Hardware Testbed for Collaborative Robotics using Wireless Communication

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Christopher Wilson

Certificate of Approval:

Prathima Agrawal
Ginn Distinguished Professor
Electrical and Computer Engineering

Thaddeus Roppel, Chair
Associate Professor
Electrical and Computer Engineering

John Hung
Professor
Electrical and Computer Engineering

George T. Flowers
Dean
Graduate School

Hardware Testbed for Collaborative Robotics using Wireless Communication

Christopher Wilson

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 18, 2009

Hardware Testbed for Collaborative Robotics using Wireless Communication

Christopher Wilson

Signature of Author

Date of Graduation

Vita

Christopher was born on May 8, 1984 in Montgomery, Alabama to Glenn and Faye Wilson. He graduated from Loveless Academic Magnet Program (LAMP) High School in May 2002 and enrolled at Auburn University in August 2002. After completing a Co-op rotation with the US Army Aviation and Missile Research Development and Engineering Center's Automatic Target Recognition Group, he graduated with a Bachelor of Electrical and Wireless Engineering in December 2006. In January 2007, he entered graduate school at Auburn University.

Thesis Abstract

Hardware Testbed for Collaborative Robotics using Wireless Communication

Christopher Wilson

Master of Science, December 18, 2009
(B.E.W.E., Auburn University, 2006)

59 Typed Pages

Directed by Thaddeus Roppel

Collaborating mobile robots equipped with WiFi transceivers are configured as a mobile ad-hoc network. Search and rescue algorithms are developed to take advantage of the distributed processing capability inherent to multi-agent systems. The focus of this study is to investigate the effect of team size on target acquisition performance with the amount of inter-robot communication as a parameter. A hardware testbed is described which is used to examine these trade-offs in an indoor laboratory-scale test area. Tests involving up to five robots employing three different amounts of communication are performed. The results show that increased communication significantly reduces the overall number of steps to find a target, and that while inter-robot interference is a significant factor regardless of the amount of communication which occurs, more communication leads to less interference.

Acknowledgments

I would like to acknowledge that God has provided great assistance, direction, and comfort during the times it was needed throughout my life. I would like to thank Dr. Roppel for all his time, encouragement, and mentoring during the past two and a half years. I gratefully acknowledge the technical discussions, encouragement and financial support provided by Dr. Prathima Agrawal, director of the Wireless Engineering Research and Education Center at Auburn University. I also thank Dr. John Hung for his time and support as a member of my thesis committee. A significant part of this work was funded by the U.S. Army Research Office through Grant Number W911NF-06-1-0334 under the thoughtful guidance of Dr. Chris Arney. I thank the late Dr. A. Scottedward Hodel for his invigorating love of life and learning and passing on a little bit of that to me while I traveled this earth with him. Finally, I could not have accomplished this with out the love and support of my family and friends.

Style manual or journal used Bibliography conforms to those in the IEEE Transactions.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`, the C, C++, and MATLAB programming languages, and the GTKmm, GTK, GLIB, CIMG, liboctave, AVR ps2lib libraries.

Table of Contents

Nomenclature

Acronyms

ADC Analog-to-Digital Converter

IO Input/Output

IPv4 Internet Protocol version 4

IR infrared

MANET Mobile Ad-hoc Network

MDF medium density fiberboard

SARA-1 The Search and Rescue Algorithm version 1

SARA-2 The Search and Rescue Algorithm version 2

UDP User Datagram Protocol

Terms

Communication Sharing of information between agents.

Cost map Map showing the cost to reach open areas of the local map

Floorplan the actual structure of the environment

Local map large area, low resolution map containing all features the robot has ever known about

Pose position (x,y) and orientation (θ) from the reference axes.

View map small area, high resolution map with features within the robot's sensor range

List of Figures

List of Tables

Chapter 1

Introduction

On August 29, 2005, Hurricane Katrina slammed into the Louisiana-Mississippi gulf-coast region flattening homes, destroying businesses, and destroying levies designed to hold back the waters of the Mississippi River. In the aftermath of the hurricane, people became disoriented. The landmarks they navigated by were completely wiped out.

It took 43 days for the flood waters to be pumped out of New Orleans [1]. These flood waters contained a mixture of raw sewage, bacteria, heavy metals, pesticides and toxic chemicals [2]. Thus, as they were pumped out, the residue that remained caked on the remaining homes and businesses also contained these hazardous materials. In order to assist the cleanup effort, an inventory of what areas were still toxic had to be taken. The only method available was to send humans into the buildings and classify what materials were present for later safe cleanup and disposal. This exposed the inventory workers to the hazardous residue that was still lingering.

Since the flooded disaster area encompassed almost 80% of the city of New Orleans, nearly 144 square miles had to be searched [3] [4]. This extensive search could have been done more safely with a group of robots doing the dirty work of going into unstable houses and measuring using a full array of sensors. Thus a coordinated mapping robotic system could be of great use to keep humans out of harm's way and possibly increase the reliability and accuracy of the mappings.

Similar disaster recovery applications include fire, earthquake, and tsunami scenarios, among others. Each present their own hazards, but all can be made safer for humans by using a collaborating team of robots to search the area.

Another important scenario is in building security. A team of robots can collaboratively and continuously scan an area of interest for intrusions or hazardous materials leaks.

In military applications, collaboration can bring about robustness due to the ability of other robots to continue operating if one is destroyed or compromised. However, this collaboration's usefulness must be weighed against the amount of active communications sent from each robot and the possibility to give away positions.

One of the outstanding research problems in this field is the question of how much communication really needs to occur between robots. This question was previously studied through software simulation [5]. The present work implements a hardware testbed employing algorithms which are as close to the software simulation as possible, for the purpose of comparative analysis. In this work, a team of robots is programmed to explore an unknown area. The software simulation was limited in that it did not account for the presence of noise on any of the sensor inputs, nor did it have a realistic wireless communications channel. In an effort to realistically translate the software simulation results onto a real hardware testbed, the effect of communication and team size are examined. The robots are assumed to have a working localization routine, which on the small scale of the laboratory environment, is a non-trivial issue. Additionally, a map of the environment is given to each of the robots. This is a restriction that is not present in simulation, but one that makes the initial implementation on the hardware platform more feasible. Later work may remove any additional restrictions on the hardware testbed that are not present in the software simulations.

In Chapter 2, an overview of the literature is provided. This is followed by a description of the hardware used for the robots in Chapter 3. The searching algorithm is described in Chapter 4. The software architecture implementation of the SARA-2 algorithm for the experiment is described in Chapter 5. Experimental results are presented in Chapter 6, followed by conclusions in Chapter 7.

Chapter 2

Literature Survey

This chapter introduces a few key areas of research pertaining to robotics in general and with particular interest in inter-robot communication. Sensing methods which pertain to techniques for localization are presented. Next, a method using multiple robots that sense each other to assist in localization is presented. Finally, some literature on communication for the purpose of cooperation or robustness will be presented.

2.1 Sensing for Localization

Robot design entails specifying what kind of interactions with the environment need to occur. What information should the robot be able to gather from the environment? In general, this could lead to the use of any number of sensors from specific gas sensors to ambient light level sensors to radiation sensors. However, if the focus is narrowed to sensors that will assist in localization, then the primary types are: vision, sonar, infra-red, and laser.

2.1.1 Vision

Sim and Dudek present a method of localization based on a vision sensor system using landmark detection and tracking [6]. Initially, an offline map is generated by using an initial pass through the environment to identify and mark areas of high edge

density as possible landmarks. Principal component analysis is used to obtain a lower-ordered description of the landmarks suitable for processing during a learning phase. The learning phase takes the identified landmarks and creates *tracked landmarks*, which are landmarks suitable for tracking over the region of operation. For online localization, the robot extracts candidate landmarks and compares them to the stored landmarks, creating an estimate of position for each extracted landmark. Finally, these position estimates are merged to obtain a final position estimate.

In [7], Se, Lowe, and Little also use a vision-based approach but take into account scale-invariant landmarks. Their technique utilizes a stereo vision system composed of three cameras. The left and top cameras are mounted 10 cm away from the right camera. Unlike Sim's work, which requires an a-priori knowledge of suitable landmarks, Se's work dynamically identifies target landmarks via Scale Invariant Feature Transform (SIFT) and tracks them in near real-time via a linear least-squares method.

