and R. Murray, “The RoboFlag competition,” in *Proceedings American Control Conference*, June 2003, pp. 650–655.

[10] E. Matson and S. DeLoach, “Using robots to increase interest of technical disciplines in rural and undersered schools,” in *Proceedings of the 2004 ASEE Midwest Section Meeting*, 2004.

[11] J. M. Tur and C. Pfeiffer, “Mobile robot design in education,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 1, pp. 69–75, March 2006.

[12] D. Cruz, J. McClintock, B. Perteet, O. Orqueda, Y. Cao, and R. Fierro, “Decentralized cooperative control: A multivehicle platform for research in networked embedded systems,” *IEEE Control Systems Magazine*, June 2007, (To appear).

[13] L. Chaimowicz, B. Grocholsky, J. F. Keller, V. Kumar, and C. J. Taylor, “Experiments in multirobot air-ground coordination,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, New Orleans, LA, April 2004, pp. 4053–4058.

[14] Z. Jin, S. Waydo, E. B. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R. M. Murray, "MVWT-II: The second generation Caltech multi-vehicle wireless testbed," in *American Control Conference*, vol. 6, Boston, MA, June 2004, pp. 5321–5326.

[15] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, and G. Dullerud, "Multivehicle systems control over networks," in *IEEE Control Systems Magazine*, vol. 26, no. 3, June 2006, pp. 56–69.

[16] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How, "Coordination and control experiments on a multi-vehicle testbed," in *American Control Conference*, vol. 6, Boston, MA, June 2004, pp. 5315–5320.

[17] R. Fierro, L. Chaimowicz, and V. Kumar, "Multi-robot cooperation," in *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*, S. Ge and F. Lewis, Eds. Boca Raton, FL: CRC Press, 2006, ch. 11, pp. 417–459.

[18] E. B. Martinson and F. Dellaert, "Marco Polo localization," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, Taipei, Taiwan, September 2003, pp. 1960–1965.

[19] T. H. Chung, J. W. Burdick, and R. M. Murray, "A decentralized motion coordination strategy for dynamic target tracking," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006, pp. 2416–2422.

[20] Z. Tang and Ümit Özgüner, "On non-escape search for a moving target by multiple mobile sensor agents," in *Proceedings American Control Conference*, Minneapolis, MN, June 2006, pp. 3525–3530.

[21] T. G. McGee and J. K. Hedrick, “Guaranteed strategies to search for mobile evaders in the plane,” in *Proceedings American Control Conference*, Minneapolis, MN, June 2006, pp. 2819–2824.

[22] V. Isler, S. Kannan, and S. Khanna, “Randomized pursuit-evasion in a polygonal environment,” *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 875–884, 2005.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, March 2004.

[24] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, “Coverage control for mobile sensing networks,” *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.

[25] C. Belta and V. Kumar, “Optimal motion generation for groups of robots: A geometric approach,” *ASME Journal of Mechanical Design*, vol. 126, pp. 63–70, 2004.

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

[27] J. J. Kuffner and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in *IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA, April 2000, pp. 95–101.

[28] S. Martínez and F. Bullo, “Optimal sensor placement and motion coordination for target tracking,” *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[29] A. Ganguli, J. Cortés, and F. Bullo, "Maximizing visibility in nonconvex polygons: Nonsmooth analysis and gradient algorithm design," *SIAM Journal on Control and Optimization*, vol. 45, no. 5, pp. 1657–1679, 2006.

[30] S. Y. Chen and Y. F. Li, "Automatic sensor placement for model-based robot vision," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 1, pp. 393–408, 2004.

[31] R. Vidal, O. Shakernia, J. Kim, D. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, October 2002.

[32] S. Ferrari, "Track coverage in sensor networks," in *Proceedings of the American Control Conference*, Minneapolis, MN, June 2006, pp. 2053–2059.

[33] C. Cai and S. Ferrari, "On the development of an intelligent computer player for CLUE: A case study on preposterior decision analysis," in *Proceedings of the American Control Conference*, Minneapolis, MN, June 2006, pp. 4350–4355.

