

S-NETS: SMART SENSOR NETWORKS

by

Yu Chen

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Bioengineering

The University of Utah

December 2000

Copyright © Yu Chen 2000

All Rights Reserved

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Yu Chen

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Thomas C. Henderson

Richard A. Norman

Charles D. Hansen

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of _____ Yu Chen _____ in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Thomas C. Henderson
Chair, Supervisory Committee

Approved for the Major Department

Kenneth W. Horch
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

This thesis shows that the utilization of nonmobile, distributed sensor and communication devices by a team of mobile robots offers performance advantages in terms of speed, energy, robustness and communication requirements.

At one extreme, mobile robots can be provided with a wealth of on-board sensing, communication and computational resources; at the other extreme, robots with fewer on-board resources can perform their tasks in the context of a large number of stationary devices distributed throughout the task environment. We call the latter approach *Smart Sensor Networks*, or S-Nets.

Models of mobile robots with on-board sensors, communication protocol and the *S-Net* system are established. Algorithms are defined for the *S-Net* which perform cooperative computation and provide information about the environment. Local frames and reaction-diffusion patterns are produced to help multiple mobile robots cooperate to perform various behaviors. The sample behaviors include: one mobile robot goes to a temperature source; multiple mobile robots surround a temperature source, and multiple mobile robots go back and forth to a temperature source. The simulation experiments show that *S-Nets* perform well, and particularly robust with respect to noise in the environment. System cost versus performance is studied, and guidelines are formulated for which the *S-net* system out-perform the *non-S-Net* system.

To My Family

CONTENTS

| | |
|---|-----|
| ABSTRACT | iv |
| LIST OF FIGURES | vii |
| CHAPTERS | |
| 1. INTRODUCTION | 1 |
| 1.1 Thesis Statement | 1 |
| 1.2 Example Scenarios | 1 |
| 1.3 Study Design | 2 |
| 2. MODEL DEVELOPMENT | 4 |
| 2.1 Mobile Robot Model | 4 |
| 2.2 Communication Model | 8 |
| 2.3 Smart Sensor Network Model | 12 |
| 2.4 Simulation Model | 12 |
| 3. ALGORITHMS FOR SMART SENSOR NETWORKS | 14 |
| 3.1 <i>CoordinateFrameDetermination</i> ¹ | 14 |
| 3.1.1 Global Frame Calculation | 16 |
| 3.1.2 Constructing a Local Frame | 17 |
| 3.1.3 Moving between Local Frames | 20 |
| 3.2 <i>Reaction – DiffusionPatterns</i> ² | 22 |
| 3.3 Level Set Method | 27 |
| 3.3.1 Path Planning | 27 |
| 4. MOBILE ROBOT BEHAVIORS | 32 |
| 4.1 Goal Achievement | 34 |
| 4.2 Multiple Robot Behaviors | 36 |
| 5. PERFORMANCE EVALUATION | 37 |
| 5.1 One Robot Goes to a Temperature Source | 40 |
| 5.2 Multiple Robots Surround Temperature Source Evenly | 51 |
| 5.3 Multiple Robots Go Back and Forth to the Temperature Source ... | 59 |
| 6. CONCLUSIONS AND FUTURE WORK | 74 |
| REFERENCES | 76 |

LIST OF FIGURES

| | |
|---|----|
| 2.1 Local Frame | 5 |
| 3.1 The formulation of the global frame. | 19 |
| 3.2 Construction of a local frame. | 19 |
| 3.3 Transformation between frames. | 23 |
| 3.4 Box Pattern | 23 |
| 3.5 Result (bit=1) of distributed reaction diffusion algorithm with random distribution s-elements | 24 |
| 3.6 Result (bit=0) of distributed reaction-diffusion algorithm with random distribution of s-elements | 24 |
| 3.7 Stripe Pattern | 26 |
| 3.8 Irregular Pattern | 26 |
| 3.9 Speed Function F with Robot | 29 |
| 3.10 Time of Arrival Function T | 29 |
| 3.11 Contours of T | 30 |
| 3.12 Shortest Path | 30 |
| 4.1 Gradient Calculation | 33 |
| 4.2 Simulation of a Mobile Robot Moving against Temperature Gradient | 33 |
| 5.1 Robot goes to destination without S -Net | 41 |
| 5.2 Robot goes to destination with S -Net | 41 |
| 5.3 Time vs. Number of S-Elements with S -Net (Range = 2) | 43 |
| 5.4 Distance Traveled vs. Number of S-Elements with S -Net (Range = 2) | 43 |
| 5.5 Final Distance to Source vs. Number of S-Elements with S -Net (Range = 2) | 44 |
| 5.6 Average Distance of S-Elements vs. Number of S-Elements in the Area | 44 |
| 5.7 Time vs. Range with S -Net (Number of S-Elements = 250) | 45 |
| 5.8 Distance Traveled vs. Range with S -Net (Number of S-Elements = 250) | 45 |
| 5.9 Final Distance to Source vs. Range with S -Net (Number of S-Elements = 250) | 46 |