2.1.2 Sonar

Elfes maps and determines position using a ring of Polaroid laboratory-grade sonar sensors [8]. A sonar sensor model is developed which takes into account many of the inherent flaws of sonar: wide beam angles, multiple reflects, poor range precision, and detection sensitivity. This model is then used to merge sonar readings on to an occupancy grid of the robot's environment. Also proposed are multiple views for the occupancy grid, which incorporates multiple levels of abstraction and detail for each in a pyramid like structure. The vertical axis represents the level of abstraction with the lowest level being the raw sensor readings, and the highest level being the most abstract features used for navigation. Additionally, each planar level has two axes: resolution and geographical. The resolution axis involves how detailed each map is while the geographical axis involves how much of an area is covered.

2.2 Cooperative Sensing and Localization

Song, Tsai, and Huang use multiple robots to estimate each robot's position [9]. Each robot is equipped with two CMOS cameras and a big colored ball mounted to the top of the robot. IEEE 802.11 is used to communicate between the robots and a central server. Song promotes a combined approach to localization: generate the estimate using nothing but the current robot's sensors, then combine that estimate with the estimate generated from the other robot's sensors. The central server has the role to fuse each robot's information and generate a position estimate of each robot's current position.

2.3 Robot communication

Chen and Li note that the research being performed using multiple robots independently to try to improve tolerance against robot failure in many cases neglects the use of intra-robot communication to improve time to success [10]. Therefore, they look at the use of communication to improve the overall mission success rate and to reduce the average amount of energy consumed by the robots relative to the multiple independent robot case. Their teaming strategy consists of the robots forming a cluster and designating one of their members as the "central coordinator" of the cluster with the other robots termed "members". Each member in the cluster creates a small "mini-map" of it's surroundings and sends this information back to the central coordinator via a binary protocol defined in the paper. The coordinator's job is then to aggregate the information about the environment from the transmitted mini-maps and make decisions about which direction the entire cluster will move next to reach its goal. The authors present results from their testing showing that offloading path planning processing to a central coordinator results in reduced computation and thus

lower power usage per robot. However, they neglect to mention how to pick the central coordinator or what would happen to their algorithm if the central coordinator becomes unavailable.

Park et al investigate the performance gains of a multi-hop, single-channel network for a group of robots [11]. To start their investigation, they look at a group of robots placed in a random maze with the goal of developing a path-planning algorithm that can extract value from the multi-hop nature of their robot's wireless network. They exploit the forwarding characteristics of the relay nodes in a normal multi-hop network to provide additional benefit, most notably, having the relay nodes aggregate their information into the packets they are supposed to be forwarding. Thus, by the time the destination robot receives the information, it will contain the information from the source robot as well as any relay robots that forwarded the packet. Park is able to show that their algorithm reduces average time to escape from a maze in a hardware simulation. Additionally, they show in simulation that with larger numbers of robots, the average time to success is greatly reduced by the use of communication between robots.

However, as far as the author can tell, no paper looks at varying the amount of communication and the team size, which is the focus for this thesis.

Chapter 3

Hardware

The simulations presented in [5], hereinafter denoted as the SARA-1 algorithm , assume each robot has 16 sonar range finders. This would be prohibitively expensive and difficult to interface for many applications, thus a lower complexity solution was designed. There are two key parts to the hardware used in the testing of the algorithm: the experimental field and the robots themselves. This chapter will be divided up in the same way. The experimental field will be described first followed by the robotic hardware.

3.1 Experimental Field

The experiment is housed in an 8 ft. x 16 ft. field designed as a scale-model of an indoor environment consisting of a hallway with three rooms on either side. The field is constructed by using four 4 x 8 foot x 3/4" medium density fiberboard (MDF) panels for the floors. Each board is attached to the others using straight metal brackets and 3/8" screws. The walls are 12" tall and are also constructed from MDF. The walls are secured to each other and to the floor boards using A66 angle brackets. In order to ensure a smooth floor, pieces of poster board are put across every seam and attached using 1" masking tape. The field is shown in Figure 3.1.

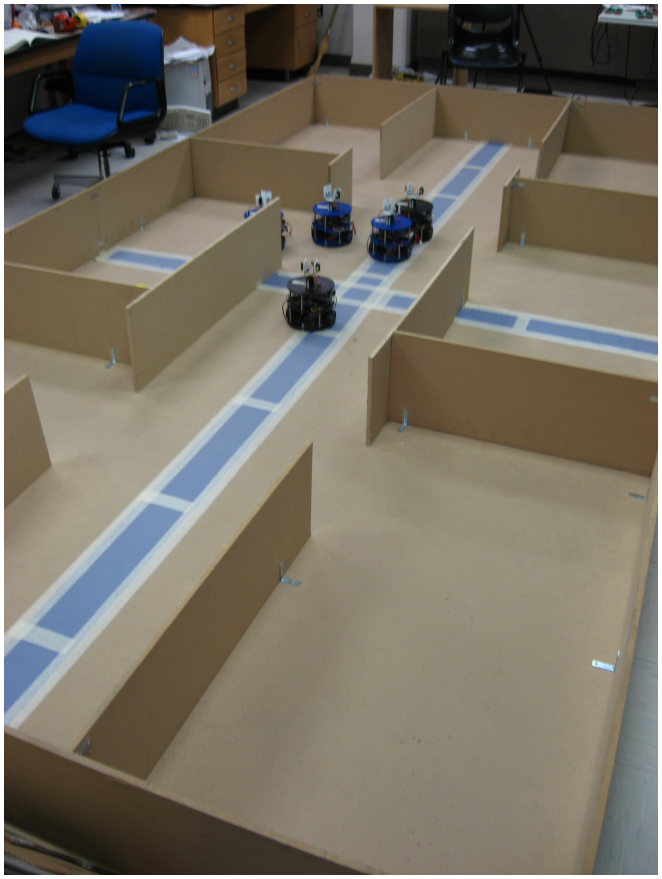


Figure 3.1: Image of the Experimental Field

3.2 Robotic Hardware

This section provides a brief overview of the hardware possibilities for the robots and explains why the specific hardware was chosen. This will be divided into five subsections each describing one specific aspect: Chassis, Drive System, Sensors, Control System, and Batteries.

3.2.1 Chassis

The chassis choice is a tradeoff between platform space, ease of manipulation, cost, and construction time. Initially, a thin PVC was chosen to fabricate the chassis. This was due to prior experience with the material and surplus supplies on hand for

use in making a prototype. The main advantage to this approach was customization; the robots could be made in any shape necessary. Unfortunately, this approach was also very time consuming as each piece had to be individually cut, drilled, and bent, consuming valuable time that could be spent on implementing the software for the robots.

A search for cheaper and faster alternatives led to the discovery of the BudgetRobotic's ScooterBot II chassis kit. This kit consists of a 7" diameter round chassis assembly designed for differential drive steering. It consists of the base platform with cut-outs for the drive wheels and an upper platform for housing the computer and interface board. A third platform was used to serve as a base in mounting the rotating sensor turret described in Section 3.2.3.

3.2.2 Drive System

The selection of a drive system is a bit more straightforward as there are two main choices: legged and wheeled vehicles. Legged vehicles are better suited for traveling over rough or unknown terrain. Wheeled vehicles perform better on smooth level terrain and can usually carry more weight and travel at faster speeds than a legged vehicle. Since this is a laboratory grade experiment in an indoor environment, a wheeled design was chosen.

The next decision confronting the design is what type of wheeled system: tracked, differential or car-style drive system. Tracked vehicles can successfully navigate rougher terrain better than other wheeled vehicle types. They also have the advantage of being able to turn inside their vehicle diameter. However, these advantages come at the complexity of maintaining the tracks.

Car style drive systems have two main drive motors with one steering motor that can drive one or two wheels. These systems are not normally used in robotic applications.

Differential drive vehicles, like tracked vehicles, also have the advantage of being able to turn inside their vehicle radius. Additionally, they do not suffer from the complexities of maintaining the track system.

Since the SARA-1 algorithm simulations used Pioneer-based simulation robots, a differential drive approach was preferable to maintain similarity with previous work.

