A Simplistic Approach to Reactive Multi-Robot Navigation in Unknown Environments

William MacKunis, Dr. Daniel Raviv
Department of Electrical Engineering
Florida Atlantic University, Boca Raton, FL 33431
E-mail: ElectronWave@aol.com
954.421.7597

Abstract

Multi-agent control is a very promising area of robotics. In applications for which it is difficult or impossible for humans to intervene, the utilization of multi-agent, autonomous robot groups is indispensable. In this paper, a novel approach to multi-agent control is analyzed in which concepts from several different engineering disciplines are unified within a simplistic yet robust framework. While many recently researched strategies for multi-agent control require cumbersome calculations and complex sensor layouts, the approach presented here enables a multi-agent group to perform very accurate navigational tasks using minimal sensory equipment and simple geometrical calculations. This approach incorporates various concepts from geometry, trigonometry, sensor fusion, software/hardware integration, and RF communication in order to explore a method that can be used to simplify the overall process of coordinating multi-robot navigation in unknown (unmapped) environments. Further, the purpose of this paper is to present a possible new direction for engineering education in the area of multi-agent robotics.

Simplicity is one of the most important aspects of a multi-agent control scheme. Many recently researched strategies to multi-agent control tend to increase the complexity of their respective systems for the purpose of attaining superior results. The research presented here utilizes the concept of Occam's Razor or the principle of parsimony, which states: "It is vain to do with more what can be done with less." This concept is extremely profound from an engineering standpoint, and engineering education would be greatly enhanced by emphasizing this axiom. In this paper, a multi-agent robot group is shown to demonstrate extremely efficient navigational patterns based upon a control scheme that incorporates the concept of Occam's Razor. The fundamental idea upon which this approach is based is that a multi-robot team can cooperate to determine the shortest path through an unknown environment given only a very simple set of rules. In addition, the approach being presented in this paper can be successfully implemented using very simple sensors. The idea was implemented with two robots. In simulation, groups consisting of over sixty agents were tested, and it was shown that a near optimal (shortest) path emerges through the environment without mapping the environment. The experimental results clearly show that the robots' navigational ability through various unknown environments improves with time under the control of this algorithm. Furthermore, the simplicity of this approach makes implementation very practical and easily expandable to reliably control a group

comprised of many agents.

1. **Introduction**

While current trends in robotics toward reactive, multi-agent control have demonstrated systems that function efficiently and robustly in various environments [24], there still exists a need for a simple reactive method for multiple-agent control that maximizes efficiency, robustness and scalability while minimizing the need for complicated calculations and sophisticated sensory equipment [34]. This paper presents a simplistic, reactive approach to cooperative multi-robot navigation that is easily expandable and remarkably practical in its simplicity.

In order to test the concept presented in this paper, a simple navigational task was devised. For this task, various simple maze-like regions were constructed as the robot group's experimental environments. The robot group began from a designated starting point, called the "nest," and a single, "scout," robot was deployed from the nest to graze the environment in a random fashion in search of a goal location. While grazing, the scout robot was constantly updating information concerning its relative location (direction and displacement) traveled from the nest. Once the goal location was found, the scout robot sent a message back to the other robots, the "workers," containing the direction and displacement of the shortest (straight line) path to the goal from the nest. Once the data was received by the workers, they actively sought the goal location. If the workers encountered obstacles as they sought the goal location, they avoided them and recalibrated their desired goal trajectory to account for the unexpected course alteration. The experiment tested the robot group's ability to cooperate to navigate an unknown environment using very simple calculations and minimal sensory equipment. In addition, it demonstrated a technique for multi-agent robot navigation in unknown environments that minimizes the risk to the robot group as a whole by deploying only a single robot initially to investigate the environment. Further, the minimalism of this approach enables practical implementation and exhibits safeguarding techniques that reduce the risk for robot agents working in unknown and possibly hazardous environments.

1.1 Problem

In evaluating the effectiveness of multi-agent control schemes, three of the most important aspects to be considered are practicality, robustness, and scalability. If the scheme can be inexpensively implemented, it is practical; if the group can quickly recover from individual agents' failures, it is robust; if the scheme allows for easy expansion to include many agents while retaining or improving its overall efficiency, it is scalable. The purpose of the scheme being described here is to investigate a novel scheme for multi-agent control that demonstrates the aforementioned virtues. The basic concept of this scheme concerns simple communication between minimal agents in order to perform a cooperative navigational task in an unknown environment. This approach investigates the problem of developing a simplified reactive navigational scheme that can be used to effectively and reliably control a multi-agent robot group.

1.2 Importance of Effective Multi-agent Systems

The implementation of multi-agent cooperative robot groups consisting of locally reacting robot members is a very promising area of robotics research. The idea that 'many hands make light work' has been reinforced in nature many times primarily by the success of insect colonies. Examples of situations in which multi-agent robot groups would be useful are collection of scientific data from uninhabitable environments (i.e., the Moon or Mars), location of mines in a mine field, and collection and disposal of toxic materials. In order for multi-agent robot control to be practical, however, the robot constituents must be minimal in terms of their sensing and equipment. The concept presented in this paper exhibits quite robust performance and computationally inexpensive, practical scalability.

1.3 Related Work

Biological systems are often used as models for multiple robot controllers, and many successful implementations have been devised in an attempt to model nature. For example, in [24] systems were presented that dynamically encode information into the physical environment. This idea was inspired by the pheromone trails of ants. The scheme was based on "minimal" agents with local sensing and action, which formed a system that can perform position-dependent tasks without explicit communication. The importance of inter-agent communication was explored by Balch and Arkin in [3], where it was determined that it can significantly improve performance in some cases, but is apparently unnecessary in others. It was further concluded in [3] that in cases for which communication between agents helped, the simplest level of communication was almost as effective as the more complex type. In addition, communication of only a single bit of information was shown to provide an advantage in the multi-robot group foraging task outlined therein. An alternative communication scheme was presented in [13], in which two robots developed shared groundings that allowed them to form a symbolic relationship between positions consistently. It was shown in this paper that this concept enabled symbolic communication of locations between them. This idea was illustrated by means of a cooperative robot cleaning task. In [34], an approach to multi-robot control that is based upon role assumption is presented, in which each agent selected its own task-performing role. This approach used the physical bodies of the robots to replace the chemical pheromones of ants. A similar technique based upon formation-keeping was presented in [2], where the robot's functional purpose was dependent upon its position within a certain formation. Many RoboCup soccer systems utilize formation-dependent control schemes. In [19] behavior-based techniques were utilized to control a RoboCup soccer team without explicit communication between the team members. A robot soccer team control scheme was illustrated in [12], which used Hidden Markov Models to recognize and represent strategic behaviors of robotic agents. In [5] another robot soccer system was employed that utilized inter-robot communication. This work proposed that the use of communication among the players may improve the team's performance, but that an increase in the amount of transmitted data does not imply better results. While many recently developed systems incorporate behavior-based algorithms [2,19,22,24,26,28,31,33,34,35], systems involving path-planning are also popular [1,6,7,8,10,14,15,16,21,30,32]. The system in [14] dealt with finding the time-optimal path of a robot in the presence of moving obstacles. This system

considered the time axis as the z-axis of a 3-D configuration space, thereby allowing a planar moving obstacle to be represented as a 3-D structure in the configuration space. Alternatively, the scheme in [8] utilized
a landmark-based algorithm by which two robots with different sensory capabilities and mechanics each independently explored their environment with no a priori knowledge of the
area. The two maps were then merged into a topologically correct single map. Another map-making strategy was implemented in [10] using an occupancy grid. The scheme in [10] required the agents to have sufficient memory to store the map and still be able to communicate with each other. Similarly, a scheme was presented in [32] by which a 2-D grid of an environment was mapped and then reduced to 1-D polar histograms that were constructed around the robot's momentary location. Then, the most suitable direction was selected by the algorithm based on the masked polar histogram and a cost function. Techniques utilizing internal models of the environment were demonstrated in [15,25]. In [15], a humanoid robot used an internal model of the environment in conjunction with a stochastic algorithm to plan a collision-free path from a starting point to a finishing point. The system in [25] used a Graphical User Interface (GUI) to allow a single user to control and coordinate multiple robots and deploy the group into a previously mapped area.

2. The New Approach

The approach presented here is substantially more simplified and practical, yet exhibits similar performance characteristics as those of systems that use much more elaborate sensor setups and cumbersome calculations. This concept utilizes the idea that mapping is unnecessary for a robot group to successfully navigate an unknown environment. The robot (scout) responsible for the discovery of a goal location only shares with the workers information about the general direction and displacement to the goal location, and the others (workers) act globally on this information while reacting locally to imminent obstacles in order to seek the location. The simplicity of the concept presented here permits implementation of a multi-agent group using inexpensive hardware and relatively simple software. Moreover, the concept described here offers many advantages due to its minimal nature and practical usability.

2.1 Fundamental Concept

While currently researched strategies to multi-agent robot control perform successfully in their respective tasks, there still exists a need for a control scheme that is not only scalable, but simplistic enough to allow practical, efficient implementation of a group comprised of many agents. The approach presented in this paper is based on a homogeneous group of mobile robots in which the agents navigate an unknown region acting under a very simple set of rules. This reactive scheme is layered in a two-tier fashion: **1) Avoid obstacles. 2) Keep track of the goal's relative location.** The flowchart given in Figure 2.1 provides a more detailed explanation of this algorithm: the scout robot is deployed into some environment and constantly "watches" for obstacles while simultaneously "watching" for the goal location and updating its general direction *vector* (direction and displacement) traveled from the nest. If the scout encounters obstacles during its journey, they are avoided, and the scout continues searching the environment
in a random fashion. If the goal is found, the scout sends a message to the nest workers

indicating the general direction vector from the nest to the goal. Once this message is sent by the scout, the scout's program loop is concluded. The scout robot then repeats the process by

continuing to search the environment for additional goal locations. Figure 2.2 shows the decision process for the nest worker robots: once the message containing the goal location's general direction vector is received by the nest worker, it first rotates in place until its heading matches the direction to the goal as indicated in the scout's message. The nest worker then proceeds to move forward. If the worker encounters obstacles during its journey, they are avoided, and the heading is then readjusted in order to account for the unexpected detour. If the goal is spotted, the worker sends a message to the rest of the workers waiting in the nest. The message sent by the worker contains any additional information gathered by it during its journey to the goal (i.e., location of obstacles, etc.). Once the worker sends its message, the worker's program loop is complete. The worker would then continue to the next goal location if a message containing another goal location is received. Otherwise, the worker would return to the nest to wait for its next turn to seek a goal location. Layering the algorithm as indicated in the flowcharts allows a robot team to navigate through an unknown region without making a map of the region. This is a very beneficial virtue of the system because it allows the agents to make better use of their