| | | |
|------|---|----|
| 5.10 | Time vs. Variance of Noise With <i>S-Net</i> (Number of S-Elements = 300, Range = 2) | 46 |
| 5.11 | Time vs. Variance of Noise Without <i>S-Net</i> | 47 |
| 5.12 | Distance Traveled vs. Variance of Noise With <i>S-Net</i> (Number of S-Elements = 300, Range = 2) | 47 |
| 5.13 | Distance Traveled vs. Variance of Noise Without <i>S-Net</i> | 48 |
| 5.14 | Final Distance to Source vs. Variance of Noise With <i>S-Net</i> (Number of S-Elements = 300, Range = 2) | 48 |
| 5.15 | Final Distance to Source vs. Variance of Noise Without <i>S-Net</i> | 49 |
| 5.16 | Robot goes to destination without <i>S-Net</i> (with noise variance = 10) | 50 |
| 5.17 | Robot goes to destination with <i>S-Net</i> (with noise variance = 10) | 50 |
| 5.18 | Robot goes to destination without <i>S-Net</i> (with noise variance = 8) | 51 |
| 5.19 | Robot goes to destination with <i>S-Net</i> (with noise variance = 8) | 52 |
| 5.20 | Three Robots Surround the Temperature Source Without <i>S-Net</i> | 53 |
| 5.21 | Three Robots Surround the Highest S-Element with <i>S-Net</i> | 53 |
| 5.22 | Time vs. Number of S-Elements for Robots with <i>S-Net</i> (Range = 2) | 55 |
| 5.23 | Distance Traveled vs. Number of S-Elements for Robots with <i>S-Net</i> (Range = 2) | 55 |
| 5.24 | Final Distance to Source vs. Number of S-Elements for Robots with <i>S-Net</i> (Range = 2) | 56 |
| 5.25 | Time vs. Range for Robots with <i>S-Net</i> (Number of S-Elements = 300) | 56 |
| 5.26 | Distance Traveled vs. Range for Robots with <i>S-Net</i> (Number of S-Elements = 300) | 57 |
| 5.27 | Final Distance to Source vs. Range for Robots with <i>S-Net</i> (Number of S-Elements = 300) | 57 |
| 5.28 | Robots Surround the Temperature Source without <i>S-Net</i> (with noise variance = 10) | 60 |
| 5.29 | Time vs. Variance of Noise for Robots with <i>S-Net</i> (Number of S-Elements = 300, Range = 2) | 60 |
| 5.30 | Time vs. Variance of Noise for Robots without <i>S-Net</i> | 61 |
| 5.31 | Distance Traveled vs. Variance of Noise for Robots With <i>S-Net</i> (Number of S-Elements = 300, Range = 2) | 61 |
| 5.32 | Distance Traveled vs. Variance of Noise for Robots Without <i>S-Net</i> | 62 |
| 5.33 | Final Distance to Source vs. Variance of Noise for Robots With <i>S-Net</i> (Number of S-Elements = 300, Range = 2) | 62 |

| | | |
|------|--|----|
| 5.34 | Final Distance to Source vs. Variance of Noise for Robots Without <i>S-Net</i> | 63 |
| 5.35 | Trace of robots going back and forth with <i>S-Net</i> (2 robots, 2 round trips) | 63 |
| 5.36 | Trace of robots going back and forth without <i>S-Net</i> (2 robots, 2 round trips) | 65 |
| 5.37 | Time used by up to 10 robots for up to 10 round trips with <i>S-net</i> (number of s-elements = 500) | 65 |
| 5.38 | Distance traveled by up to 10 robots for up to 10 round trips with <i>S-net</i> (number of s-elements = 500) | 66 |
| 5.39 | Time used by up to 10 robots for up to 10 round trips without <i>S-Net</i> | 66 |
| 5.40 | Distance used by up to 10 robots for up to 10 round trips without <i>S-Net</i> | 68 |
| 5.41 | Trace of robots going back and forth with <i>S-Net</i> (2 robots, 2 round trips, noise = 10) | 68 |
| 5.42 | Trace of robots going back and forth without <i>S-Net</i> (2 robots, 2 round trips, noise = 0.5) | 72 |
| 5.43 | System cost comparison vs. coefficient in quadratic distribution | 72 |
| 5.44 | System cost comparison vs. coefficient in linear distribution | 73 |

CHAPTER 1

INTRODUCTION

1.1 Thesis Statement

Thesis: *The exploitation of nonmobile, distributed sensor and communication devices by a team of mobile robots offers performance advantages in terms of speed, energy, robustness and communication requirements.*

At one extreme, mobile robots can be provided with a wealth of on-board sensing, communication and computational resources [1, 7]; at the other extreme, robots with fewer on-board resources can perform their tasks in the context of a large number of stationary devices distributed throughout the task environment [3]. We call the latter approach *Smart Sensor Networks*, or S-Nets.

In this study, all the models are simulated using software (C and Matlab), and the performance of robot tasks with and without the presence of an *S-Net* (i.e., a set of distributed sensor devices) is evaluated in terms of various measures.

1.2 Example Scenarios

The notions described above can be exploited in many situations and across several scales of application. Let us consider the following three: (1) fire fighting robots, (2) reservoir monitoring agents, and (3) wearable devices.

- Fire fighting:

Suppose mobile robots are used to fight forest fires; then, there may be several hot spots to extinguish or attempt to control. If sensor devices can be distributed in the environment, then their values and gradients can be used to direct the behavior of fire fighting robots. Such mobile robots used as fire

fighters can have several behaviors. They can transport fire extinguishing materials from a depot to the closest fire source and attempt to put out the fire. During this movement to and from the fire, collision avoidance algorithms can be employed. Sometimes coordinated activities are necessary and communication models are also important. Such a fire fighting behavior will continue until the current fire source is under control. Then the robots will move to the next serious source according to sensed temperature gradients.

- Reservoir task:

When swimming robots are used in a reservoir, some chemical sources can be detected and handled. If chemical concentration sensor devices can be distributed in the environment, their values and gradients can be used to direct the behavior of swimming robots. Such mobile robots can have several behaviors. They can transport neutralizer to the closest source, or they can block up the leaking source. During this process, coordinated activities and communication are necessary. The process will continue until the concentration in the whole reservoir is within a specified limit.