The next design decision was what type of motor, DC motors or continuous rotation servos. DC motors usually have the advantage of accepting a wide variety of voltages, but have the disadvantage of requiring some interface circuitry to connect to the computer system.

Continuous rotation servos have the advantage of accepting logic-level inputs from the computer system and automatically transforming those commands into motor movements. However, this advantage comes with the cost of usually being less powerful than a comparable cost DC motor.

The original prototype used a DC motor interfaced with an H-bridge. The DC motors were 175RPM 7.2V DC motors with an attached gear reduction system providing 50:1 reduction. The H-bridge was a TI L293NE quad half H-bridge chip to convert the logic-level signals from the micro-controller and drive the motors. Each motor was connected using two half H-bridges to allow forward and reverse movement. The L293NE chip was connected to the unregulated battery to provide maximum power. The wheels used with these motors were 3" diameter neoprene foam tires.

This system worked well and was successfully deployed on the prototype robot. However, with the discovery of the chassis kit, which included continuous rotation

drive servos, wheels, and mounting equipment at comparable cost to just the DC motor drive system, the prototype's drive system was scrapped in favor of the kit's.

The kit uses GW Servo S03NXF STD continuous rotation servos which allows speed control based on standard servo timing parameters. These motors weigh 59.6 grams and produce 35 oz-in of torque. They can rotate at a maximum speed of 60° in 0.15 seconds at 4.8 Volts. The wheels used have a diameter of 2.5 inches, giving the robot a maximum speed of 22.16 cm/s. A linearized velocity equation can be determined with input being the servo timing signal and output is wheel linear velocity:

$$v(t) = \begin{cases} 44.32 * (t - 1.5) & 1 \geq t \leq 2 \\ -22.16 & t \leq 1 \\ 22.16 & t \geq 2 \end{cases} \tag{3.1}$$

3.2.3 Sensors

The SARA-1 algorithm uses two main types of sensors: distance and odometric. The distance sensors are used for observing the world around the robot and properly plotting a course through it. The odometric sensors are used to determine the robot's position in the world. Thus, this section is subdivided into two parts: one part on distance sensors, one part on odometry sensors.

3.2.3.1 Distance Sensors

The SARA-1 algorithm uses sonar sensors as the main distance sensor. Thus the first choice we made was to use some sonar sensors that we already had experience with, Parallax PING))) sonar sensors.

The Parallax PING))) Ultrasonic sensor is a sonar with a three wire interface: power, ground, and signal. The PING))) sensor has its own processing on-board such that the input to the sensor is a pulse and the output is a duration-varying pulse. The input pulse is a five micro-second pulse, that triggers the sensor to start measurement. 750 μs later, the return pulse starts. The duration of the return pulse is proportional to the distance measured from the sonar, ranging from 115 μs to 18 ms. The sound wave used is a 40kHz wave lasting 200 μs. [12]

There is one major problem with this approach. The SARA-1 algorithm uses 16 sonar sensors to plot out a 360° degree view of its surroundings. Since each sonar sensor is $25, each robot would have $400 in sonar sensors alone. Other concerns include the possible interference effect of multiple robots with 16 active sonars, and the complexity of interfacing such a large number of sensors to the onboard microcontroller. So through the combination of complexity and cost, this solution was ruled unacceptable.

Through searching for possible solutions, the idea of rotating two sensors 180° was developed. This would allow for complete 360° coverage as desired by SARA-1, but at a fraction of the cost and complexity. The associated cost would now be two sensors, a servo, and some mounting hardware. Additionally, the possible interference risk would be greatly diminished.

This solution worked well. The sensors were mounted on a turret and the turret was mounted onto a servo. The servo then varied its position from 0° to 180°. It was decided that an increment of 15° should be used for each iteration of the servo. This would make each sensor scan 12 different positions for an effective 24 sensors on the robot.

However, further tests revealed a crucial flaw in the sonar sensing approach. In our environment, the sonar sensors suffer greatly from specular reflection, where the

sound wave would reflect off the wall and not return enough energy back to the sonar detector. They could not detect a wall at an angle of incident greater than 30°. Additionally, when the sensors were placed a medium distance from a corner, the sonar wave would bounce from one side of the corner to the other and then return giving a resulting distance estimate much greater than the actual distance. As the robot moved closer, this problem was not as pronounced, but the robot should not be "near-sighted". This effectively meant that, in our rectangular test field, sonars could not be counted upon to give reliable and accurate readings.

This led to a search for a new type of sensor that did not suffer the corner-echoing phenomena and would still be reliable at angles of incidence greater than 30°. The sensor that was eventually found was the Sharp GP2D12 infrared (IR) distance sensor. These sensors output an analog voltage that is linear with the inverse distance. They have a minimum range of 10 cm and a maximum range of 80 cm. Each sensor should have a filter capacitor from the IR signal line to ground to short any high frequency noise.

Since they produce an analog voltage, a calibration routine should be performed on them. This calibration routine included attaching both sensors to the robot and turning the turret such that one sensor pointed at the target range. The target range consisted of two boards of MDF left over from the field's wall construction. One board is placed flat on a table and marked with lines every 5 cm. The other board is then placed vertically on each line, serving as the target for the IR beam of light. A test program is then loaded on the robot which runs the analog-to-digital converter (ADC) and logs the value of each reading. The target is then slid from 10 cm to 60 cm using 5 cm increments and taking 10 samples of the ADC value. This is done for each sensor.

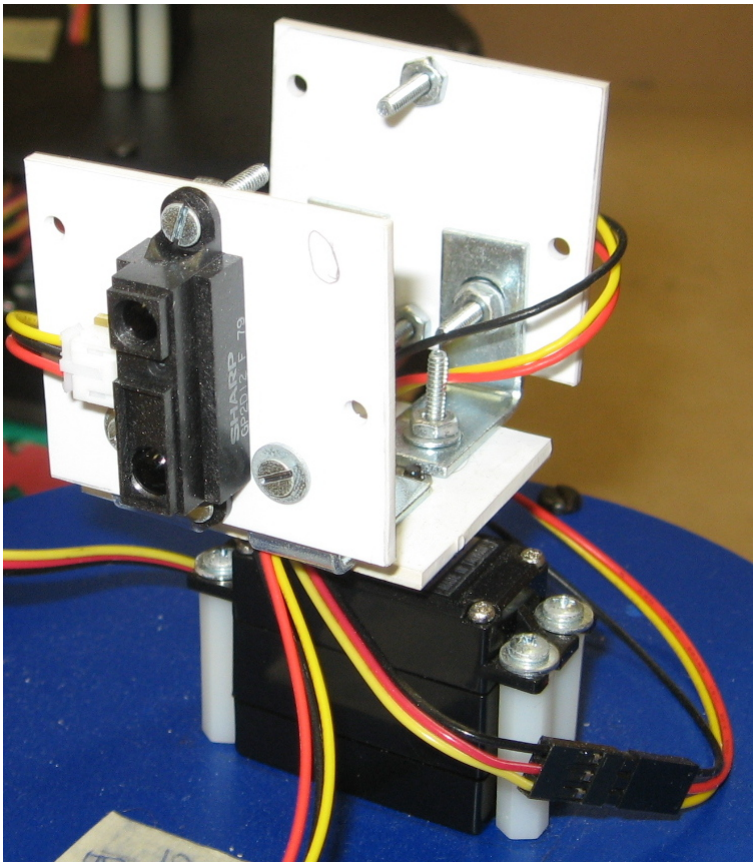


Figure 3.2: IR sensor turret assembly.

Next a curve fit line is extracted using Matlab's `polyfit`. In order to maintain accuracy, a second order curve was chosen over a linear fit.

A picture of the completed IR distance sensor turrert can be found in Figure 3.2.

3.2.3.2 Odometry Sensors

The two types of odometry sensors are wheel encoders and floor encoders. Wheel encoders are typically used in robotic applications and simply measure how far a wheel turns. The advantage with this system is that it is very simple and the accuracy can be adjusted by having different encoders. The main disadvantage with this system

is that it can not tell the difference between when the wheel is moving the robot or when the wheel is slipping.

Floor encoder systems measure the movement of the robot with respect to the floor. This system's main advantage is that it does not suffer measurement errors from the slippage of the wheel and can be very accurate. The disadvantage is that these typically have tight tolerances on mounting distance to the floor.