[34] A. Arsie and E. Frazzoli, "Efficient routing of multiple vehicles with no communications," *International Journal of Robust and Nonlinear Control*, 2007, (To appear).

[35] N. Song and D. Teneketzis, "Discrete search with multiple sensors," *Mathematical Methods of Operations Research*, vol. 60, no. 1, pp. 663–670, 2004.

[36] Z. Tang and Ümit Özgüner, "Motion planning for multitarget surveillance with mobile sensor agents," in *IEEE Transactions on Robotics*, 2005, pp. 898–908.

[37] S. A. Musman, P. E. Lehner, and C. Elsaesser, "Sensor planning for elusive targets," *Mathematical and Computer Modelling*, vol. 25, no. 3, pp. 103–115, 1997.

[38] S. Koenig, C. Tovey, and Y. Smirnov, "Performance bounds for planning in unknown terrain," *Artificial Intelligence*, vol. 147, pp. 253–279, 2003.

[39] N. Rao, S. Hareti, W. Shi, and S. Iyengar, "Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms," in *Technical Report ORNL/TM-12410*, Oak Ridge National Laboratory, Oak Ridge, TN, 1993.

[40] N. Rao, "Robot navigation in unknown generalized polygonal terrains using vision sensors," *IEEE Transactions on System, Man, and Cybernetics*, vol. 25, no. 6, pp. 947–962, 1995.

[41] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unkown environments," *IEEE Transactions on System, Man, and Cybernetics*, vol. 28, no. 3, pp. 316–333, 1995.

[42] J. Leonard, H. Durrant-Whyte, and I. Cox, "Dynamic map building for an autonomous mobile robot," *International Journal of Robotic Research*, vol. 11, no. 4, pp. 286–298, 1992.

[43] S. Thurn, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, pp. 21–71, 1998.

[44] J. C. Latombe, A. Lazanas, and S. Shekhar, "Robot motion planning with uncertainty in control and sensing," *Artificial Intelligence*, vol. 52, pp. 1–47, 1991.

[45] A. Lazanas and J. C. Latombe, "Motion planning with uncertainty - a landmark approach," *Artificial Intelligence*, vol. 76, pp. 287–317, 1995.

[46] E. U. Acar, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *International Journal of Robotic Research*, vol. 22, pp. 7–8, 2003.

[47] H. Gonzales-Banos and J. C. Latombe, "Robot motion planning with uncertainty in control and sensing," *17th Annual Symposium on Computational Geometry (SCG'01)*, 2001.

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

[49] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, pp. 4420–4426, 2003.

[50] V. Boor, M. H. Overmars, and A. F. der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1018–1023, 1999.

[51] M. Qian and S. Ferrari, "Probabilistic deployment for multiple sensor systems," *Proceedings of the SPIE Symposium on Smart Structures and Materials*, vol. 5765, pp. 85–96, 2005.

[52] "ERSP Scorpion application development robot user manual," Evolution Robotics, Inc., Pasadena, CA, Tech. Rep., March 2004.

[53] "ERSP 3.1 robotic development platform user guide," Evolution Robotics, Inc., Pasadena, CA, Tech. Rep., December 2005.

[54] N. Karlsson, E. D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, “The vSLAM algorithm for robust localization and mapping,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 24–29.

[55] B. Perteet, J. McClintock, and R. Fierro, “A multi-vehicle framework for the development of robotic games: The Marco Polo case,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007, pp. 3717–3722.

[56] (2006) Handvu: Hand gesture recognition. [Online]. Available: http://www.movesinstitute.org/˜kolsch/HandVu/HandVu.html

[57] M. Polo, *The Travels of Marco Polo*, M. Komroff, Ed. New York: Modern Library, 2001.

[58] . Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, A. Casal, and A. Baader, “Force strategies for cooperative tasks in multiple mobile manipulation systems,” in *International Symposium of Robotics Research*, Munich, October 1995.

[59] J. Clark and R. Fierro, “Cooperative hybrid control of robotic sensors for perimeter detection and tracking,” in *Proceedings of the American Control Conference*, vol. 5, June 2005, pp. 3500–3505.