- Wearable devices:

Networked devices embedded in clothing or the external surface of a vehicle may be used to sense the environment and to automatically change the coloration of the clothing or vehicle to better suit a given task. For example, this could be used to blend into the background (i.e., camouflage), or to stand out from the environment (i.e., for rescue). The *S-Net* we are studying allow such capabilities.

1.3 Study Design

Chapter 2 provides models for various components of study: (1) mobile robots with on-board sensors (2) communication, (3) the *S-Net* (includes computation, sensing and communication), and (4) the simulation environment.

Chapter 3 describes algorithms developed for the *S-Net* which perform cooper-

ative computations and provide global information about the environment. Local and global frames are defined and algorithms for their creation are given. A method for the production of global patterns using reaction-diffusion equations is described and its relation to multi-robot cooperation demonstrated. Finally, we show how the level set method can be used within the *S-Net* framework to provide mobile robot path planning.

Chapter 4 describes mobile robot behaviors which exploit *S-Nets* to achieve goals such as those given in the scenarios above.

Chapter 5 provides the results of a set of experiments designed to help us better understand the benefits and drawbacks of *S-Nets*. For behaviors of one mobile robot going to a temperature source, and multiple mobile robots surround a temperature source, in ideal situation, which means no noise, the *S-Net* takes more time and distance. But when noises are added in, which is more realistic, the *S-Net* system will be more robust than the *non-S-Net* system. For the last behavior of multiple mobile robot going back and forth to a temperature source, there are thresholds above which the *S-Net* system out-performs the *non-S-Net* system.

Finally, conclusions and future work are given in Chapter 6.

CHAPTER 2

MODEL DEVELOPMENT

Several models are required in order to explore the questions that have been posed. These include a mobile robot model, sensor models, an *S-Net* model, a communication model, and a model of the environment. The simulation provides a computational framework for the interaction of these models in terms of mobile robots performing useful tasks in the environment, and we define our simulation model as well.

2.1 Mobile Robot Model

In order to act, a robot must receive current environmental information and calculate its movement based on the information received. On-board sensors (e.g., temperature, range, etc.) provide information about the environment and inform the robot's behaviors. In addition, the mobile robot may be able to communicate with other robots or the *S-Net*. The robot achieves movement by rotating or translating based on turning and motion primitives with given rotational and linear speeds.

A *Mobile Robot Model* is defined by:

- Local Frame:

2D or 3D frame attached to the robot that provides the relative location of objects with respect to the mobile robot. An example of robot local frame is displayed in Figure 2.1.

- Position Estimate:

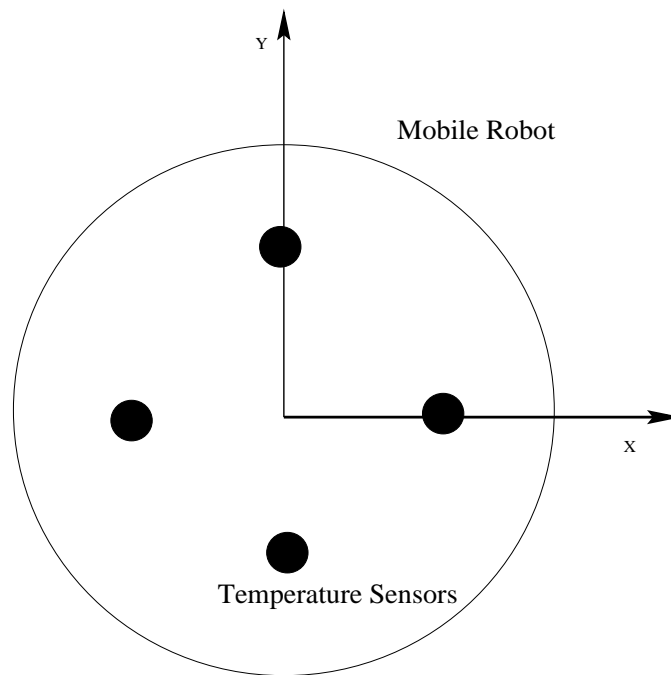


Figure 2.1. Local Frame

2D or 3D location estimate in world frame coordinates. It is used to control the robot's behavior.

- **Heading Estimate:**

Orientation estimate in world frame coordinates. It is used to control the robot's behavior.

These are the robot's estimates, and may be different from the actual values in the environment model; this is caused by various sensor, actuation and computation errors.

- **Description of On-board Sensors:**

A simple distribution of on-board sensors is displayed in Figure 2.1. Four on-board temperature sensors are located on the axes of the local frame, and at a certain distance from the center of mobile robot. In the current implementation, the on-board sensors can be temperature sensors or range sensors.

- Primitive Behaviors:

Primitive motion functions available to the robot (e.g., turn, go forward, go backward, stop turn, stop go).

- Turn: the robot can set its rotational speed as the maximum rotating speed in either the clockwise or counterclockwise directions.
- Go Forward and Go Backward: the mobile robot can set the linear motion speed as maximum linear speed in either the positive or negative X direction of its local frame.
- Stop: there are commands to stop rotation and stop linear motion.

- High-level Behaviors:

The high-level behavior of the robot is specified by a program which maps the robot state and environmental information to primitive behavior sequences. For example, the behavior for the mobile robot to go to the closest temperature source will include: mobile robot sensing to get environmental temperature and gradients, turning as well as going forward to the source, and finally stopping when it reaches a certain distance from the temperature source.

The *Environment Model* consists of:

- World Frame:

Base frame for world objects

- Actual Robot Location:

This is the actual 2D or 3D position of robot in the world frame.

- Obstacles:

Location and shape of obstacles in the world frame. The central location of obstacles may be generated randomly by using a random number generator.

The basic shape of obstacles will be square blocks with certain length edge. If two or more obstacles overlap, various shapes will form.