For this experiment, a floor encoder system based on using optical mice was selected. The optical mice have the advantage of converting movements of the surface to δx and δy. The mice also have varying resolution from 1 counts per mm to 8 count per mm. Due to the inexact nature of the mounting, a calibration procedure from Bonarini et al in [13] is followed to mitigate any systemic errors. An image of the optical mice mounted to the robot can be found in Figure 3.3.

3.2.3.3 Target Sensors

For simplicity, target sensors are simulated in software. The location of the target is programmed into the code, which then has the responsibility to tell the robot when it has found the target and where it is in relation to the robot.

3.2.4 Control System

In order to control the robots, the Gumstix platform was recommended based on prior research and use. This platform has a few major advantages. It runs embedded linux, has an 802.11b wireless card expansion module called Wifistix, and has a micro-controller based expansion module called Robostix.

The Gumstix Connex 400xm was chosen to allow the use of both the Wifistix and Robostix. This processor is a 400 MHz Intel X-Scale ARM based chip with 64MB of RAM and 16MB of flash. The one major limitation that eventually prompted a

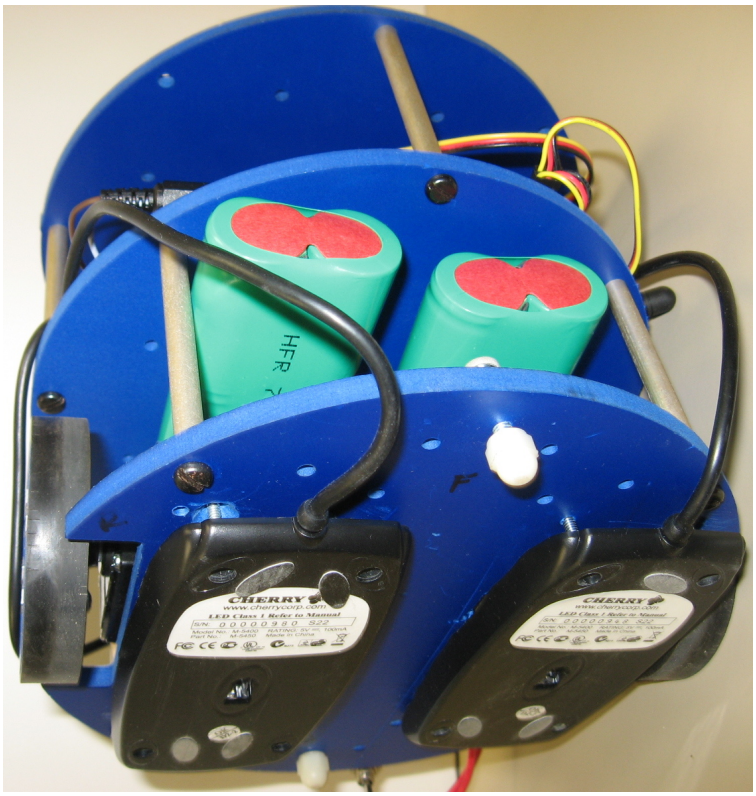


Figure 3.3: Mounted optical mice and batteries.

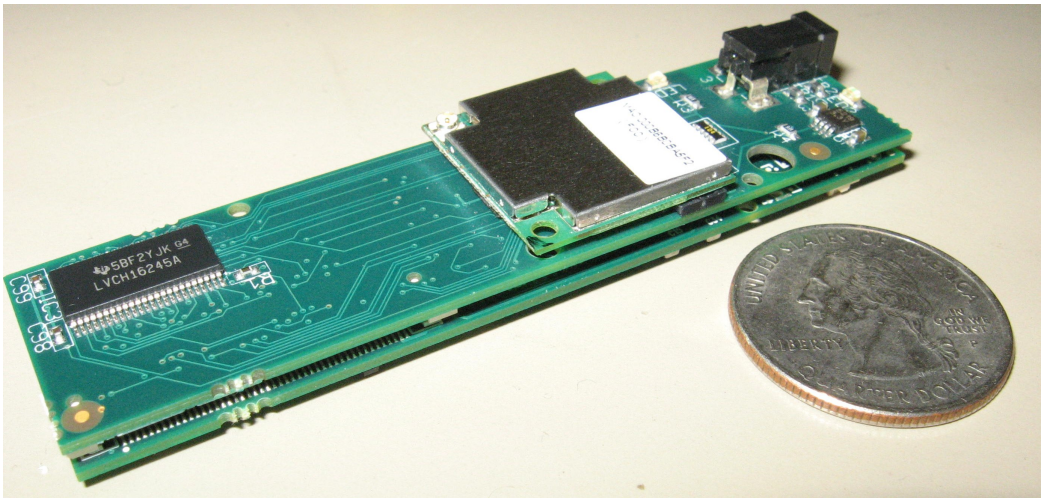


Figure 3.4: Wifistix (top card) and Gumstix (bottom card).

major redesign of the code is the lack of a floating point unit. More information can be found in the software discussion in Chapter 5.

The Robostix module contains an AVR ATmega128 micro-controller designed to drive IO and serve as either the main brains of a robot or serve as interface glue between the Gumstix and the hardware. In this particular application, the Robostix serves as the interfacing glue, allowing the Gumstix to send commands to the Robostix which turns those commands into motor commands. The Robostix is also responsible for interfacing with the sensors and sending that data back to the Gumstix.

The Wifistix is based on the Marvell DRCM81 module from Wistron NeWeb Corporation with the Marvell 88W8385 chipset [14]. The chipset supports 802.11b/g in both infrastructure and ad-hoc connection modes. An image of both the Gumstix Connex and the Wifistix can be found in Figure 3.4.

The other part of the control system is a desktop computer workstation. This is used for logging all of the communications during experimental runs, as well as giving the robots on their initial positions and computing the poses of the robots.

3.2.5 Batteries

To power the robots, the decision was made to use normal RC car batteries. The selected batteries provide 7.2V at 2800 mAh. These were primarily chosen due to familiarity and reliability. Two batteries are used per robot. One battery powers the electronics, the sensors and the microcontroller stack, while the other powers the three servos, two drive servos and one turret servo. An image of the batteries mounted on the robot can be found in Figure 3.3.

Chapter 4

The Search and Rescue Algorithm - Version 2

The Search and Rescue Algorithm version 2, SARA-2, is an outgrowth of SARA-1 [5]. Whereas SARA-1 was developed for simulation, SARA-2 is intended for application to real hardware. During initial testing of the SARA-1 algorithm as implemented on the hardware, many issues were identified that required modification of the algorithm. This chapter will focus on those necessary modifications to the SARA-1 algorithm. An overview of an initial version of SARA-2 with limited success with a physical single robot can be found in [15]. This work extends [15] to the multi-robot case. First, the underlying assumptions of SARA-2 will be presented. Second, each phase, or behavior, of the algorithm will be presented.

4.1 Assumptions

SARA-1 utilized many assumptions, including a few unrealistic assumptions which may be acceptable in simulation but break down upon implementation in actual hardware. The revised assumptions used in SARA-2 are:

1. All maps have a predetermined size and resolution.
2. Each robot knows the floorplan of the arena.
3. Each robot knows its starting pose in reference to some defined point on the map.
4. All motion and sensing takes place in a two-dimensional plane.

5. Communication is not guaranteed to be available and communication can fail, without the robot being informed of it.

6. All obstacles are treated as boundaries. Openings are assumed to be at least 2 robot widths.

7. The target does not move.

Assumption 1 is useful for initializing map memory requirements at compile time and reducing memory management overhead. Assumption 2 is required to assist in small-scale localization procedures. Assumption 3 is used to enable efficient and fast map merging between robots during communication steps. Assumption 4 is to reduce processing and memory requirements and is a reasonable assumption made often in the literature. Assumption 5 is made due to the random nature of the unreliable wireless communication channel. Assumption 6 is made currently for reduced complexity during the path planning and following behaviors. Assumption 7 is made to reduce the need to continually explore previously explored spaces.