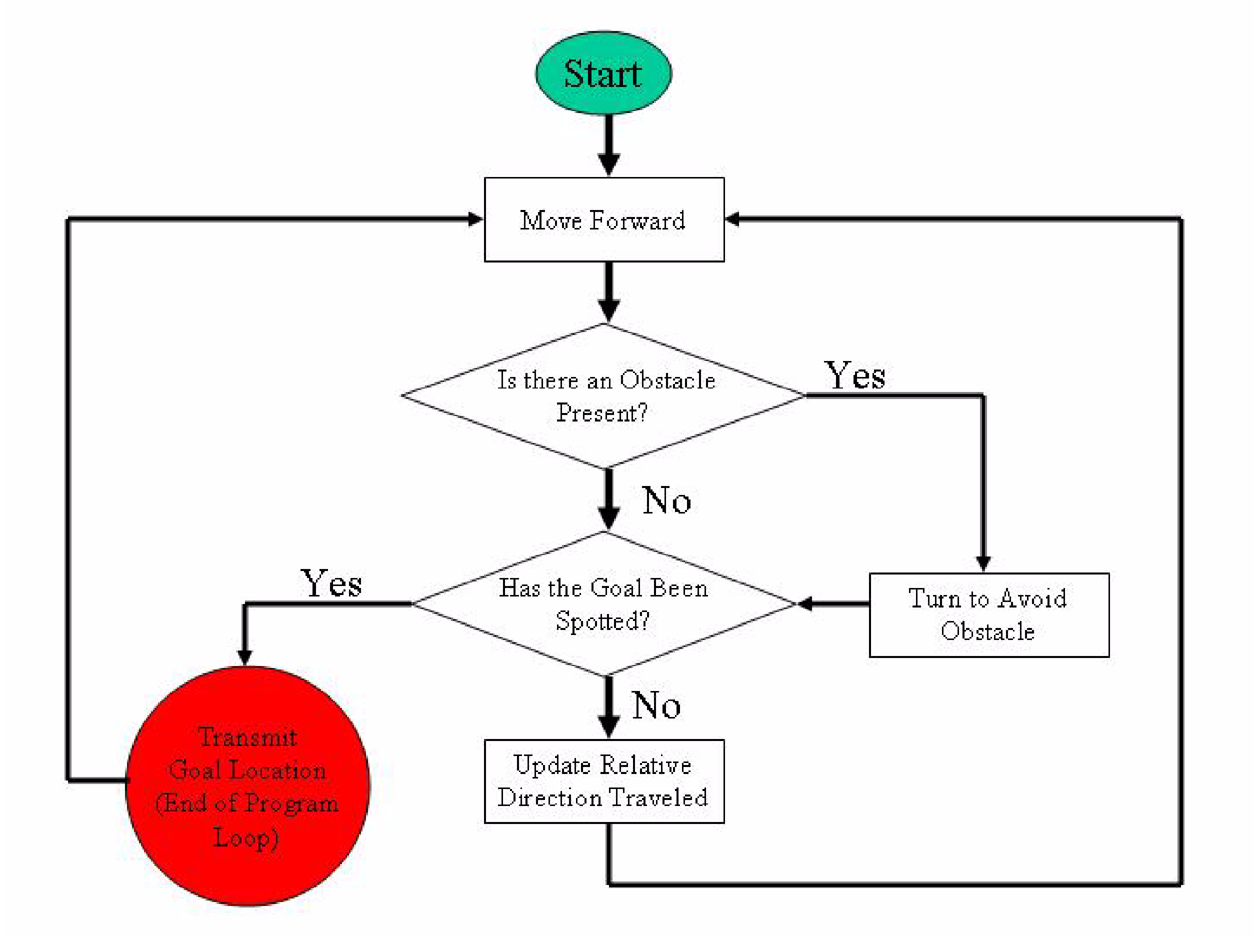

Start
Move Forward
Is there an Obstacle Present?
Yes
No
Has the Goal Been Spotted?
Yes
Turn to Avoid Obstacle
No
Transmit Goal Location (End of Program Loop)
Update Relative Direction Traveled

Figure 2.1: Scout robot's program routine

data storage resources by using very little data for navigation. Furthermore, the minimalism of this scheme permits the use of simple sensory resources. This makes for a control scheme that is practical and easily scalable.

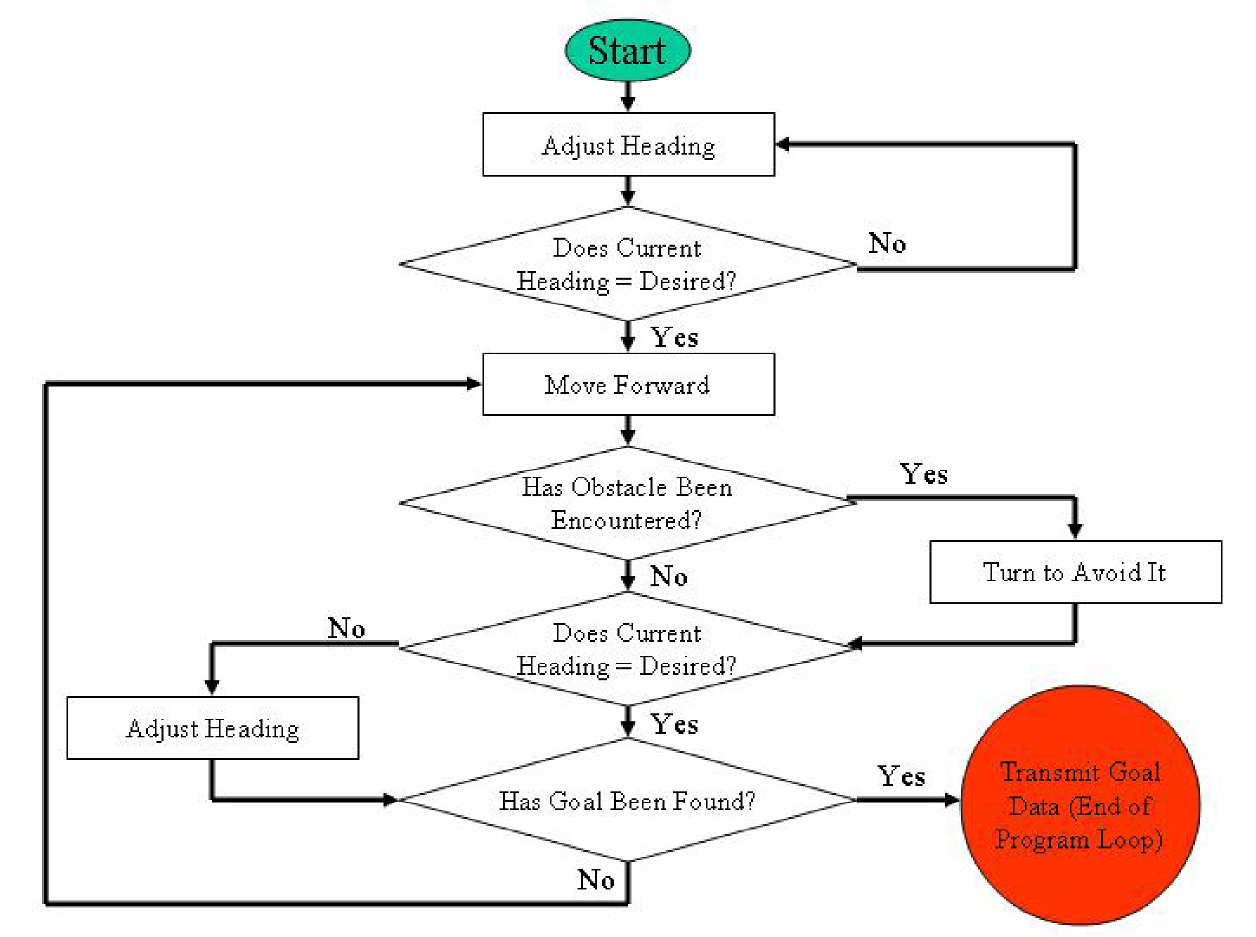

Start
Adjust Heading
Does Current Heading = Desired?
No
Yes
Move Forward
Has Obstacle Been Encountered?
Yes
Turn to Avoid It
No
Does Current Heading = Desired?
No
Adjust Heading
Yes
Has Goal Been Found?
Yes
Transmit Goal Data (End of Program Loop)
No

Figure 2.2: Flowchart showing the worker robot's goal seeking program routine

The idea of this control scheme is that all of the robots begin from a common starting point, the 'nest'. Initially, a 'scout' robot is deployed into the unknown environment and grazes within the environment while constantly retaining a sense of its general direction and displacement traveled from the nest. If the scout discovers a location of interest, it sends a message to the rest of the agents in the group. The message gives enough information to the other agents to allow them to actively seek the location(s) of interest. Once the scout has delivered the message, it continues to graze the region looking for more locations of interest. Since all of the agents begin from the same starting point, the nest, they are each able to share meaningful goal location information with the others. More importantly, the goal location information is improved each time the next robot in succession discovers the goal location. This is due to the fact that every robot sends a message to the rest of the group when it discovers the goal location, thus any improvements to the original goal location data are passed on. The redundancy of this process results in the data being improved with each iteration.

This concept was inspired in part by the cooperative foraging method of ants. In an ant colony only one ant is deployed at a time to search for a food source. When a food source is discovered, the scout ant returns to the nest while leaving a trail of pheromones. The worker ants in the nest can then follow the scout's trail of pheromones back to the food source while each leaves their own pheromone trail. In the case of ants, a shortest path to the food source quickly emerges due to the fact that chance variations in each ant's path to and from the food source which lead to a faster route are reinforced at a slightly faster rate [13]. This happens because the more frequently followed pheromone trail will have the stronger pheromone concentration. Due to its highly evolved survival instinct, the ant colony does not risk the lives of many ants in an attempt to search for food, but instead only deploys a large number of workers in the event of a food source discovery. The control concept being described here applies a similar idea to a multi-agent robot group. Since this concept may be used for applications in unknown and possibly hazardous environments, robots are deployed one at a time to search the unknown environment. The inter-agent communication provides a means for the scout robot to call for the worker robots only when an item of interest is discovered. The virtual elimination of the robots' aimless wandering time resulting from this concept minimizes the risk to the robot group as a whole. Once the message of a discovery is received by the robot nest workers, they begin their journey to the specified location with knowledge of the absolute shortest straight-line path to it. Each time a worker discovers the goal location, the next worker in succession can only benefit from the data gathered by its predecessor. Unlike the worker ants, for whom a shortest path to the food emerges as a result of *local* reaction to the scent of a pheromone trail, the worker robots in this approach seek out an optimal path to the goal by acting *globally* on information given to them by the scout and their predecessor worker robots while reacting *locally* to proximate obstacles.

Figure 2.3 (below) shows a qualitative description of the robot agents implementing the multi-agent control scheme outlined herein.