- Sources:

Functions describing sources and distribution of energy, material, etc. (e.g., heat, chemicals, etc.) Multiple sources may be defined, the number and position of sources can be decided by the user. The formula for distribution of temperature is:

$$T(x, y) = \frac{C}{\sqrt{(x - x_s)^2 + (y - y_s)^2 + 1}} \quad (2.1)$$

The temperature in a given location will be the maximum of all the temperature sources.

[Note: the robot can only estimate the actual environment variables' values by using its sensors.]

The *Sensor Model* is given by a specific model for each modality; this includes noise effects and others for each type of sensor. Here we specify models for the sensors used in our simulation.

- Temperature Sensor Model:

Ideally the sensors do not have any noise, but in practice, sensor data is corrupted by noise, e.g., Gaussian noise. A simple model for a temperature sensor is:

$$\hat{T}(x, y) = T(x, y) + N(\mu, \sigma) \quad (2.2)$$

where $T(x, y)$ is the actual temperature at location (x, y) in the environment and $N(\mu, \sigma)$ is a normal distribution function with mean μ and variance σ . The sensor response may also be affected by nonlinear effects such as hysteresis or failure modes.

- Range Sensor Model (generic):

A range sensor (e.g., sonar) can detect objects within a certain distance. Range sensor models depend on the sensor device geometry and physics as well as the structure of sensed surfaces, here we only give the generic form of expression:

$$\hat{R}(x, y, \theta) = R(x, y, \theta) + N(\mu, \sigma) \quad (2.3)$$

where $R(x, y, \theta)$ is the actual range of the nearest object from location (x, y) in direction θ , and $N(\mu, \sigma)$ is Gaussian noise.

2.2 Communication Model

The *communication model* consists of a protocol, message layout, error model and performance characteristics.

The protocol specifies the meaning of the bits in a message, as well as a set of commands for communication between robots and *S-net* elements (s-elements). A group of s-elements sharing a common frame is called s-clique.

- Protocol:

Commands for a robot to communicate with s-elements:

| <i>value</i> | <i>command</i> | <i>meaning</i> |
|--------------|------------------|--|
| X0000000 | reset | reset all the s-elements |
| X0000001 | is anybody there | robot requesting all s-elements in range to respond |
| X0000010 | talk to origin | robot communicates with origin of local frame to get information of s-elements in local frame |
| X0000011 | talk to one | robot communicates with s-elements separately |
| X0000100 | local position | robot communicates with origin of local frame, provides its position in local frame and asks origin for gradient |

Commands for s-elements to communicate with each other (suppose s-element1 sends to s-element2):

| <i>value</i> | <i>meaning</i> |
|--------------|---|
| X1000000 | reset all s-elements |
| X1000001 | s-element1 requesting all s-elements in range to respond (so distance can be calculated) |
| X1000010 | s-element1 responding to s-element2 range request |
| X1000011 | s-element1 asserting it is origin in s-clique |
| X1000100 | origin provides the robot gradient in local frame |

In the simulation, the message sent out by robots or each s-element are renewed every hundredth of second. The highest order bit (shown by the “X” in the command value is a bit used to specify whether it’s a new message or not, i.e., if a robot or a s-element sends out a new message, the “X” bit in this agent’s message is set to 1, otherwise it keeps as 0.

Robots and *S-Net* elements use the message structures described above in order to achieve various goals. For example, to determine a local frame, the following sequence is used:

- Form local frame and find origin:
 - * For each s-element, send command X1000001 to form s-clique
 - * After s-element receives other s-elements’ command to form s-clique, it responds to the ones within range
 - * After finding the origins of local frames, they send out command X1000011 to assert that they are origin of s-clique
 - * Next, the origin determines frame
- Message layout indicates the structure of each message sent out by either a mobile robot or a s-element. In the first byte of the message, there is a bit to indicate the new message, and a bit to indicate whether its a s-element or a robot. The other bits in this byte are reserved for future use. The next two bytes are the IDs for each agent. The bytes after depend on each command.

| bytes\bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|-----------------------------------|---|---|---|---|---|---|---|
| 1 | new s-element message or robot | | | | | | | |
| 2 3 | ID | | | | | | | |
| | command dependent | | | | | | | |

The command dependent part for robot communicates with the s-elements:

– Is Anybody There:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|--------------------|---------------------|
| 4 | empty | |

– Talk To Origin:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|---|---------------------|
| 4 | s-element1 ID | 2 |
| 5 | X position of s-element1 in local frame | 2 |
| 6 | Y position of s-element1 in local frame | 2 |
| 7 | s-element2 ID | 2 |
| 8 | X position of s-element2 in local frame | 2 |
| 9 | Y position of s-element2 in local frame | 2 |

– Talk To One:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|------------------------------------|---------------------|
| 4 | number of agents want to talk with | 1 |
| 5 | agent 1 ID | 2 |
| 6 | agent 2 ID | 2 |
| . | | |
| . | | |
| . | | |

– Local Position:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|------------------------------------|---------------------|
| 4 | s-element ID of origin | 2 |
| 5 | X position of robot in local frame | 2 |
| 6 | Y position of robot in local frame | 2 |

– Is *Home* Around:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|--------------------|---------------------|
| 4 | empty | |

- Talk to Blake Origin:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|---------------------------------|---------------------|
| 4 | ID of s-element want to talk to | |

The command dependent part for robot communicates with robots:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|------------------------------------|---------------------|
| 4 | ID of the first cooperating robot | 2 |
| 5 | ID of the second cooperating robot | 2 |
| 6 | the distance to the first robot | 2 |
| 7 | the distance to the second robot | 2 |

The command dependent part for s-elements communicate with each other or with mobile robots:

- Command 0000000:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|--------------------|---------------------|
| 4 | empty | |