4.2 Behaviors

Each robot action falls under a category of behaviors. While these behaviors are typically independent of each other, they serve to complement the other actions and allow the algorithm as a whole to work. Each behavior could also be considered a mode or state of the robot. This is reflected in the software for the Gumstix which is described in Section 5.3. There are eight different behaviors for each robot. Many of these behaviors flow after each other automatically, but some are optionally triggered by user oversight. These modes are: Robot initialization, map-building, error correction and localization, map-merging, path-planning, path-following, obstacle-avoidance, and communication. The algorithm will automatically

run error-correction, map-merging, path-planning, obstacle-avoidance, path-following immediately after the map-building behavior is finished.

In the SARA-2 algorithm, it is useful to consider a series of behaviors as a *step*. Each step consists of the map-building, error correction, path-planning, and path-following behaviors. Thus, one step will involve the robot building a map, correcting sensor errors, planning and following a path to the next destination point. Then the process will repeat with the robot performing another step and so forth until the target is found.

4.2.1 Robot Initialization

The main requirement for SARA-2 to work is accurate knowledge of the initial pose so that all robots can correctly merge their maps during communication stages. Thus before the robots can search, the operator is required to tell each robot their initial pose. Additionally, the map information must be known at run time. In the current implementation, the map information is incorporated at compile time.

4.2.2 Map-Building

Upon initialization of the robots or arriving at the destination point, the robot begins the map-building behavior. This involves sweeping the distance sensor turret assembly multiple times and obtaining an average distance reading for each angle at which the turret assembly stops. These distance and angle measurements are then plotted on a *view map* which represents the area immediately around the robot. An example view map can be found in Figure 4.1(a). Since the returned sonar data will have sensing errors in it, the algorithm extracts line features from the individual sonar data in the next behavior.

4.2.3 Error-Correction and Localization

One of the dramatic limitations of the SARA-1 simulations is the inability to model noise during the simulation runs. Even a small amount of noise in sensor readings can result in a very inaccurate mapping.

There are many sources of errors in realistic hardware deployments. Futhermore, many of these sources have parameters that vary over time. Bearings slowly wear over time; lighting conditions can change from room to room; sonar signals can bounce off a corner and give wildly inaccurate and varying data; varying levels of sunlight can affect infrared sensors; wheel encoders can measure distance including slippage; and optical encoders can return noisy measurements. All of these sources of errors degrade performance of the SARA-1 algorithm in practice.

Thus methods must be developed to counteract the major sources of errors. In experimental tests with SARA-1 deployed on robots, the greatest error occurred in the angle of orientation of the robot. The distance-traveled sensors were properly giving the correct distance traveled; however, due to a combination of measurement and quantization noise, the resulting angle was often highly inaccurate. This inaccuracy would only compound as each robot moved around the experimental field. Consequently, the map became unusable in just a few short steps.

Since SARA-1 relies so heavily on exact position information for the merging of the maps, localization becomes the Achilles heel of the algorithm. Thus in an effort to reduce the orientation error, a restriction is added: the field's walls are required to be known. This allows the angle of the walls with respect to the robot to be determined via the mapped data from the IR sensors mounted on the turret.

4.2.3.1 Map Correction

In order to reduce the amount of error in the robot's orientation angle, the view map is further processed to extract line features that are then compared to the expected position. In order to extract line features, line segments are created between every distance sensor reading. Then a processes of merging close line segments is performed. If two line segments share a common point, then the if the angle between the two line segments is less than 45°, the line segments are merged together. This merging operation is performed for all line segments in the view map. An example of the extracted lines can be found in Figure 4.1(b).

Then, the remaining line segments are parameterized via a ρ and θ form:

$$\rho(\theta) = \left(x - \frac{dimx}{2}\right)\cos(\theta) + \left(y - \frac{dimy}{2}\right)\sin(\theta) \tag{4.1}$$

where $dimx$ and $dimy$ are the x and y size of the view map image, respectively.

The parameterized form of the lines are then compared with the known wall lines, which are parameterized in the same way, inside a Kalman filter described in Section 4.2.3.3

4.2.3.2 Odometry Correction

The other major source of error comes from the odometry sensors. Since these sensors are integrated over the entire movement phase, small variations can quickly sum into a large error. Thus, in addition to calculation of the expected pose that the robot is currently located at, the variance of the pose estimate is also calculated. Should the pose's variance exceed a certain threshold, the robot is stopped and a localization procedure is performed. The localization procedure entails another set

of map-building and error correction phase which are merged via the Kalman filter described in Section 4.2.3.3.

4.2.3.3 Localization

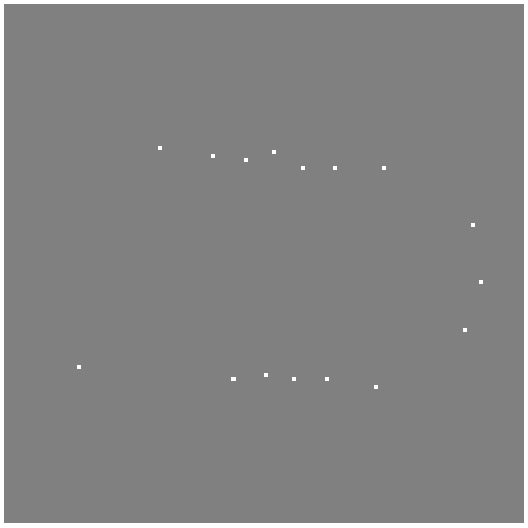
In order to obtain an optimal position estimate, the odometry sensors and line-feature information has to be fused. This is performed via a Kalman filter as discussed on pages 186-190 and 227-244 in [16].

After the Kalman filter produces a pose estimate and associated variance, the known wall lines which match with observed lines are then plotted on the view map, as shown in Figure 4.1(c).

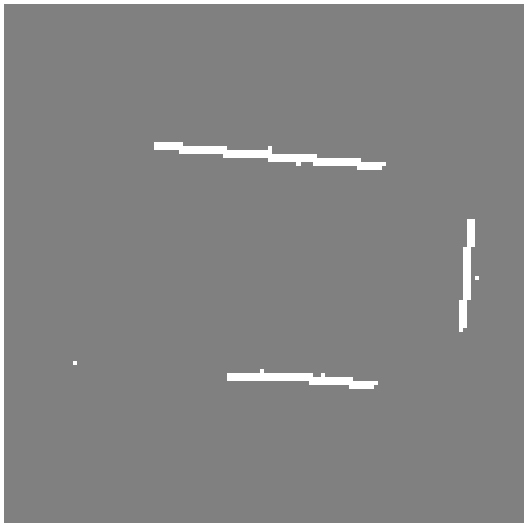
The robot's field of view is added by creating a line between the center of the view map and each point on the circle with a radius of the maximum sensing range. The line is traversed until it finds an occupied cell or the end of the line, and then labels all cells between the center and the current point as open. The finalized view map as shown in Figure 4.1(d), which will be used for the merging operations.

4.2.4 Map-Merging

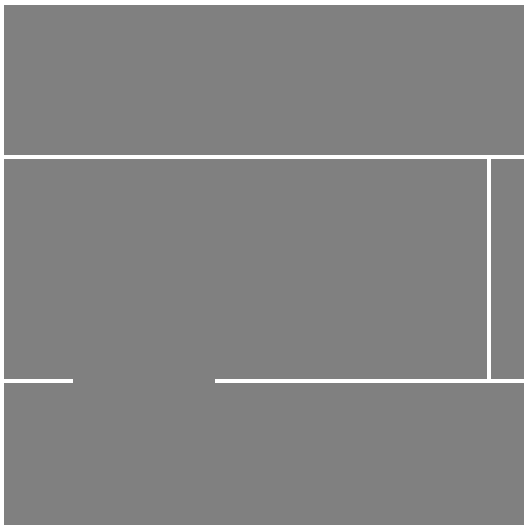
Once the view maps are generated by each robot then the maps can be merged during a communication phase. Each robot's view maps are only shared if communication occurs during this step. If communication does not occur this step, then each robot merges just their own view map with their *local map*. The local map is a map that each robot keeps of the entire field which is usually at a lower resolution than the view map.



(a) Example view map from direct distance sensor readings



(b) Example view map with extracted and merged line segments overlaid on top of the direct distance sensor readings.



(c) Matched known wall lines

(d) Known wall lines with field of view

Figure 4.1: View maps at various stages of processing. The robot is located at the center of the view map.