2.2 The Algorithm

The basic idea upon which this algorithm is based is that only information (a vector) about the *general direction* of a target location and the *displacement* from a given starting point are required by a robot in order for it to actively seek a goal location. This statement is quite obvious for a robot seeking a target location in a region containing no obstacles, but it can also be applied to a random environment that contains various obstacles. The reason for this lies in dual layer nature of the algorithm. The first layer of the program causes the robot to turn to avoid obstacles if encountered. The second layer monitors the incoming compass data and adjusts the robot's heading to match the desired target location heading. If an obstacle is spotted while the robot is in transit to a desired location, the robot rotates in place by some known increment until the obstacle is no longer in its path. At that point, the robot begins to proceed forward until it detects that its heading is not within the prescribed error margin of the desired heading. The heading is then adjusted until the compass reading is within an acceptable range. Refer to Figure 2.4 for a graphical interpretation of the robot's general direction calculation routine. Each of the solid black arrows in Figure 2.4 represents a vector measured by the robot as it moves forward.

The relative length of the vector is exaggerated in Figure 2.4 for the sake of clarity. The vectors are measured by the robot at regular time intervals, and the corresponding x and y components are summed at each interval such that Equation 1 calculates the general direction traveled by the robot from the nest at any instant in time. The thick green line indicates the robot's calculated

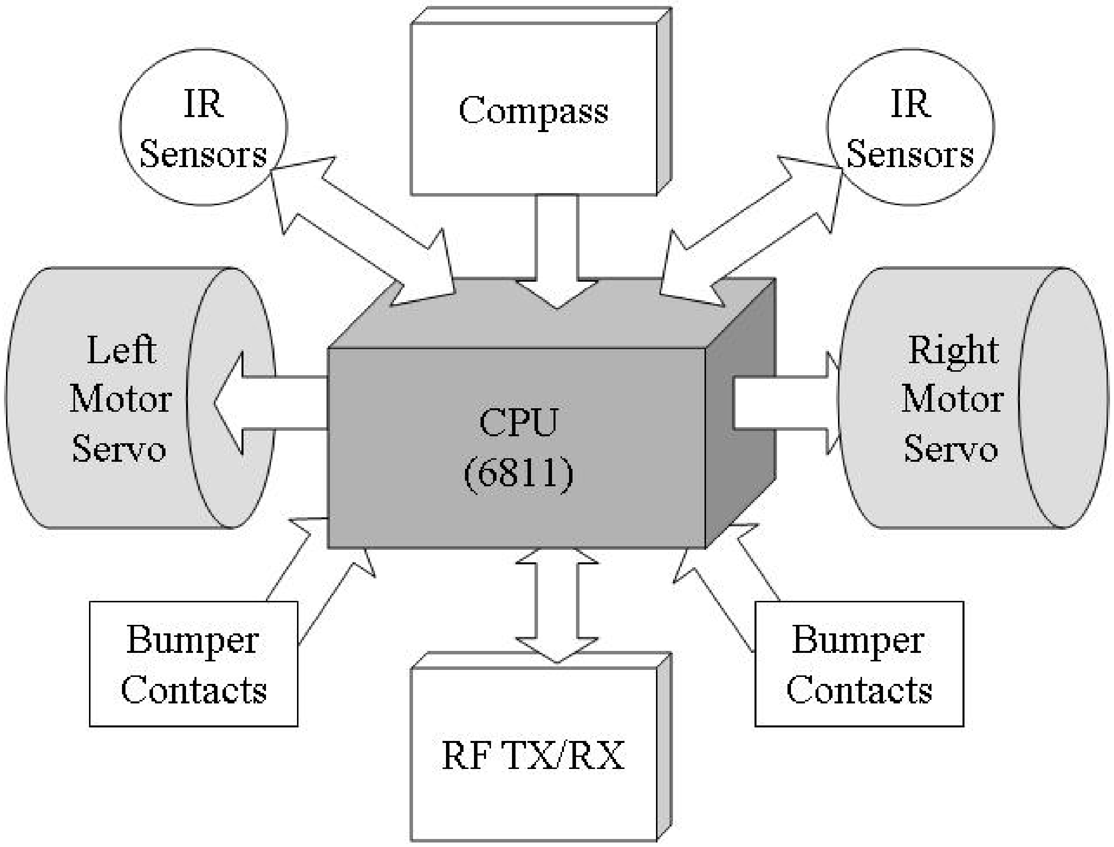


Figure 2.3: Diagram indicating the devices interfacing the CPU for a robot agent utilizing the control scheme being described here.

general direction traveled using Equation 1. The $v_i(t)$ term in Equation 1 represents the instantaneous velocity of the robot, and the Δt term represents the heading sampling time.

$$qg = \mathbf{arctan}\{(Sv_i(t)*Dt*\sin[qi]) / (Sv_i(t)*Dt*\cos[qi])\} \qquad \mathbf{(1)}$$

Equation 1 can be used to calculate the general direction traveled at any instant in time by a robot traveling at various velocities and sampling its heading at various rates. A simplification of Equation 1 will be given in chapter 3.

The robots' ability to retain its sense of direction (as indicated in Figure 2.4) involves the recording of directional data at regular time intervals in order to allow each robot in the group to find its way back to the nest or any previously marked location of interest at any given time. The directional information gathered by the robot provides it with the direction of the shortest straight line path to such a location of interest. The instantaneous heading is sampled at regular time intervals by the robot while the robot is moving forward. Since the robots comprise a

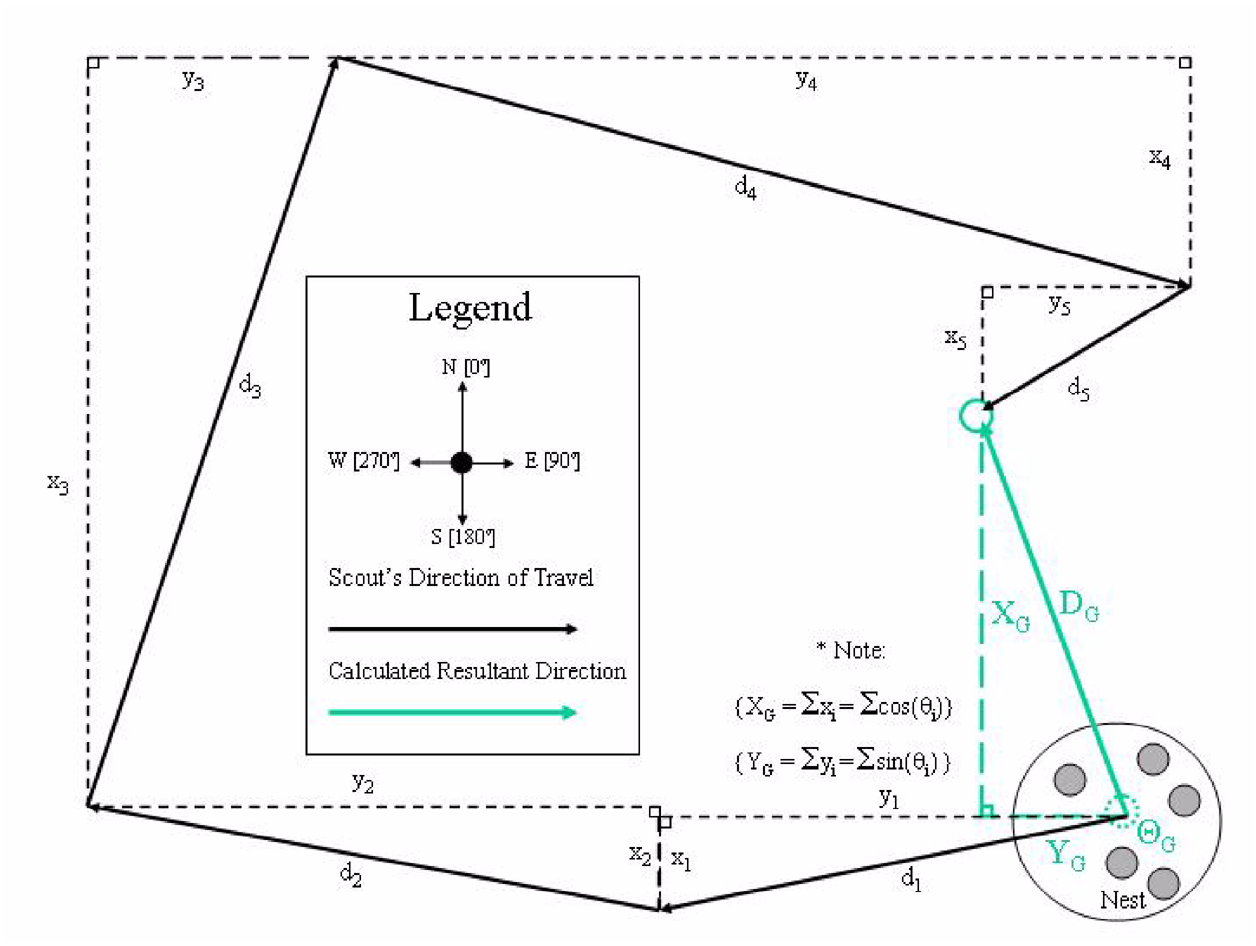
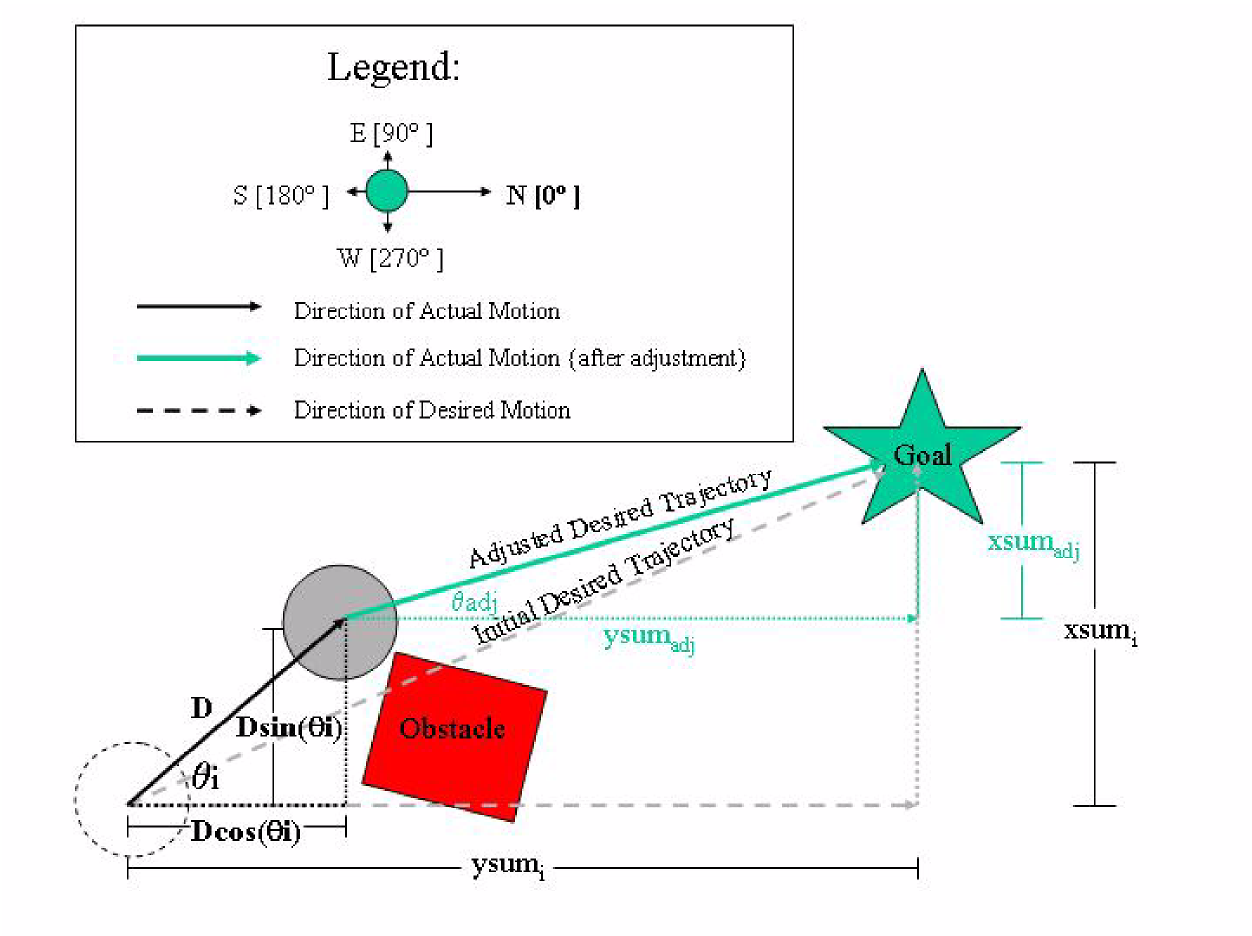