- Command 0000001:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|--------------------|---------------------|
| 4 | empty | |

- Command 0000010:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|-----------------------------|---------------------|
| 4 | agent ID of which asked | 2 |
| 5 | distance to the agent asked | 2 |

- Command 0000011:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|--------------------|---------------------|
| 4 | empty | |

- Command 0000011:

| <i>field No.</i> | <i>description</i> | <i>No. of bytes</i> |
|------------------|--|---------------------|
| 4 | robot ID | 2 |
| 5 | gradient in robot's position (local frame) | 4 |

- Error:

The messages will be exchanged as packages. Messages can be lost during the process of communication; the effect of this error is that some agents do not

receive the message. Various methods may be used to recover from this error (e.g., wait one communication period and let the source send the message one more time). Another kind of error is a wrong message, e.g., bad bits may be sent. Parity checking can be used to detect and correct this. For this study we explore only such simple kinds of errors and recovery.

- Performance:

The performance characteristics which can be set include:

- bandwidth: bits per second transmission
- range: maximum distance from device that signals can be received.

2.3 Smart Sensor Network Model

S-Net devices consist of three essential components: computation, sensing and communication. The *computation element* is described by the speed of the processor, its storage capacity, power requirements and cost. Sensors used by the *S-Net* devices are modeled as described above, but also include bandwidth, latency, power requirements and cost. The *communication model* is like that given for mobile robots, but includes power requirement and cost as well.

2.4 Simulation Model

We use discrete event simulation with a fixed time step. In that we must model and simulate continuous events (e.g., during robot motion) as well as discrete events, we allow for an *every-time-step* event which can be put at the head of the event queue and must be handled every time step. Any number of these may be added to the event queue.

The event list is a table recording all events that will happen in each time step. At the beginning of each time step, we copy it to a temporary list, and new events will be generated and added to the event list during the movement of the robots. The table has three fields:

- *Agent ID*: all the mobile robots and distributed sensors are agents with unique IDs. Robot IDs are distinguishable from s-element IDs.
- *Event type code*: indicates the type of behavior, including:
 - RunRobots: executes the robots' behaviors; this transforms the high-level behavior into a sequence of primitive behaviors such as: turn, go forward, go backward.
 - RunS-Elements: executes the s-elements' behaviors; this includes communication between sensors and robot, forming local frames, and running local computations.
 - Turn, Go Forward, Go Backward: primitive behaviors for mobile robots.
 - Stop Turn: will stop the rotation of the robot.
 - Stop Go: will stop the linear motion of robot.
 - Broadcast: produces communication events.
- *Time code*: time of event execution. To handle continuous movements of mobile robots, some basic behaviors such as rotating and going will happen all the time, so we set up a special code which means to execute every time step. Other behaviors such as StopGo and StopTurn, are discrete and occur at the scheduled time.

During each time step, all scheduled events are handled and new events will be generated and added to the event list according to the different robot behaviors. At the end of each time step, the resulting state is evaluated to determine its feasibility. If an impossible state has occurred (e.g., a robot penetrates an obstacle), then a special handler is called to resolve the problem. Once a possible state is achieved, the status of each robot such as position and local direction is updated. This procedure is repeated until the simulation terminates.

CHAPTER 3

ALGORITHMS FOR SMART SENSOR NETWORKS

A *Smart Sensor Network* (or *S-Net*) is a collection of individual devices (called *s-elements*) which have sensors, communication and computation and are distributed either in a specified pattern or randomly to provide environmental information. The *S-Net* provides the framework for information gathering, analysis and presentation — it is an “information field” for the mobile robot. We propose three major algorithmic infrastructures:

- *Coordinate frames*: both local and global frames can be determined and exploited for robot tasks.
- *Pattern formation*: global patterns in the *S-Net* can be calculated and used to provide information for the robots.
- *Level sets*: the ability to model and compute moving boundaries in the S-Net adds significant capabilities for robot tasks, including constrained shortest path information.

These three methods are developed and shown useful for robot behavior specification.

3.1 *CoordinateFrameDetermination*¹

Groups of s-elements form a *s-clique*, which share a local coordinate frame. The information in one frame can be transferred to others based on coordinate

¹This section is an improved version of [3] and built on the work of Mohamed Dekhil.

transforms; i.e., a position in one frame can be converted into a position in another frame provided the s -cliques share enough s -elements. This can then be exploited to specify application-specific spatial patterns in the sensor set, to determine local or global coordinate frames, and to use those to support mobile robot behaviors for robots that communicate with the s -elements. For example, mobile robots can move along the concentration gradient or temperature gradient to find the source of the sensed quantity.

At least three points, which are not on a straight line, are needed to establish a global frame for a set of n points ($n \geq 3$). According to the absolute coordinates of three points and the distances between them, we can decide the absolute coordinates of any other point.

A local frame can be viewed as a special case of the global frame, where the position and orientation of the frame can be chosen arbitrarily [3]. Clusters of small simple sensors can be used by a mobile robot to identify and locate certain events in the environment. Sometimes it is sufficient to use the transformation of local frames to navigate the mobile robot.

To utilize the sensor population for navigation and localization purposes, local and/or global frames of reference need to be established. The assumption here is that the only information each sensor knows about the other sensors is their distance from it. We assume that sensors can have uniquely identifiable tagged chirps, and can broadcast a signal that other sensors hear and relay back after some delay. Thus, a set of distances can be determined and from these an algorithm can compute positions with respect to a local sensor frame. The following is a formulation of this problem and its solution.

Problem: Given a set of n points, where $n \geq 3$, and the distances between each pair of points, $\{d_{ij}\}$, where $i = 1..n, j = 1..n$, and $i \neq j$, it is required to establish a planar frame of reference F where point i can be represented by the coordinates (x_i, y_i) , where $i = 1..n$.