[60] J. Reimann and G. Vachtsevanos, “UAVs in urban operations: Target interception and containment,” *Journal of Intelligent and Robotic Systems*, vol. 47, pp. 383–396, November 2006.

[61] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, “Cooperative air and ground surveillance,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–26, September 2006.

[62] D. H. Shim, H. Chung, and S. S. Sastry, “Conflict-free navigation in unknown urban environments,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 27–33, September 2006.

[63] S. Ferrari, C. Cai, R. Fierro, and B. Perteet, “A multi-objective optimization approach to detecting and intercepting dynamic targets in pursuit-evasion games,” in *Proceedings American Control Conference*, New York City, NY, July 2007, (To appear).

[64] T. A. Wettergren, R. L. Streit, and J. R. Short, “Tracking with distributed sets of proximity sensors using geometric invariants,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 4, pp. 1366–1374, 2004.

[65] J. C. Latombe, *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[66] D. Le, “An efficient derivative-free method for solving nonlinear equations,” *ACM Transactions on Mathematical Software*, vol. 11, no. 3, pp. 250–262, 1985.

[67] M. Sarfraz and M. A. Khan, “Automatic outline capture of arabic fonts,” *Information Sciences*, vol. 140, no. 3, pp. 269–281, 2002.

[68] J. Sgall, “Solution of David Gale’s lion and man problem,” vol. 259, no. 1-2. Essex, UK: Elsevier Science Publishers Ltd., 2001, pp. 663–670.

[69] B. Gerkey, R. T. Vaughn, and A. Howard, “The Player/Stage project: Tools for multi-robot and distributed sensor systems,” in *Proc. of the 11th Int. Conf. on Advanced Robotics*, Coimbra, Portugal, June 2003, pp. 317–323.

[70] (2007) Microsoft robotics studio. [Online]. Available: http://msdn.microsoft.com/robotics/

VITA

Brent Perteet

Candidate for the Degree of

Master of Science

Thesis: A MULTI-VEHICLE FRAMEWORK FOR THE DEVELOPMENT OF ROBOTIC GAMES: THE MARCO POLO CASE

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Tulsa, Oklahoma, on April 6th, 1981, son of Jerry and Tracy Perteet. Married to Jessica Marie Perteet.

Education: Received a Bachelor of Science degree in Electrical Engineering from Oklahoma State University in December 2005. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University in May, 2007.

Experience: Employed by the U.S. Army Defense Ammunition Center, McAlester, Oklahoma, 2000–2005.
Employed by Nomadics, Inc., Stillwater, Oklahoma, 2004–Present.
Employed by Oklahoma State University, School of Electrical Engineering as an undergraduate research assistant, 2005, and graduate research assistant, 2006–Present.

Professional Memberships: Phi Kappa Phi Honor Society, Tau Beta Pi Honor Society, Institute of Electrical and Electronics Engineers

Name: Brent Perteet Date of Degree: May, 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: A MULTI-VEHICLE FRAMEWORK FOR THE DEVELOPMENT OF ROBOTIC GAMES: THE MARCO POLO CASE

Pages in Study: 73 Candidate for the Degree of Master of Science

Major Field: Electrical Engineering

This thesis presents a multi-vehicle platform and framework for robotics education and research. The framework has been designed primarily as a tool for teaching children about engineering in general and robotics in particular. The framework is composed of a unique combination of hardware components and software libraries that allow users to easily design and implement sophisticated robotics behaviors. Several example games are presented including "Obstacle Course," "Scavenger Hunt," "Robot Jeopardy," and "Marco Polo." This thesis also introduces "Marco Polo" as a robotics problem that mimics the pursuit-evasion game often played by children in swimming pools. Specifically, the question of finding an optimal pursuit strategy under the condition of intermittent communication is addressed. Finally, a problem related to "Marco Polo" involving a multi-agent sensor network optimally placed in an environment for the purpose of detecting and intercepting intruders is presented together with a proposed solution methodology and simulation and experimental results.

ADVISOR'S APPROVAL: Dr. Rafael Fierro