Figure 2.4: Illustration of the robot's method of recording its general direction and displacement traveled from the nest at any given instant during its journey.

homogeneous group, it is assumed that they all travel at the same speed. This allows each trajectory sampling interval to be considered mathematically as an incremental unit of distance. A simplified version of Equation 1 is utilized by the scout robot as it randomly searches an unknown region. In addition to recording the general direction traveled, the approximate displacement from the nest is also updated at regular time intervals. The calculation for displacement from the nest, **D**, is given by:

$$\mathbf{D} = \{ (Sv_i(t)*Dt*\sin[qi])^2 + (Sv_i(t)*Dt*\cos[qi])^2 \}^{1/2} \qquad \textbf{(2)}$$

The recording of the displacement data along with the general direction traveled from the nest enables the scout robot to send enough information to the workers to allow them to not only find the goal location, but also to keep track of the goal's relative location in the event of unexpected detours due to obstacles. When the scout robot discovers the first goal location, it transmits to the worker robots the direction and displacement from the nest to the goal. When the nest worker robots receive the goal location information, the first worker robot begins its journey to the goal. If an obstacle is encountered by the worker during its journey to a goal location, the obstacle is avoided, and the worker recalibrates its goal heading to account for the unexpected path change.



Legend:
E [90°]
S [180°]
N [0°]
W [270°]
Direction of Actual Motion
Direction of Actual Motion (after adjustment)
Direction of Desired Motion
Goal
Adjusted Desired Trajectory
Initial Desired Trajectory
θadj
ysum_adj
xsum_adj
xsum_i
D
Dsin(θi)
θi
Obstacle
Dcos(θi)
ysum_i

Figure 2.5: Graphical depiction of the worker robot's heading recalibration routine. The black dotted line indicates the robot's initial desired trajectory, and the solid black line indicates the robot's actual motion due to the obstacle being detected. The thick green line indicates the robot's recalibrated desired trajectory to the goal location.

Figure 2.5 shows how the robots work backwards to seek goals and recalibrate their desired goal trajectory to account for unexpected path alterations. Once the first worker discovers the goal location, any improvements on the path from the nest to the goal as determined by the worker are passed onto the next worker robot in succession. This results in an improvement in the path from the nest to the goal each time a worker robot locates the goal. Thus, armed only with data of the directional *vector* from the nest to the goal, the worker robots are able to seek goal locations in the presence of random, unknown obstacles without a priori knowledge of the environment.

3. The Experiment

This multi-agent control scheme was implemented using two identical robots. The experimental process involved the robots beginning from a common starting point and cooperating to determine the shortest solution to various maze-like environments. The scout robot began the experiment by randomly grazing the environment while keeping track of its relative location vector traveled from the nest. Once the scout discovered the goal location, it transmitted a wireless message containing the direction and displacement of the shortest straight-line path from the nest to the goal. Once the message was received by the worker robot waiting in the nest, the worker sought the goal location and avoided obstacles if necessary. The experiment was concluded when the worker robot found the goal location.

This chapter is divided into six sections. The first section describes the experimental setup including robots' physical dimensions, equipment utilized, experimental environment, and sensor capability. The second section describes the details involved with the robots' mathematical determination of their sense of direction. The third section explains the process and equipment involved with the wireless communication of goal location data. The fourth section describes the process involved with the worker robots seeking a pre-discovered goal location. The fifth section explains how this multi-agent control algorithm can be illustrated using a group consisting of only two agents. Finally, the sixth section describes and analyzes various data gathered in the experiment.

3.1 Experimental Set Up

The experiment was implemented using a pair of Mekatronix TJ Pro mobile robots. Both of the robots were equipped with identical Motorola MHC6811A9 microcontrollers and were programmed using the C programming language. Figures 3.1a through 3.1c show diagrams of the robots used in the experiment. As depicted in Figure 3.1a, each of the robots had two forward-looking infrared emitters and detectors for obstacle detection. The infrared detectors were set to detect obstacles at a distance of approximately 10 cm. Additionally, the robots had contact sensors built into the bumpers. If the infrared detectors failed to sense an imminent obstacle, the bumper sensors would detect an impact and the robot would respond accordingly.
The navigational capability of the robots was achieved using the Vector 2X electronic compass by Precision Navigation Instruments. The compass gave a resolution of roughly 2 degrees, which was sufficient for this experiment. The wireless communication was achieved using Ming Microsystems' 300 MHz amplitude modulated RF transmitter and receiver. Since communication was only required in one direction (scout robot to worker robot) for the experiment, the scout was equipped with a transmitter and the worker was equipped with a receiver as illustrated in Figures 3.1b and 3.1c. The environments in which the experiment took place were various maze-like environments. Figure 3.2 shows four of the experimental environments utilized.

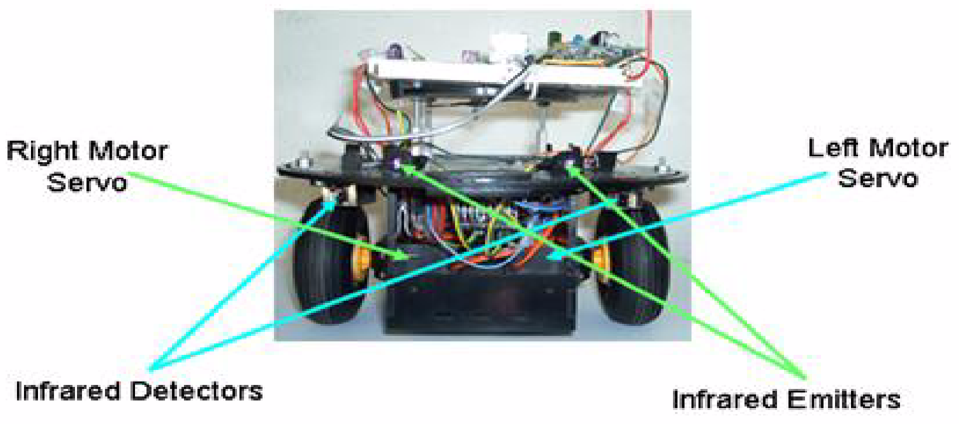


Figure 3.1a: *(Top)* Layout of infrared (IR) sensors for obstacle detection and motor servos for mobility. The worker and the scout robots have identical servo and IR sensor layouts.

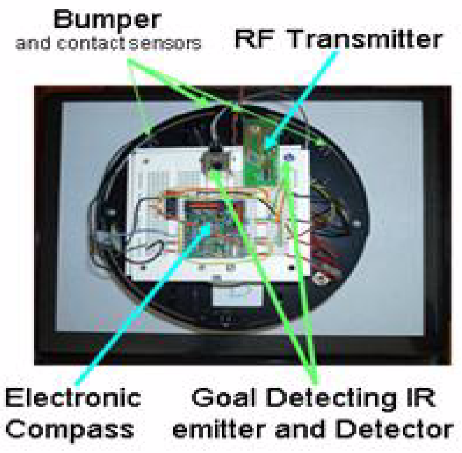


Figure 3.1b: (Left) Top view of Scout robot used in experiment showing various equipment employed

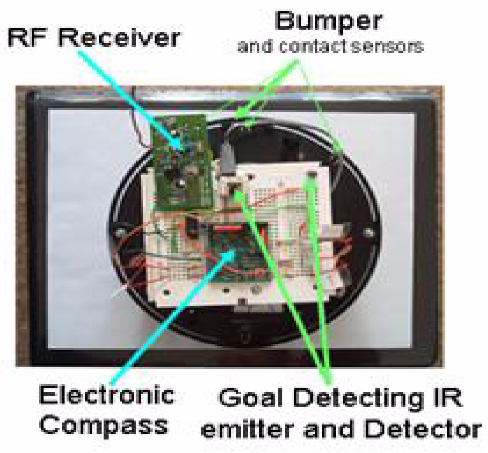


Figure 3.1c: (Right) Top view of Worker robot used in experiment showing various equipment employed

3.2 Robots' Sense of Direction

The crucial factor which allowed the robots to share meaningful target location information was that every robot involved in the experiment began from the same starting location, the nest. This provided all the robots with a common reference point to which relative location information (relative direction and displacement) could be shared.