First we construct a global frame for the set of points and then show how to construct a local frame in which the frame origin is selected either arbitrarily or

relative to a certain event or landmark.

3.1.1 Global Frame Calculation

In this formulation we show that the global (absolute) coordinates of at least three points, p_1 , p_2 , and p_3 , not laying on a straight line are sufficient to establish a global frame for a set of n points ($n \geq 3$).

Assume that the global coordinates of the first three points are: (x_1, y_1) , (x_2, y_2) , and (x_3, y_3) , respectively, and the distances between these points and point p_i are d_{i1} , d_{i2} , and d_{i3} , respectively, where $i = 4..n$ (see Figure 3.1). The distance can be expressed in terms of the unknown coordinates of point i , (x_i, y_i) , as follows:

$$(x_i - x_1)^2 + (y_i - y_1)^2 = d_{i1}^2 \quad (3.1)$$

$$(x_i - x_2)^2 + (y_i - y_2)^2 = d_{i2}^2 \quad (3.2)$$

$$(x_i - x_3)^2 + (y_i - y_3)^2 = d_{i3}^2 \quad (3.3)$$

By subtracting Equation 3.2 from Equation 3.1 we get:

$$-2x_1x_i + x_1^2 + 2x_2x_i - x_2^2 - 2y_1y_i + y_1^2 + 2y_2y_i - y_2^2 = d_{i1}^2 - d_{i2}^2$$

and subtracting Equation 3.3 from Equation 3.1 yields:

$$-2x_1x_i + x_1^2 + 2x_3x_i - x_3^2 - 2y_1y_i + y_1^2 + 2y_3y_i - y_3^2 = d_{i1}^2 - d_{i3}^2$$

Simplifying the last two equations yields:

$$(x_2 - x_1)x_i + (y_2 - y_1)y_i = C_1 \quad (3.4)$$

and

$$(x_3 - x_1)x_i + (y_3 - y_1)y_i = C_2 \quad (3.5)$$

where

$$C_1 = \frac{1}{2}(d_{i1}^2 - d_{i2}^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2)$$

and

$$C_2 = \frac{1}{2}(d_{i1}^2 - d_{i3}^2 - x_1^2 + x_3^2 - y_1^2 + y_3^2)$$

From Equation 3.4 we get:

$$x_i = \frac{C_1 - (y_2 - y_1)y_i}{(x_2 - x_1)} \quad (3.6)$$

Substituting 3.6 into 3.5 we can calculate y_i as:

$$y_i = \frac{C_2 - \frac{(x_3 - x_1)}{(x_2 - x_1)}C_1}{(y_3 - y_1) - \frac{(x_3 - x_1)}{(x_2 - x_1)}(y_2 - y_1)} \quad (3.7)$$

Simplifying this equation yields:

$$y_i = \frac{(x_1 - x_3)C_1 + (x_2 - x_1)C_2}{y_1(x_3 - x_2) + y_2(x_1 - x_3) + y_3(x_2 - x_1)} \quad (3.8)$$

Substituting for y_i in Equation 3.6 we get:

$$x_i = \frac{C_1 - (y_2 - y_1) \frac{(x_1 - x_3)C_1 + (x_2 - x_1)C_2}{y_1(x_3 - x_2) + y_2(x_1 - x_3) + y_3(x_2 - x_1)}}{(x_2 - x_1)}$$

Simplifying this last equation we get the solution for x_i as:

$$x_i = \frac{(y_3 - y_1)C_1 + (y_1 - y_2)C_2}{y_1(x_3 - x_2) + y_2(x_1 - x_3) + y_3(x_2 - x_1)} \quad (3.9)$$

This formulation shows that the absolute position (x, y) of three points not forming a straight line are sufficient to construct a frame and calculate the location of the other points with respect to that frame. The condition that the three points should not be on a straight line is necessary, otherwise, the denominator in Equations 3.8 and 3.9 will be zero.

3.1.2 Constructing a Local Frame

The main goal of the *S-Net* paradigm is to provide sensory information within a wide spatial area where clusters of small simple sensors are used to identify and locate certain events or actions in the environment. In many cases it is sufficient to use a local (or relative) frame to navigate and locate local events within the cluster range.

A local frame can be viewed as a special case of the global frame, where the position and orientation of the frame can be chosen arbitrarily. We now show how to construct a local frame for a set of n points, where $n \geq 3$.

1. Select any three points p_1 , p_2 , and p_3 not laying on a straight line. Assume that the distances between these points are d_{12} , d_{13} , and d_{23} . The condition for non linearity is that the distance between any two points should be less than the sum of the distances between these two points and the third point:

$$d_{ij} < d_{ik} + d_{jk}$$

2. Set p_1 as the frame origin (i.e., $p_1 = (0, 0)$).
3. Form the x-axis of the local frame as p_1p_2 , where p_2 is constrained by a circle centered at p_1 with radius d_{12} . This means that $p_2 = (d_{12}, 0)$.
4. Calculate the location of the third point by selecting one of the two intersection points of two circles centered at p_1 and p_2 with radii d_{13} and d_{23} , respectively (see Figure 3.2). To calculate the location of p_3 we solve the following two equations:

$$(x_3 - x_1)^2 + (y_3 - y_1)^2 = d_{13}^2$$

$$(x_3 - x_2)^2 + (y_3 - y_2)^2 = d_{23}^2$$

Substituting the values of x_1 , y_1 , x_2 , and y_2 from steps 2 and 3 above we get:

$$x_3^2 + y_3^2 = d_{13}^2$$

$$(x_3 - d_{12})^2 + y_3^2 = d_{23}^2$$

By solving these two equations for x_3 and y_3 we get:

$$x_3 = \frac{d_{12}^2 + d_{13}^2 - d_{23}^2}{2d_{12}} \quad (3.10)$$

$$y_3 = \pm \sqrt{d_{13}^2 - \frac{d_{12}^2 + d_{13}^2 - d_{23}^2}{2d_{12}}^2} \quad (3.11)$$

We then select one of the two locations for p_3 . Here we select the positive value of y_3 .