In the simplest case of no communication, the view map is merged with the local map using the same probabilistic technique found in SARA-1:

$$^{LM}\text{ViewMap}_{x,y} = \mathbf{H}\left(^{VM}\text{ViewMap}_{i,j}\right) \tag{4.2}$$

$$top_{x,y} = \left(\frac{LocalMap_{x,y}}{1 - LocalMap_{x,y}}\right) * \left(\frac{^{LM}\text{ViewMap}_{x,y}}{1 - ^{LM}\text{ViewMap}_{x,y}}\right) \tag{4.3}$$

$$CombinedMap_{x,y} = \frac{top_{x,y}}{1 + top_{x,y}} \tag{4.4}$$

In Equation 4.2, $\mathbf{H}$ represents a coordinate transformation from the view map coordinate system $^{VM}\text{ViewMap}_{i,j}$ to the local map coordinate system, $^{LM}\text{ViewMap}_{x,y}$, where the superscripts ^{LM} and ^{VM} denote the local map and view map coordinate systems, respectively. In Equation 4.3, the variable *top* is used to simplify the expression of Equation 4.4. The combination of Equations 4.3 and 4.4 result in a probabilistic merging that will tend toward "closed", a value of 0.9, if a closed cell is present. In the actual implementation, the maps are implemented as 8-bit integers for computational speed, thus the Equations 4.3 and 4.4 become:

$$top_{x,y} = \left(\frac{LocalMap_{x,y}}{256 - LocalMap_{x,y}}\right) * \left(\frac{^{LM}\text{ViewMap}_{x,y}}{256 - ^{LM}\text{ViewMap}_{x,y}}\right) \tag{4.5}$$

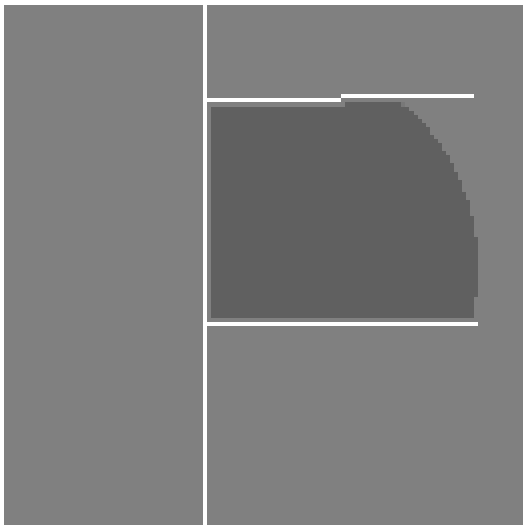
$$CombinedMap_{x,y} = \frac{top_{x,y}}{256 + top_{x,y}} \tag{4.6}$$

This operation is done for everywhere the view map is defined in the local map's coordinate system. After the calculations are complete, the combined map then becomes the local map for later use.

In the case where communication occurs, each robot first performs the merging of their own view map. Then they process the other view maps and pose estimates from the other robots using the same method as the single robot case. However, there

is "ring of exclusivity," which means that each robot will only trust its own sensors in an area immediately around itself. Thus, it will not merge any other robot's view map in the area that is extremely close to it. This prevents one robot from labeling itself as an obstacle because another robot has confused the robot for an obstacle.

(a) Robot 1's view map.



(b) Robot 2's view map.

(c) Robot 1's local map with exclusivity.

(d) Robot 2's local map with exclusivity.

Figure 4.2: View maps and merged local maps of two robots in a three robot test. In the view maps, the robot is in center of image. In the local maps, the robot is in the center of the "ring of exclusivity"

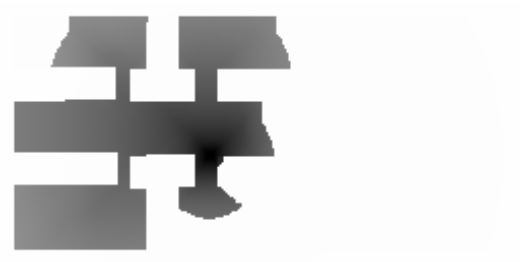
4.2.5 Path-Planning

The SARA-1 path-planning algorithm uses the idea of a *cost map*, which is a map that is the same size of the local map and has the cost associated with going to each pixel as the pixel's value.

One major limitation in SARA-1 that directly affects the destination point picking and path-planning behavior is the assumption that one robot is one pixel in the map. In this hardware testbed, the field's wall's thickness are much smaller than the robot so this assumption is not feasible because any large areas of open space next to a wall would be labelled closed. Therefore, another strategy had to be devised. If the obstacles are enlarged to the radius of the robot, then the same path-planning logic used in SARA-1 can be used in SARA-2 with only a minimal preprocessing of the local map. However, if any area on the cost map is within one robot radius away of the robot then it will not be enlarged. This is so a robot can actually plan a path out and away from the obstacle. It is illustrated in Figure 4.3(b) where a dark circle cuts into the rectangular white section.

Once the cost map is generated, the same method is followed as in SARA-1 to identify destination points using frontier cells, which are separated by a minimum distance and the cost to reach each of these frontier cells is calculated. The entire frontier cell list is then sorted by cost. The top of the list is the new destination point, which is the frontier cell with the lowest cost.

In the case of communication, each robot's list of cells and associated costs is shared between all the other robots and the entire list is then sorted. The robot with the lowest cost wins the first destination point and any points within a certain distance of that destination point are removed from the collective lists.



(a) Robot 1's cost map.

(b) Robot 2's cost map.

(c) Robot 1's local map with exclusivity.

(d) Robot 2's local map with exclusivity.

Figure 4.3: Cost maps and corresponding local maps of two robots in a three robot test. Robot position can be found in the darkest areas of the cost map, which correspond to the least cost.

Finally, the path can actually be planned using a similar approach as SARA-1 and the modified cost map. This is accomplished by following a steepest descent down the cost map from the destination point. If the path falls to a local minima for a few iterations of the path building process, then it arbitrarily doubles the current cost to push itself out of local minima until it arrives at the robot's position.

4.2.6 Path-Following

Once the path is planned, the robot should immediately execute it via the path-following behavior. This is accomplished by setting a maximum angle threshold. For each odometry sensor input, the offset angle between the front of the robot and the current path point is calculated. If this angle is greater the maximum angle threshold,

the robot will turn until the offset angle is less than the threshold. Additionally, another threshold called the *minimum distance threshold* controls the removal of path points. If the distance between the center of the robot and the current path point is less than the minimum distance threshold, then the robot is considered to have arrived at the destination path point and it is removed from the path. The robot travels towards the next point in the path list. This process is repeated until the path is exhausted and the robot arrives at the destination point or until the robot hits an obstacle.

4.2.7 Obstacle-Avoidance

Upon collision with an obstacle, the robot ends the movement phase of this step and restarts the map-building behavior.

4.2.8 Communication

Communication between robots is accomplished via an IEEE 802.11b mobile ad-hoc network, MANET . Each robot has an assigned individual IPv4 address and joins a UDP multicast group. Communication is accomplished by sending packets to this multicast group. This is effective due to the inherent broadcast communications that SARA-2 requires; all robots should be aware of what the other robots are doing to ensure their global maps are correct. In cases where all robots are not within range of each other, then node routing becomes an interesting issue and an area for further study.

Chapter 5

Experimental Setup

Due to the diverse nature of the hardware platforms used, three distinct code bases were developed to support them and the experiment. The three code-bases used for this project are the Robostix micro-controller code, the Gumstix ARM-based code, and the desktop base-station code. The ATmega128 micro-controller on the Robostix is programmed using C while the Gumstix and desktop code are both C++. This chapter is divided into four sections. The first section describes the concessions that had to be made to obtain a usable testbed. The remaining sections describe each code base that implements the SARA-2 algorithm.

5.1 Algorithm Concessions

In the actual implementation of SARA-2 onto low cost robots, there are two concessions that were made to allow a usable testbed to perform tests in a reasonable amount of time. In our implementation, the map-merging and path-planning behaviors take too long to perform on the robots themselves, easily taking over 5 minutes to allow the cost map to converge. Thus, the computations are offloaded onto the workstation and take approximately 2 seconds for all the robots. Additionally, the issue of small-scale localization was never successfully resolved. Fundamentally, the map based algorithm requires an accurate position estimate with which to merge the maps. The localization methods tried could not reduce the variance to acceptable levels with the sensors on the robots. Thus, during the tests, the robots were allowed

to plan their destination points and the associated paths, but were manually placed there by the test operator. The robot's positions were then updated to their proper position. This does not impact the fundamental question as the amount of communication and the number of robots on the team still varies and plays an important role in the success or failure of the mission.