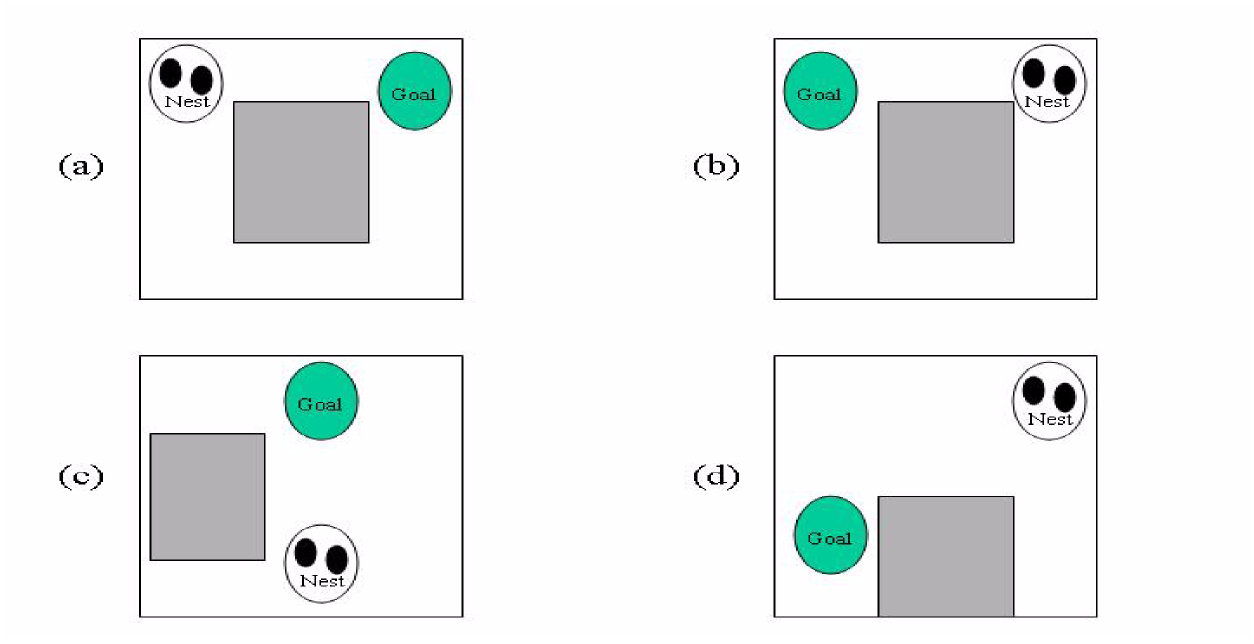


Figure 3.2

The general direction and displacement traveled from the nest was updated approximately twice per second as the robot moved forward. The rate of execution of the computer program itself determined the rate at which the general direction from the nest was updated. The computer program took approximately one-half of a second to execute one program loop; therefore, the compass heading data was sampled approximately once every half a second. Thus, calculations of general direction and displacement traveled from the nest were updated roughly twice per second. Each robot was calibrated to move at the same speed so that each increment of program execution time corresponded to an increment of distance, which was equal for both robots. Since the speed of the robots and the compass heading sampling period were constant, the robots' calculation of their general direction traveled, as depicted in Figures 2.3 and 3.3, is given as follows.

$$qg = \arctan\{(S\sin[qi]) / (S\cos[qi])\} \quad (3)$$

where qi denotes the instantaneous heading (in degrees) measured by the robot approximately twice per second. Equation 3 is merely a simplification of Equation 1 in which the velocity of the robot and the heading sampling period remain constant. The equation utilized to update the displacement from the nest is:

$$D = \{(S\sin[qi])^2 + (S\cos[qi])^2\}^{1/2} \quad (4)$$

Figure 3.3: Diagram showing the details involved in calculating the robot's general direction traveled using incremental measurements taken from the compass. Note that $D_i = 1$ since the distance traveled by the robot during one computer program loop is constant.

Equation 4 was a fairly quick calculation because the same quantities, Σcos[θi] and Σsin[θi], were used in Equation 4 as in Equation 3. Equation 4 is a simplification of Equation 2 with constant velocity and sampling time. The units for Equation 4 were in increments of program execution time as explained previously and can be approximated to be approximately 3 inches of displacement (meaning that the robots moved approximately 3 inches of distance in a half of a second of time). Figure 3.3 shows the mathematical derivation of Equations 3 and 4. As shown in Figure 3.3, Equation 1 was derived using a simple extension of the trigonometric relationship ταν(θ) = sin(θ)/cos(θ). This equation can be manipulated algebraically to yield:

$$q = \{\mathbf{arctan}[\mathbf{sin}(q)/\mathbf{cos}(q)]\} \qquad \mathbf{(5)}$$

Thus, Equations 3 and 1 are simply an extension of Equation 5. Due to the fact that the arctangent function, arctan(θ), only yields angular solutions between -90^o and 90^o, the robots had to be programmed to account for this discrepancy. This was done by programming the robots to make logical decisions about the sign of the sine and cosine quantities. The decision process was based upon the following reasoning: for angles between 0^o and 90^o, sine is positive and cosine is positive; for angles between 90^o and 180^o, sine is positive and cosine is negative; for angles between 180^o and 270^o, sine is negative and cosine is negative; for angles between 270^o and 360^o, sine is negative and cosine is positive. Thus, if the sine and the cosine were both negative, the result attained from equation (5) was added to 180^o in order to correct the discrepancy introduced by the arctangent function. In the diagram on the left hand side of Figure 3.3, the small black arrows represent the instantaneous heading vectors, which were sampled approximately once every 0.5 seconds as the robot moved forward. The small black arrows, therefore, represent approximately 3 inches of distance. Figure 3.3 illustrates how the general direction traveled was determined by calculating the cosine and sine of the instantaneous heading values and summing the x-components (cosine components) and the y-components (sine components) and calculating the inverse tangent of the ratio of the sum of the y-components (ysum) to the sum of the x-components (xsum). Additionally, Figure 3.3 illustrates the derivation of Equation 4. As shown in the Figure 3.3, the displacement calculation given in Equation 4 was determined from the Pythagorean Theorem. The total displacement is represented in Figure 3.3 by the hypotenuse of the right triangle formed by legs of lengths xsum and ysum. The quantities xsum and ysum are simply the summation of the x components and y components, respectively, of the robot's instantaneous heading. The instantaneous x and y components are given by the cosine and sine, respectively, of the instantaneous heading due to the fact that the instantaneous displacement, D_i in the figure, is taken to be 1. Thus, the displacement was calculated as the square root of the sum of the squares of xsum and ysum as shown in Figure 3.3. The only time the general direction vector calculation was updated was when the robot was moving forward or backwards; therefore, directional calculations were only updated during those times. If an obstacle was encountered, the robot simply rotated in place to avoid it. This improved the overall directional calculation because no distance was traversed as the robot was turning to avoid obstacles. In the event of a bumper impact, the robot would back up for about a half of a second (approximately one program execution interval) and would then proceed. The general direction calculation was updated to account for backing up also. During backing, the cosine and sine increments from Equation 3 were subtracted from xsum and ysum instead of being added.

Figure 3.4 illustrates an example of a robot using the techniques described here to seek a goal in the presence of two obstacles. In step 1 of Figure 3.4, the robot knows the general direction to the goal (its desired heading), and so it begins traveling in that direction. In step 2, the robot senses an obstacle due to an IR detection and reacts by turning (left) to avoid it. The robot continues to move forward after the obstacle is avoided until it detects that its heading is

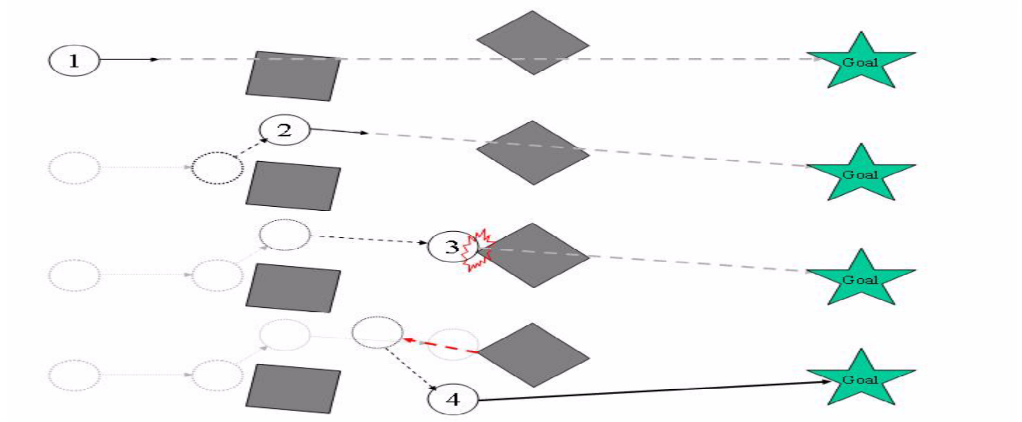


Figure 3.4: Qualitative depiction of a robot navigating an environment containing 2 obstacles. The first obstacle is avoided due to an IR detection, and the second is avoided after a bumper impact. The final heading (shown as a solid black line) was recalibrated according to the techniques shown in Figures 2.5 and 3.3.

different than its desired heading by some preset value. The robot then readjusts (rotates in place) to match its new desired heading as calculated using the method outlined in Figure 2.5. The robot then continues to proceed in its recalibrated desired direction. In step 3 of Figure 3.4, the failure of the IR detectors results in an impact on an obstacle. Sensing the bumper impact, the robot backs up for approximately one half of a second (one program loop or approximately 3 inches), rotates slightly, and then proceeds forward. As the robot moves forward and recalibrates its desired heading, it detects that its current heading differs from its desired heading by too much and rotates in place until its heading matches its desired heading. Finally, the robot's desired heading does not have any obstacles in its path, and the robot makes it to the goal location unobstructed. This illustration shows the benefit of using this control scheme in a navigational task. With a sense of direction this good, who needs a map!

3.3 Inter-agent Communication

The inter-agent communication involved in this experiment was achieved using Ming Microsystems' 300 MHz amplitude modulated RF transmitter and receiver. The goal location information transmitted by the scout consisted only of trajectory information (1-byte) and displacement information (1-byte) along with a checksum value for each byte (refer to Figure 3.5 for the exact data transmission pattern). The checksums appended to the trajectory and

Dislpacement Byte's Checksum Value (one byte)	Displacement from Nest to Goal [software time increments] (one byte)	Direction Byte's Checksum Value (one byte)	Direction from Nest to Goal [deg] (one byte)