5. Use the results from the global frame calculation to get the (x, y) locations of the remaining points with respect to this local frame.

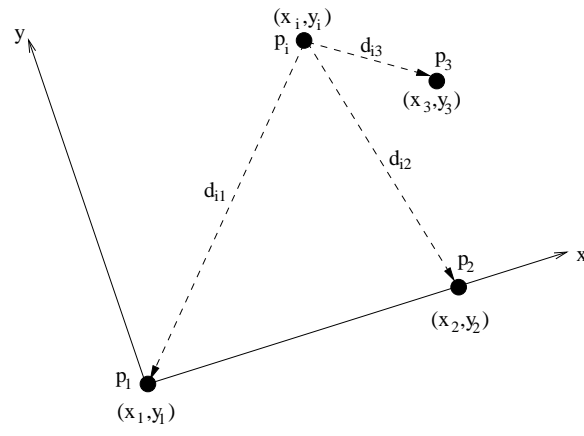


Figure 3.1. The formulation of the global frame.

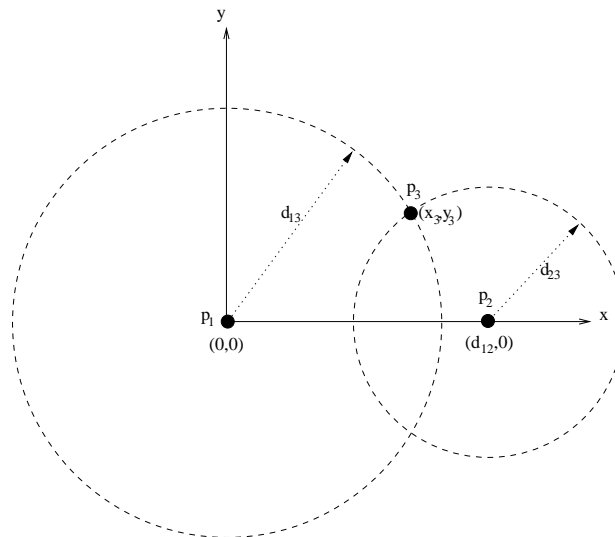


Figure 3.2. Construction of a local frame.

We assume that all local frames are right hand frames. A problem with this is that the actual location of \bar{P}_3 may be such that a left-handed frame is formed. To avoid this may require exploration by a mobile robot. For example, suppose the mobile robot has aligned its X-axis with the X-axis of the local frame; it can then turn $\pi/2$ counter-clockwise, and move along that direction for certain distance. In the new location, it can communicate with \bar{P}_3 , and if the distance between itself and \bar{P}_3 has increased, then the local frame is left-handed. A right-handed frame is formed by using the negative value of y_3 .

3.1.3 Moving between Local Frames

Moving from one sensor cluster to another requires a transformation between the two local frames. To determine the transformation functions, there must be at least two points common to both frames. Using a homogeneous transformation, the relationship between the (x, y) locations of a point p_i with respect to both frames can be written as (see Figure 3.3):

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & d_x \\ \sin \theta & \cos \theta & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \quad (3.12)$$

Thus, we have two equations in three unknowns; d_x , d_y , and θ .

$$x_i = x'_i \cos \theta - y'_i \sin \theta + d_x \quad (3.13)$$

$$y_i = x'_i \sin \theta + y'_i \cos \theta + d_y \quad (3.14)$$

Therefore, we need at least two common points, p_1 and p_2 , to solve Equations 3.13 and 3.14.

Assume that we have two common points p_1 and p_2 with coordinates (x_1, y_1) and (x_2, y_2) with respect to the first frame, and (x'_1, y'_1) and (x'_2, y'_2) with respect to the second frame. Substituting these points in Equations 3.13 and 3.14 yields:

$$x_1 = x'_1 \cos \theta - y'_1 \sin \theta + d_x \quad (3.15)$$

$$y_1 = x'_1 \sin \theta + y'_1 \cos \theta + d_y \quad (3.16)$$

$$x_2 = x'_2 \cos \theta - y'_2 \sin \theta + d_x \quad (3.17)$$

$$y_2 = x'_2 \sin \theta + y'_2 \cos \theta + d_y \quad (3.18)$$

By subtracting Equation 3.17 from Equation 3.15 and Equation 3.18 from Equation 3.16 we get:

$$(x_1 - x_2) = (x'_1 - x'_2) \cos \theta - (y'_1 - y'_2) \sin \theta \quad (3.19)$$

$$(y_1 - y_2) = (x'_1 - x'_2) \sin \theta + (y'_1 - y'_2) \cos \theta \quad (3.20)$$

Solving the last two equations results in:

$$\theta = \text{atan2}(A, B) \quad (3.21)$$

where

$$A = (x'_1 - x'_2)(y_1 - y_2) - (y'_1 - y'_2)(x_1 - x_2)$$

and

$$B = (x'_1 - x'_2)(x_1 - x_2) + (y'_1 - y'_2)(y_1 - y_2)$$

Substitute for θ in Equations 3.15 and 3.16 to solve for d_x and d_y as follows:

$$d_x = x_1 - x'_1 \cos \theta + y'_1 \sin \theta \quad (3.22)$$

$$d_y = y_1 - x'_1 \sin \theta - y'_1 \cos \theta \quad (3.23)$$

Finally, we use Equation 3.12 to transform the rest of the points between the two frames.