5.2 Robostix Software

The Robostix code is responsible for sending the PWM signals to all three servos (IR scan turret, left drive wheel, right drive wheel) and acquiring the IR distance sensor and PS/2 mouse data. The software initially sets up the necessary variables and timing parameters and then enters an infinite loop, handling serial data as it comes in and checking status flags. Each servo is controlled via a PWM register in the ATmega128 micro-controller. Changing the value in the PWM register allows control over forward and reverse rotational speeds of the corresponding servo. Thus, full drive control over the robot is maintained by modifying the two PWM registers. The turret servo is controlled via a third PWM register. Since the servo needs to have settled to its new position before the IR sensor readings are taken, an additional timer is used for ensuring that the servo has enough time to move and settle to its new position. Once the turret servo has settled, the IR sensors are read via the ATmega128's A2D inputs. Each IR is polled four times with 32 milliseconds between readings. This data is then sent to the Gumstix where a conversion factor determined by calibration testing is applied to the raw data. The micro-controller polls the optical mice for movement data every 1/8 second via Jan Pieter de Ruiter's Atmel PS/2 library [17]. The returned three byte packets are parsed and sent to the Gumstix.

The Robostix disables the turret servo while the robot is moving. This serves to reduce movement errors caused by the torques exerted on the robot via the turret servo. Once the robot stops moving, the turret is activated again.

5.3 Gumstix Software

The Gumstix code is responsible for processing the raw mouse data, extracting robot position and keeping the robot following the planned paths. It is also responsible for sending the sensor data obtained from the Robostix back to the base station for more efficient processing. The Gumstix code is programmed in C/C++ using Glib for its signaling and IO monitoring system. Glib 1.2.10 was chosen due to its inclusion into the build system of the Gumstix. Using Glib's event loop and IO notification system allows a single threaded implementation to monitor both the wireless network IO from other robots and the serial port IO from the Robostix. The program is structured such that three channels are watched with corresponding functions designed to handle the respective channels. As mentioned before, the network connection is converted into a Glib channel, and a function is specified to handle parsing of the packets and to determine if any actions should be performed. The serial port is also monitored for communications with the Robostix with a function designed to parse the serial stream and execute any necessary functions for the robot based on that input. Lastly, *stdin* is monitored for user debugging of the code and the state of the robot.

The Gumstix code can also be thought of as a finite state machine with a few operating modes: Calibration, Setup, Search, and Target Found. The robot initially boots into the setup mode. In this mode, the servos are disabled but the sensors are initialized. For the calibration mode, the sensors are enabled, and the motors are

put under manual user control. The search mode is the active mode, with the robot actively scanning its sensors, transmitting data back to the base station, and driving any paths sent from the base station. Once a robot finds the target, it is put into Target Found mode, whereupon it shifts into a state similar to the setup mode. At this point, the robot has effectively finished its task, but still needs to communicate with its neighbors to help them find the target. When a robot reaches the Target Found mode, it then tells the other robots the position of the target.

5.4 Desktop Software

The desktop code is used for data logging and for computation of the destination point and path. The code is written using C++ and based on the gtkmm C++ wrappers for the GTK+ library. It is designed such that each robot is kept isolated in its own class instance, referred to as the *robotClass*, with any communication originally occurring between robots now taking place between the *robotClass* instances. In this way, the communication aspect is preserved and can be migrated back to the robots when more processing power is available. Additionally, this allows a variable number of robots to be supported: it is only necessary to instantiate a new class instance per additional robot.

The key then becomes knowing what data coming in over the network corresponds to which robot. The robots communicate over an ad-hoc network via an XML-based data format. This format was chosen since it is desirable to have all data logged to a MySQL database, and for the to data be in a human-readable format. In keeping with this idea, each packet is required to have a minimum of two tags. The source tag which serves as the root element is either <r> for transmission from robot or <b> for transmission from the base station. This is followed by the robot number

tag <n>#</n>. If the source tag is <r>, then the robot number is the number of the robot sending the packet. If the source tag is <b>, then it is the robot to which the packet is to be sent. If the packet is from the base station and has a number of zero, then all robots will listen to the data. The element names were chosen to be one letter long in an effort to reduce the amount of data transferred while still providing human readability. A full listing of used element tags is presented in Tables 5.1 and 5.2. Since the original implementation had all processing take place on the robots, a sensor data element was not added to the data format. However, this information is sent back within the debugging tag and is correctly parsed by the desktop program.

Using the source number from the robot packets, the desktop code takes the sensor readings from each robot and distributes the data appropriately to the specific robot class instance associated with that robot. The *robotClass* is responsible for converting the raw IR sensor values to an approximate distance through a calibrated curve fit function for use in the map-building behavior. The error-correction, map-merging, and path-planning behaviors of the SARA-2 algorithm are performed as described in Chapter 4.

A robot is considered to have found the target when it is within line-of-sight at a pre-defined range.

<r>	Robot header **(required)**
<n> # </n>	Source Robot's Number **(required)** # = integer
<L> x y theta </L>	Location (optional).
<D>x y </D>	Destination (optional).
<T>0</T> <T>1 x y</T>	Target information (optional). If 0, target not found. If 1, target found. x,y location.
<d x=# y=# w=# h=#>data</d>	Map data (optional) x = upper left x y = upper left y w = width h = height data = Base 64 encoded binary data.
<g> string </g>	debugging values (Optional). (ignored by other robots)
<M/>	Generate Destination and Path (ignored by other robots) Tells the base-station to generate a destination and path for the robot.
<P/>	Path received acknowledgement (ignored by other robots). Tells the base-station that the robot received the previously sent path.
</r>	End of the robot packet **(required)**.

Table 5.1: Robot to Robot/Base station packet format.

	Base station header **(required)**.
<n> # </n>	Destination Robot's Number **(required)**. # = integer.
<L> x y theta </L>	Location (optional).
<T>0</T> <T>1 x y</T>	Target information (optional). If 0, target not found. If 1, target found. x,y location.
<s> code </s>	State to put the robot in. (Optional). Code: 0 Setup 1 Start Search 2 Homing
<P n=#> x0 y0 x1 y1 ... </P>	Path to Destination (Optional). n = the # of points in the path. x0/y0 = the first point, followed by subsequent points.
	End of the robot packet **(required)**.

Table 5.2: Base Station to Robot packet format.

Chapter 6

Results

A series of tests were performed following the SARA-2 algorithm as described in Chapter 4 and implemented as in Chapter 5. The tests can be divided into single robot tests, three robot tests, and five robot tests. The multiple robot tests are divided into three sections: no communication, occasional communication, and full communication. Communication in this context means that the view maps and the destination points of each robot are shared during that step with the other robots. Occasional communication means communication occurs on the first step and every third step afterwards. Full communication means that communication occurs at every step.

Up to three tests were performed for each run. The results are listed in Table 6.1 and plotted in Figure 6.1.

	Amount of Communication		
Number of Robots	None	Occasional	Full
1	19, 20 (19.5)	NA	NA
3	∞* (∞)	18, 15 (16.5)	9, 9 (9)
5	23, 18 (20.5)	16 (16)	7, 7, 8 (7.33)
*Inter-robot interference resulted in no robot finding the target.			

Table 6.1: Number of Steps to find the Target. Average values are in parentheses.