Example Message Transmitted by Scout

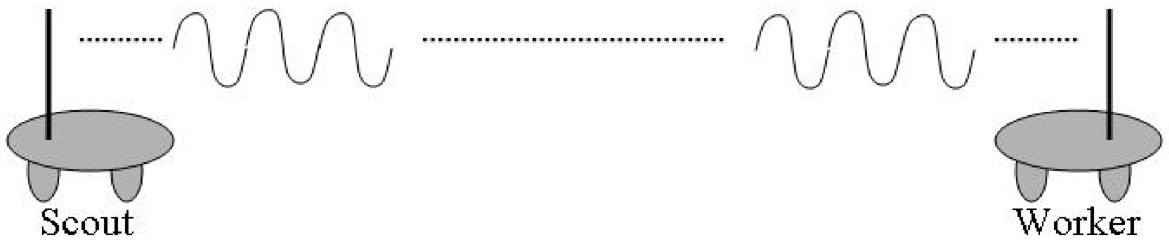
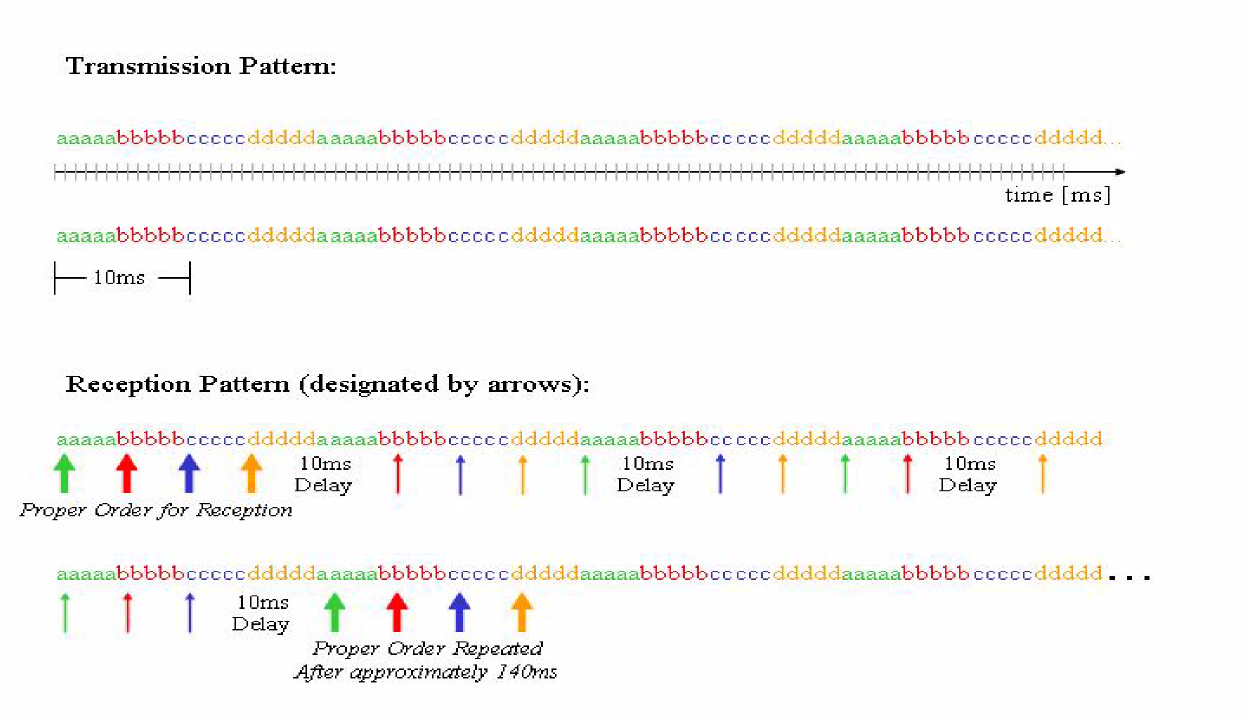


Figure 3.5

displacement data bytes were calculated using the 8-bit Cyclic Redundancy Check (CRC) method. Error detection was enhanced by using the division form of this method. In order to reduce the processing power required to communicate data, the directional data was scaled down for transmission to **{qg : 0 < qg < 128}** so that the transmitted data was represented as a 7-bit character in the ASCII data set (7-bit => [0000000 ~ 1111111] => [0 ~ 127])**.** This reduced the resolution of the directional data received from the electronic compass to approximately 2.8 degrees, which was sufficient for the experiment. The displacement value was rounded to the nearest whole number. In order to assure reliable one-way communication, the data were sent in long numerical streams. An example of a typical transmission pattern is shown below:

25,25,25,25,25,2,2,2,2,2,9,9,9,9,9,3,3,3,3,3,25,25,25,25,25,2,2,2,2,2,9,9,9,9,9,3,3,3,3,3...

A more detailed explanation of the transmission and reception patterns is presented in Figure 3.6. As indicated in Figure 3.6, each of the bytes was sent in streams of five. The reason for repeating each data byte was to synchronize the transmission and reception patterns of the robots. On the receiving end, the computer program executed a 'receive' command followed by a 'wait' command. This resulted in the receiving robot receiving one data byte every 5 milliseconds approximately. The choice to repeat each data byte five times was arbitrary, and synchronization could have been achieved using many different repetition patterns. The first five data in the



Transmission Pattern:
aaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccddddd...
time [ms]
aaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccddddd...
10ms
Reception Pattern (designated by arrows):
aaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccddddd
10ms Delay
10ms Delay
10ms Delay
Proper Order for Reception
aaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccdddddaaaaabbbbbcccccddddd...
10ms Delay
Proper Order Repeated After approximately 140ms

Figure 3.6: Illustration of the transmission and reception patterns used by the robots in the experiment in order to maximize the reliability of one-way communication. The letters a, b, c, and d, represent the directional data [deg], direction checksum, displacement [3 inch increments], and the displacement checksum, respectively. Each of these data items were represented by one byte of information.

above stream are the scaled compass heading value from the nest to the goal, the second five are the checksum value for the compass heading, the third five are the displacement from the nest to the goal, and the fourth five data are the checksum for the displacement value. The pattern was repeated as shown. The timing was carefully selected in order to make sure that the data would be received in the proper order. Each individual byte took approximately one millisecond to be transmitted, so one individual data took about 5 ms to be transmitted because it was sent in streams of 5. The software driving the receiver was programmed to sample the incoming data at a rate of approximately 1 byte per 5 ms, but a pause of approximately 10 ms (as shown in the Figure 3.6) was programmed to occur after four groups of data were received. The colored arrows indicate the reception pattern used by the receiving robot. The colors of the arrows in Figure 3.6 serve to indicate the order in which the data are received. Since the data must be received in the correct order to assure proper communication, it is imperative that the receiving robot samples the incoming data such that proper sequencing of data is certain to be achieved. In Figure 3.6, it is required that the receiving robot receives the data in the order a, b, c, and then d. The color sequence of the arrows in the reception pattern in Figure 3.6 must therefore occur in the order green, red, blue, then orange. As illustrated in Figure 3.6, the correct sequencing of the reception pattern occurs approximately once for every fourth occurrence of a 10 ms delay. This way, the correct sequencing of received data (direction byte, direction checksum byte, displacement byte, displacement checksum byte => green, red, blue, orange) was certain to happen approximately once every 140 milliseconds or roughly 7 times per second.

3.4 Seeking a Pre-discovered Goal Location

Once the worker robot successfully received a goal location message, it began its journey. The worker first rotated in place until its current heading matched (to some error margin) the desired heading. The displacement data received by the worker allowed it to recalibrate its desired heading in the event of an unexpected course alteration. In actuality, the robot was constantly recalibrating its desired heading to account for the slight deviations from the desired heading measured in each compass reading. The slight fluctuations in the compass readings were negligible, though, since the deviations were so small. Large path alterations due to obstacle avoidance necessitated the implementation of the recalibration routine. As depicted in Figure 3.7, imminent obstacles were avoided by the robot, and the desired heading was adjusted when the robot detected that its heading was not within the prescribed error margin. The error margin (represented by the large gray triangles in Figure 3.7) is what enabled the (worker) robot to successfully seek the location of a goal in the presence of unknown obstacles. The reason for this was that the error margin provided the robot with flexibility in its selected trajectory, thereby enabling it to go around obstacles when encountered. Step (a) of Figure 3.7 shows the robot moving in the direction of the goal (its desired heading). Step (b) indicates the benefit of programming the worker robots with an error margin. In step (b), the robot senses an obstacle due to an IR detection and turns to avoid it. As the robot moves forward, its current heading no longer matches the desired trajectory. In step (b), the robot determines that its heading is outside the error margin of the desired heading. As shown in step (b), however, the robot was able to pass the obstacle before readjusting its heading to match the desired heading. Finally, in step (c) the robot readjusts its heading and finds the goal location. The worker robot in this experiment was programmed with a sufficient allowable error margin between its current and desired headings such that goal locations could be sought in the presence of unknown obstacles.

When the worker detected an obstacle on its way to the goal, it avoided the obstacle and then recalibrated its desired goal trajectory using Equation 6 (see below). Figure 2.5 shows an example of the heading calibration routing using Equation 6. In Equation 6, θ_{adj} (shown in Figure 2.5) represents the new desired heading after the robot's path was altered due to obstacle avoidance. The quantities ysum and xsum are indicated in the triangle diagram on the right side of Figure 3.3. The D term in Equation 6 represents the distance traveled by the robot to avoid an obstacle. The D term was taken to be one for calculations involving a single increment of the heading sample. The quantities xsum and ysum represent the lengths of the legs of the right triangle whose hypotenuse is the vector indicating the direction and displacement from the nest

$$\begin{aligned} &\mathbf{qadj = arctan\{ysum_{adj} / xsum_{adj}\},} \\ &\mathbf{where \quad ysum_{adj} = ysum - Dsin[qi]} \\ \mathbf{\&} \quad &\mathbf{xsum_{adj} = xsum - Dcos[qi]} \qquad \mathbf{(6)} \end{aligned}$$

to the goal. The quantities $ysum_{adj}$ and $xsum_{adj}$ (refer to Figure 2.5) represent the new values for ysum and xsum after the robot's path was altered due to obstacle avoidance. The quantities Dsin[θi] and Dcos[θi] represent the deviations from the original path caused by an unexpected detour due to an obstacle.

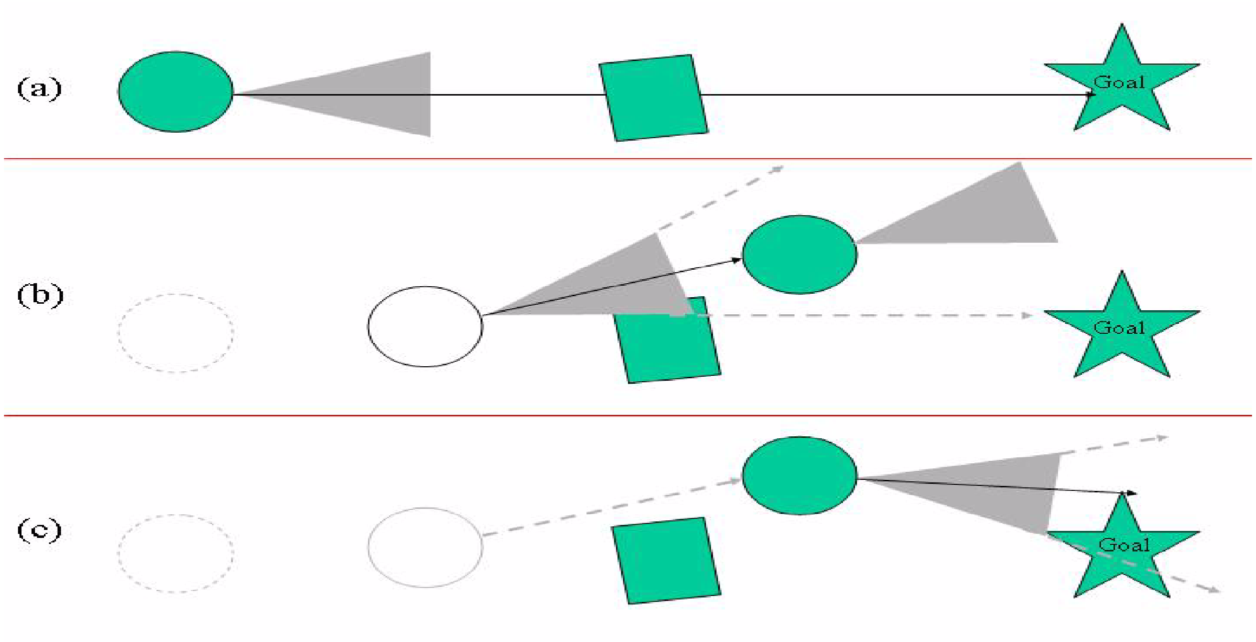


Figure 3.7: Illustration of the error margin (gray triangle) programmed into the robots for the purpose of altering the desired path for obstacle avoidance. The gray triangle represents the robot's allowable error margin between the robot's desired heading and its measured heading.

3.5 Reduction to Two Robots

Although this approach to multi-agent control was intended to facilitate the implementation of robot groups consisting of several members, an experiment involving a team consisting of only two robots can demonstrate the fundamental idea of the approach. This was done by dubbing one robot the 'scout' and the other a 'nest worker.' The scout and nest worker roles are the only two roles that exist in the multi-agent control scheme described here regardless of the total number of robot agents involved in the group. In this two robot experiment, the nest worker robot represents any number of robots in the nest that are waiting for the scout (or another worker) to transmit goal location information. The scout robot in the simplified experiment has exactly the same responsibility as it would in a scenario involving many robots. The scout robot was initially deployed from the nest alone and retained its sense of direction as it searched an unknown environment in a random fashion until the goal was detected. At that point, the scout marked its displacement from the nest and its general direction traveled from the nest. The scout then transmitted the goal location data to the nest worker robot waiting in the nest. The nest worker then had enough information to seek out the goal. The nest worker began its journey to the goal and ‘watched’ for obstacles in its path. If it detected an obstacle on the way to the goal, it avoided the obstacle and then recalibrated its desired goal trajectory using Equation 6. Thus, the nest worker was able to seek out the goal in the presence of any random, unknown obstacles given only the location information gathered by the scout. If there were several nest worker robots, each of them would simply view the others in the group as obstacles and avoid them as necessary while retaining knowledge of the goal’s relative location. Since all of the nest workers would have the entire vector of goal location information (direction and displacement), they would all be able seek out the goal location regardless of unexpected robots or obstacles in their path. The simplified two-robot experiment illustrated the benefit of layering the algorithm in a two-tier fashion. The bottom, or most fundamental, layer is obstacle avoidance. The obstacle avoidance algorithm is constantly being executed by the robots as they explore. The second layer is the updating of the goal’s relative location at regular time intervals as calculated using Equations 3 and 4. Enabling the robots with a sense of direction allowed the robots to evaluate simple mathematical calculations that gave the robots the ability to find a goal in any unknown environment given only the direction and displacement of the goal from a common starting point.

One important aspect of this multi-agent control scheme that was not demonstrated in the two robot experiment was the improvement over time of the robots’ paths to the goal by virtue of the redundant information gathered by several robots working in succession. After the first worker discovered the goal from the information given to it by the scout, that worker would send a message to the rest of the workers waiting in the nest. This new message would contain any possible improvements (i.e., location of obstacles, shorter path to the goal, etc.) to the path to the goal as discovered by the first worker during its journey. If no improvements were discovered by the second worker, no new message would be sent. The second worker would then have an improved version of the data indicating the path to the goal. The second worker would follow the same process and send an updated message to the rest of the workers when it found the goal if it measured its path to have an improvement over the old path. The third worker, therefore, has better information than the second worker. This process would continue as long as workers are present in the nest. By this method of redundancy of gathered information, the data gathered by the workers would only improve as more and more workers discover the goal. Thus, adding additional agents can result in an overall improvement in the efficiency of this control scheme.

3.6 Experimental Data Analysis

The results show that under the control of this algorithm a multi-agent robot group successfully navigated various unknown environments given a very simple set of rules and minimal sensory equipment. As indicated in Figure 3.8, after the scout robot grazed the environment and discovered the goal, the worker robot took the shortest path to reach the goal, thereby reducing the travel time significantly. The black arrows in Figure 3.8 mark the path taken by the scout robot during its initial searching trip, and the red arrows mark the path taken by the worker robot after receiving the goal location information gathered from the scout. Approximate travel times for worker and scout robots are given in the gray squares (obstacles) in Figure 3.8. Clearly, the path to the goal chosen by the worker robot was much more efficient than that of the scout robot in every case. This two robot experiment demonstrated that after only one iteration, a significant amount of learning occurred between the robots under the control of this scheme. Furthermore, a surprising degree of robustness was shown by the agents in this experiment. In several instances, sensory and equipment failures occurred that did not significantly diminish the overall success of the robots' navigation.

Some sources of error encountered in this experiment were erroneous data received from the electronic compass, long (0.5 second) delays between heading sampling times, erroneous data reception, skidding of the robots' tires on the ground, and slight differences in the speeds of the scout and worker robots' motor servos. The electronic compass incurred some error in its measured heading due to electrical noise and ambient magnetic fields. Since these errors fluctuated between positive and negative discrepancies, their combined effect was minimal and did not significantly hinder the performance of the system. The delay between successive compass heading sampling times caused the robots to calculate slightly degraded values for general direction information. This caused slight discrepancies between measured and actual goal locations. Since the robots were programmed to allow a certain error margin in their heading measurements, the overall performance was not significantly hindered by the delay. Since the inter-agent communication in this experiment was one-way only, erroneous data reception was a problem. This problem can be remedied by utilizing two-way communication with handshakes. This would assure that the receiving robot acted upon the correct data. The difference in the speeds of the robots was usually caused by the different rates of power consumption by each of the robots. This, too, caused slight miscalculations in goal location data but did not compromise the success of the control scheme. The ability of this control scheme to
function adequately in the presence of errors is one indication of the benefit of the simplicity of this control scheme. Some of the errors encountered in this experiment can be remedied, however, and this issue will be addressed later in chapter 4.

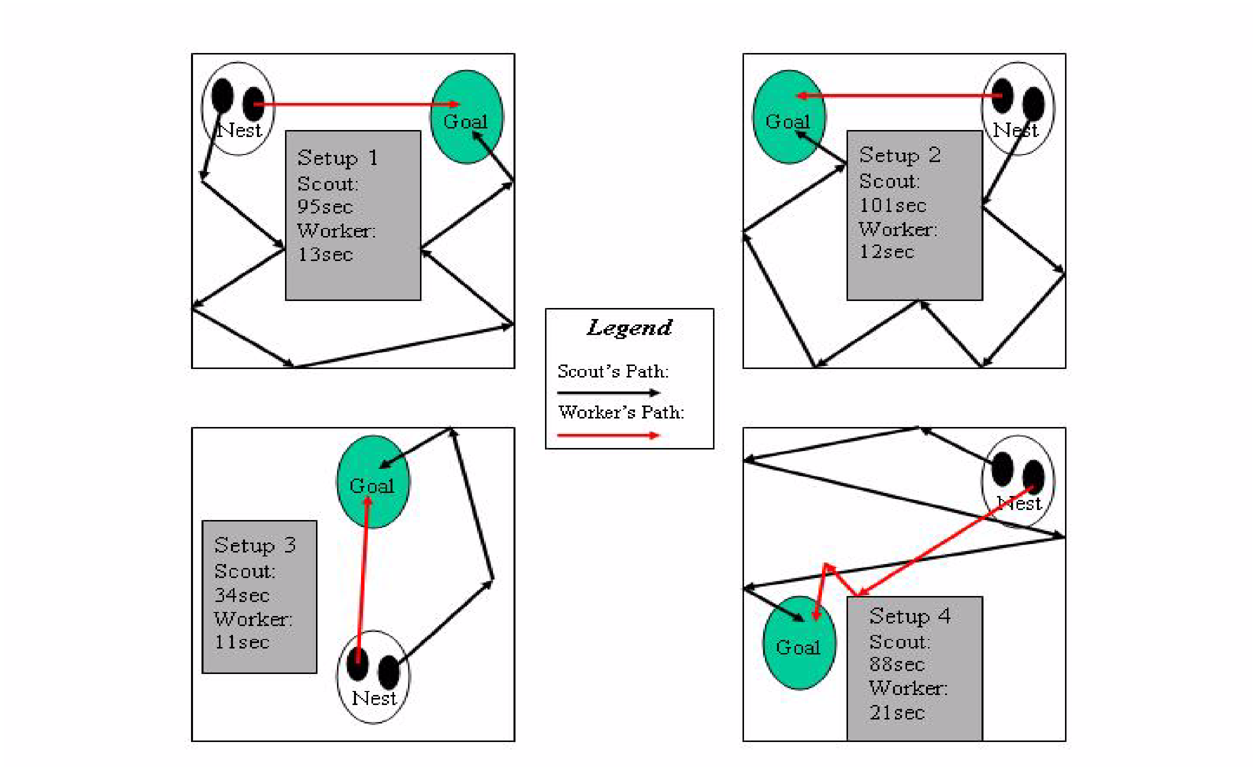


Figure 3.8

The results further demonstrated that only minimal sensing and equipment are required in order for a robot group to successfully navigate an unknown environment *without* the use of a map. This shows that this control scheme offers inexpensive, practical implementation due to its minimal nature, and demonstrates rather robust results as shown in schemes that use much more cumbersome calculations and elaborate sensory equipment.

4. Conclusions

The multi-agent control scheme presented here showed promising results by virtue of redundant sharing of simple information between minimal agents. It is unnecessary for each of the robots in a multi-agent group to have sophisticated equipment or complicated mathematical algorithms in order to successfully navigate unknown environments. If the agents each gather and share location information with one another, successful navigational patterns will emerge as a result of the incremental improvement of the group's overall information due to the contributions of each individual agent. Furthermore, minimalism of the agents in a multi-agent group is necessary for practical implementation of groups consisting of many agents. This scheme demonstrated successful navigation by a multi-agent group using very simple rules and inexpensive sensory equipment.

It is evident from the experimental results that the navigational scheme presented here can be scaled up to control groups consisting of many robots while still retaining its robustness and simplicity. The ability of this multi-agent control scheme to recover from sensory malfunctions and regain control after erroneous data reception is an indication of the level of robustness that can be achieved by implementing this scheme. Furthermore, reduced numerical accuracy of directional data does not compromise the system's overall performance. The capability of this scheme to successfully navigate an unknown environment using erroneous and/or low resolution data suggests that this scheme can be even further simplified without loss of functionality. In conclusion, this highly simplified control scheme implemented with redundancy of information communicated between agents can result in the emergence of sophisticated navigational patterns for multi-agent robot groups comprised of any type of mobile robot in virtually any environment.

4.1 Future Work

Several factors contributed to the implementation of this control scheme. These factors included an electronic compass for directional information, a microcontroller for data processing, RF equipment for data communication, infrared and contact sensors for obstacle detection, and DC servos for mobility. It is possible, however, to implement this control scheme utilizing much simpler equipment while still achieving similar results.

The experiment outlined here utilized only one-way communication between the agents, which is not very reliable. One way to improve this scheme would be to use two-way communication with handshakes. Implementing any sort of reply back to the transmitting agents from receiving agents would improve the reliability of inter-agent communication.