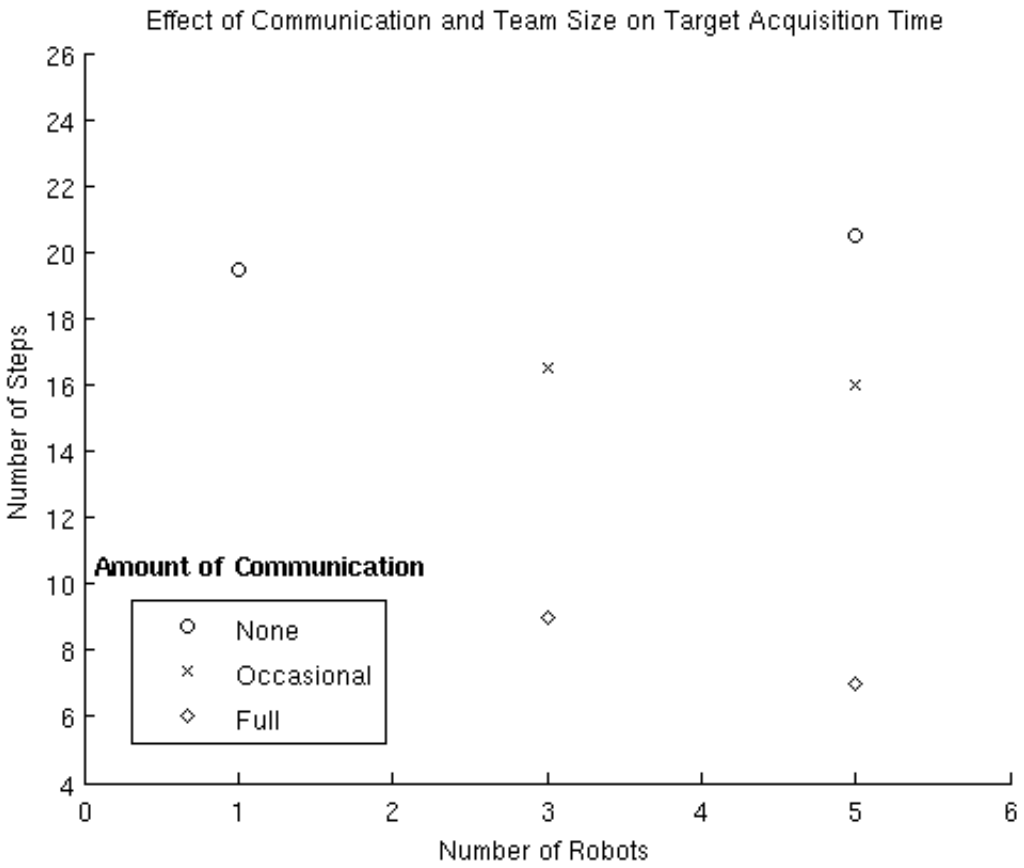


Figure 6.1: Plot showing the effect of team size and amount of communication on average target acquisition time.

6.1 Single Robot Results

The single robot tests serve as the baseline from which to compare the effect of cooperation and communication. The single robot case took an average of 19.5 steps.

6.2 Multiple Robot Results

6.2.1 Three Robots

For the three robot case, communication results in lower number of steps per run. In the no communication test, the robots never found the target as they got

stuck blocking each other. In the occasional communication test, the robots found the target in an average of 16.5 steps. In the full communication test, the robots were able to find the target in 9 steps every time.

For the case of no communication between robots, the robots actually interfered with each other so much as to deadlock each other in a doorway, resulting in a completely unsuccessful run. Further analysis reveals that many times the robots would not see each other as the turrets turned to present their slimmest profile as the other robot would scan that area. Additionally, the times that it did pick up the other robots, the algorithm itself failed to adequately place the obstacles due to the matching of the known lines only, which was done for localization purposes.

In the occasional communication runs, the robots would often interfere with each other, especially in the beginning, to continuously block each other from moving on that step as in the no communication test. However, a communication step would resolve the conflict and allow the robots to plan appropriate destinations and move.

The full communication test runs resulted in a much lower number of iterations to find the target, only 9 steps per run.

6.2.2 Five Robots

In the five robot test, much like the three robot case, increased communication results in fewer steps per run. In the no communication test, the robots took an average of 20.5 steps. In the occasional communication test, the robots found the target in an average of 16 steps. In the full communication test, the robots were able to find the target in an average of 7.5 steps.

With no communication, most of the robots deadlocked as in the three robot case, however, one robot was successfully able to escape the traffic and find the target.

In the occasional communication runs, the robots duplicated much of the search effort and ended up deadlocking themselves until a communication step, in which they were able to resolve their destinations.

In the full communications test runs, the increased number of robots ended up not spreading out efficiently enough to explore the full area, with multiple robots consistently grouping together. These would pick destination points that were the minimum distance away. However, the increased number did help in that one robot would go into the target's room faster than in the three robot case.

Chapter 7

Conclusions

This chapter will consist of two sections. The first section will summarize the results. The second section will list possible modifications for future work.

7.1 Summary

In the implementation of SARA-2, increased amount of communication serves to dramatically decrease the number of steps required to find the target. If multiple robots are present and no communication is available, then the robots will more often get in each other's way and reduce the success rate of the mission.

With occasional communication, the search effort ends up getting duplicated often. In the steps without communication, the robots often ended up blocking each other until a communication step occurred, which greatly increased the number of steps required to find the target.

The full communication case consistently resulted in the lowest number of steps to find the target. This is intuitive as the robots spread out at least as much as the minimum distance threshold specified in Section 4.2.5.

7.2 Future Work

One of the main issues in getting SARA-2 to work well on actual hardware is the error in the localization procedure. Small scale localization is currently not feasible with inexpensive sensors, but improvements in this area should be studied and used

to improve the algorithm. Extending this work to outdoor scenarios using GPS is also suggested.

Additionally, a look at multi-hop routing protocols for the cases where robots are out of range should be evaluated. This can be accomplished by varying the power levels on the Wifistix card in the Linux Gumstix interface.

Bibliography

[1] U. D. of Commerce. PDF. [Online]. Available: http://www.weather.gov/om/assessments/pdfs/Katrina.pdf.Retrievedon2006-07-14.

[2] P. A. Sheikh. (2005, October) The impact of hurricane katrina on biological resources. PDF. Congressional Research Service. [Online]. Available: http://www.opencrs.com/rpts/RL33117_20051018.pdf

[3] V. Murphy, "Fixing new orleans' thin grey line," *BBC News*, October 4, 2005.

[4] U. S. C. Bureau. (2004, April) United states summary: 2000. PDF. [Online]. Available: http://www.census.gov/prod/cen2000/phc3-us-pt1.pdf

[5] A. Ray, "Cooperative Robotics using Wireless Communication," Master's thesis, Auburn University, 2005.

[6] R. Sim and G. Dudek, "Mobile robot localization from learned landmarks," *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 2, pp. 1060–1065 vol.2, Oct 1998.

[7] S. Se, D. Lowe, and J. Little, "Vision-based mobile robot localization and mapping using scale-invariant features," *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2, pp. 2051–2058 vol.2, 2001.

[8] A. Elfes, "Sonar-based real-world mapping and navigation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 3, pp. 249–265, June 1987.

[9] K.-T. Song, C.-Y. Tsai, and C.-H. C. Huang, "Multi-robot cooperative sensing and localization," in *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, Sept. 2008, pp. 431–436.

[10] J. Chen and L.-R. Li, "Path planning protocol for collaborative multi-robot systems," in *Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on*, June 2005, pp. 721–726.

[11] S. Park, J. H. Jung, and S.-L. Kim, "Cooperative path-finding of multi-robots with wireless multihop communications," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*, April 2008, pp. 673–678.

[12] I. Parallax. Ping))) ultrasonic distance sensor (#28015). [Online]. Available: http://www.parallax.com/Portals/0/Downloads/docs/prod/acc/28015-PING-v1.5.pdf

[13] A. Bonarini, M. Matteucci, and M. Restelli, "A kinematic-independent dead-reckoning sensor for indoor mobile robotics," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, pp. 3750–3755 vol.4, Sept.-2 Oct. 2004.

[14] Gumstix, "Gumstix Wiki - Wifi," http://docwiki.gumstix.org/Frequently_asked_questions/Wifi. [Online]. Available: http://docwiki.gumstix.org/Frequently_asked_questions/Wifi

[15] C. Wilson and T. Roppel, "Low-cost wireless mobile ad-hoc network robotic testbed," in *Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*, April 2009, pp. 1–6.

[16] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. MIT Press, April 2004, iSBN-10: 0-262-19502-X.

[17] J. P. de Ruiter, "PS/2 library," http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/ps2/ps2.html. [Online]. Available: http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/ps2/ps2.html