The simple reactive nature of the algorithm would permit the use of a smaller processor. Even better, no processor at all. A group comprised of fully electronic agents can achieve similar results without a central processor. This would drastically reduce the cost of implementation of this scheme. Refer to Figure 4.1 for an illustration of a fully electronic agent to be used with this scheme. This will be done utilizing a simpler compass in conjunction with a Boolean network. In this version, the deviation between the desired direction and current direction itself will control the motor servos, thereby guiding the agents to their desired destination without the use of a CPU. In this digital electronic version, a simple electronic compass with a resolution of approximately 45 degrees will be used for directional measurements. The output of the compass is a 3-bit binary number from 000 ~ 111 {0 - 7}, which represents the eight compass directions, E, NE, N, NW, W, SW, S, SE. This simple 3-bit number will be recorded incrementally by the robots as they move forward just as in the aforementioned scheme, but this time using simple electronic binary counters with delay circuits instead of using a CPU with if and then statements. The use of a binary counter for timing the compass readings would improve the accuracy of the robots' directional calculations substantially. Obstacle avoidance will be done using tactile sensors (whiskers) in conjunction with simple make/break circuits. Since turning left and turning right for a two-wheel, mobile robot are simply matters of turning the right motor servo on the left off and vice versa, the use of a CPU to control the agents' servos in this scheme is actually

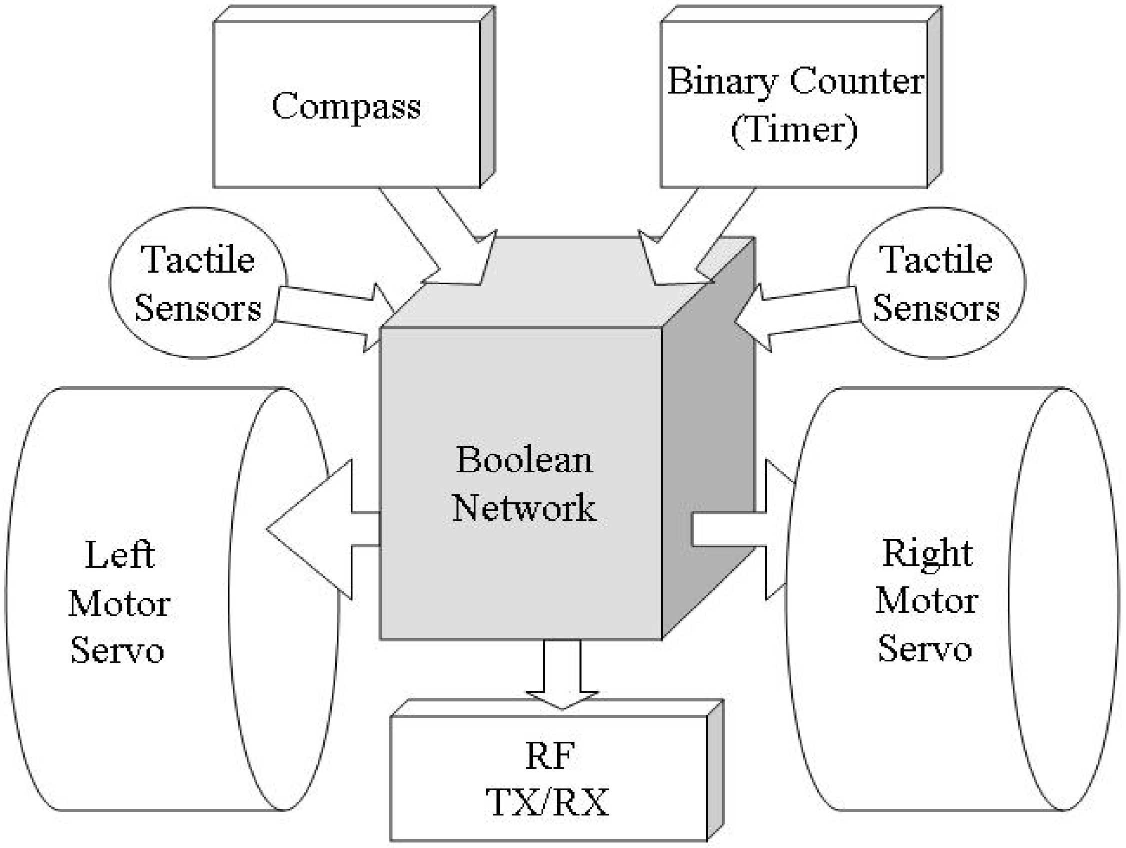


Figure 4.1: Qualitative diagram of a fully electronic agent utilizing this control scheme

unnecessary. The wireless communication between the electronic agents can be achieved quite simply with the use of frequency-shift-keying (FSK). Using FSK modulation, the amplitude of the transmitted carrier signal will indicate one of the eight compass directions. The lowest amplitude would represent 000 (East), and the highest amplitude would represent 111 (Southeast). Although this would significantly reduce the resolution of the directional data, the experimental results gathered from the experimentation outlined here suggest that the reduced numerical accuracy will not compromise the scheme's overall performance. Moreover, the results of the original experiment showed that the provision of an error margin between the desired trajectory and the current (measured) trajectory was what allowed the agents in this scheme to seek pre-discovered goal locations in the presence of obstacles. Thus, the reduced resolution of the compass used in an electronic version of this scheme will provide a natural error margin for the agents. This will enable them with enough flexibility in their desired trajectories to circumnavigate random, unknown obstacles.

The ability to exemplify this control scheme using a fully electronic version is one indication of the minimalism of this control scheme. It is not necessary to increase the complexity of a multi-agent control scheme in order to attain profound results. Simple agents reacting *locally* to their environment can gather and share simple *global* information about the environment, thereby resulting in a quite profound cooperative navigational dynamic.

Bibliography

[1] Arkin, R., Balch, T., "Cooperative Multi-Agent Robotic Systems." 1996.

[2] Balch, T., Arkin, R., "Behavior-based Formation Control for Multi-robot Teams." *IEEE Transactions on Robotics and Automation.* 1999.

[3] Balch, T., Arkin, R., "Communication in Reactive Multi-Agent Systems." 1994.

[4] Beard, R., Lawton, J., Hadaegh, F., "A Coordination Architecture for Spacecraft Formation Control." *IEEE Transaction on Control Systems Technology.* July, 2000.

[5] Castelpietra, C., Iocchi, L., Nardi, D., Rosati, R., "Coordination in Multi-Agent Autonomous Cognitive Robotic Systems." 2000.

[6] Dale, L.K., "Optimization Techniques for Probabilistic Road Maps." *Ph.D. Dissertation.* December, 2000.

[7] Dedeoglu, G., Sukhatme, G., "Landmark-based Matching Algorithm for Cooperative Mapping by Autonomous Robots." 1999.

[8] Fujimura, K., Singh, K., "Planning Cooperative Motion for Distributed Mobile Agents." 1995.

[9] Gagne, D., Pang, L., Trudel, A., "A Spatio-Temporal Logic for 2D Multi-Agent Problem Domains." 1996.

[10] Ghavamzadeh, M., Mahadevan, S., "Continuous-Time Hierarchical Reinforcement Learning." 2001.

[11] Goldberg, D., Mataric, M., "Robust Behavior-Based Control for Distributed Multi-Agent Collection Tasks." 2000.

[12] Han, K., Veloso, M., "Automated Robot Behavior Recognition Applied to Robotic Soccer." 1999.

[13] Jung, D., Zelinsky, A., "Grounded Symbolic Communication between Heterogeneous Cooperating Robots." 1999.

[14] Kimmel, R., Kiryati, N., Bruckstein, A. M., "Multi-Valued Distance Maps for Motion Planning on Surfaces with Moving Obstacles." 1997.

[15] Kuffner, J. Jr., Kagami, S., Inaba, N., Inoue, H., "Dynamically-stable Motion Planning for Humanoid Robots." 2000.

[16] Leroy, S., Laumond, J.P., Simeon, T.S., "Multiple Path Coordination for Mobile Robots: A Geometric Algorithm." 1998.

[17] Makar, R., Mahadevan, S., Ghavamzadeh, M., "Hierarchical Multi-Agent Reinforcement Learning." *Agents' 01. 2001.*

[18] Murray, D., Little, J., "Using Real-time Stereo Vision for Mobile Robot Navigation."

1998.

[19] Pirjanian, P., Mataric, M., “ Multi-Robot Target Acquisition Using Multiple Objective Behavior Coordination.” 1999.

[20] Rabie, T., “Animat Vision: Active Vision for Artificial Animals.” 1999.

[21] Rekleitis, I.M., Dudek, G., Milios, G., “Multi-Robot Collaboration for Robust Exploration.” 2000.

[22] Schneider-Fontan, M., Mataric, M., “Territorial Multi-Robot Task Division.” *IEEE Transactions on Robotics and Automation.* June, 1998.

[23] Se, S., Lowe, D., Little, J., “Vision-based Mobile Robot Localization and Mapping Using Scale-invariant Features.” 2001.

[24] Sim, R., “To Boldly Go: Bayesian Exploration for Mobile Robots.” May, 2000.

[25] Simmons, R., Apfelbaum, D., Fox, D., Goldman R., Haigh, K.Z., Musliner, D., Pelican, M., Thrun, S., “Coordinated Deployment of Multiple, Heterogeneous Robots.” 2000.

[26] Suryadi, D., Gmytrasiewicz, P., “Learning Models of Other Agents Using Influence Diagrams.” 1998.

[27] Sutton, R., Precup, D., Singh, S., “Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning.” July, 1999.

[28] Svestka, P., Overmars, M., “Coordinated Path-Planning for Multiple Robots.” 1997.

[29] Tambe, M., “Teamwork in Real-world Dynamic Environments.” 1996.

[30] Ulrich, I., Borenstein, J., “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots.” *Proceedings of the 1998 International Conference on Robotics and Automation. Leuvan, Belgium, May 16-21, 1998. Pages 1572-1577.*

[31] Werger, B. B., “Ayllu: Distributed Port-Arbitrated Behavior-Based Control.” 2001.

[32] Werger, B.B., “Cooperation Without Deliberation: A Minimal Behavior-based Approach to Multi-robot Teams.” December, 1998.

[33] Werger, B.B., Mataric, M., “Exploiting Embodiment in Multi-Robot Teams.” 2000.

[34] Werger, B.B., Mataric, M., “From Insect to Internet: Situated Control for Networked Robot Teams.” 2000.

[35] Ziemke, T., “Adaptive Behavior in Autonomous Agents.” *Autonomous Agents.* December, 1998.

[36] Ziemke, T., “The Construction of Reality in the Robot: Constructivist Perspectives on Situated Artificial Intelligence and Adaptive Robotics.” 2000.

WILLIAM MACKUNIS is a Masters Degree candidate at Florida Atlantic University in Boca Raton, FL. Specializing in control systems and robotics under the advisement of Dr. Daniel Raviv, William plans to begin a career within the Space Program after graduation in May of 2003.

Dr. DANIEL RAVIV is a professor in the electrical engineering department at Florida Atlantic University. He was the main driving force behind the installment of the course "Inventive Problem Solving" at Florida Atlantic University, which has recently become available at various high schools in Florida.