When a mobile robot needs to move relative to a s-clique frame, it need to transform the coordinates in the s-clique frame into its own frame to perform the movement. In order to get two common points in these two frames, the mobile robot must:

- 1 Communicate with the three s-elements that form the s-clique frame.
- 2 After obtaining the distances between the mobile robot and the three s-elements, it can compute its location in the s-clique frame; this is the first common point.

- 3 Move along its own X axis for a certain distance, and communicate with s-clique s-elements again.
- 4 Repeat step 2 and get second common point.
- 5 Transform information such as location, gradient from the s-clique frame to its own frame according to the formulas given above.

3.2 Reaction – Diffusion Patterns²

When given a rather dense set of distributed sensors, it is sometimes useful to assign a pattern to them automatically and by means of a distributed algorithm. For example, given a 200 by 200 grid of sensors, it might be more useful to group them into larger units (like the sectors shown in Figure 3.4). What is interesting is that this very sharp box pattern appears globally after running a specific reaction-diffusion equation locally in each s-element for a given amount of time. If a smaller set of randomly distributed sensors is used instead, the patterns shown in Figures 3.5 and 3.6 are typical of the set of sensors with bits set to 1 and 0, respectively (we have used 200 sensors here out of the 200 by 200 grid). Alternatively, striped patterns (see Figure 3.7) might be useful in another situation. Less regular figures may be useful in other cases.

We use Turing’s reaction-diffusion mechanism[9] to generate such patterns in the *S-Net*. The basis of this mechanism is a set of equations that captures the reaction and diffusion aspects of certain chemical kinetics:

$$\frac{\partial \mathbf{c}}{\partial t} = f(\mathbf{c}) + D\nabla^2 \mathbf{c} \quad (3.24)$$

where $f(\mathbf{c})$ describes the reaction and $D\nabla^2 \mathbf{c}$ expresses the diffusion component. The simplest such systems have two *morphogens* or variables; one of these acts as the activator and the other acts as the inhibitor. The two variable system can be modeled by:

$$\frac{\partial u}{\partial t} = \gamma f(u, v) + \nabla^2 u, \quad \frac{\partial v}{\partial t} = \gamma g(u, v) + d\nabla^2 v \quad (3.25)$$

²This section is an improved version of [3] and built on the work of Scott Morris.

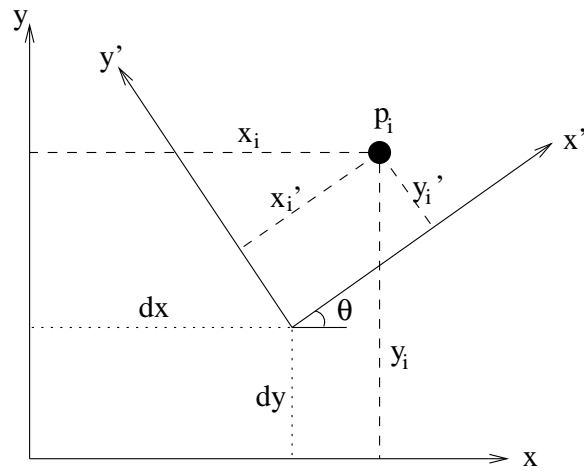


Figure 3.3. Transformation between frames.

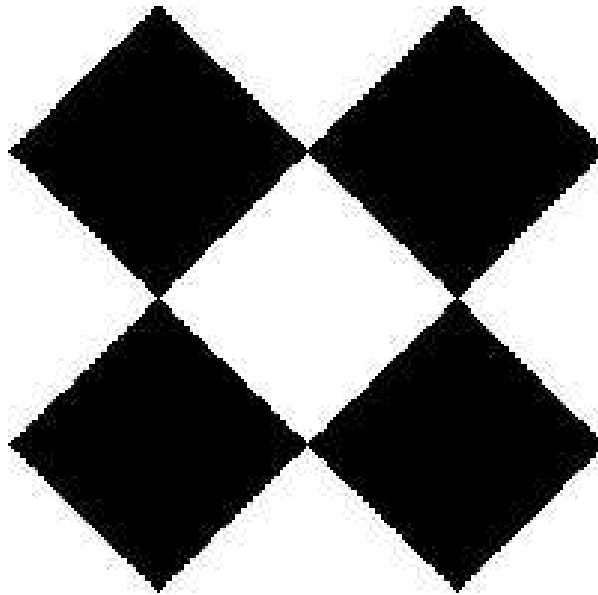


Figure 3.4. Box Pattern

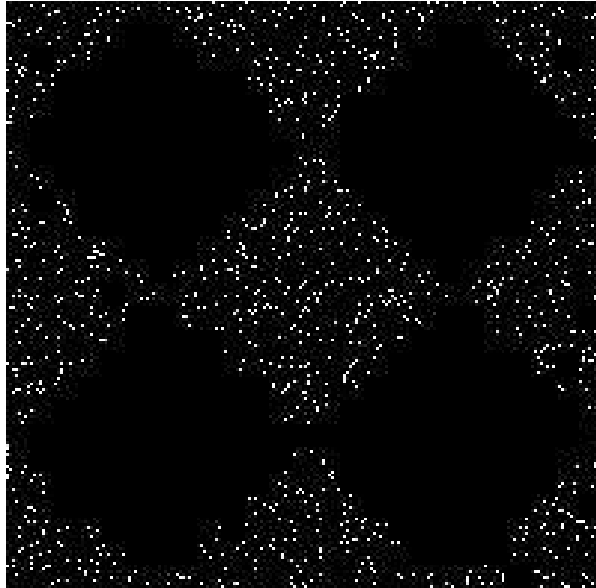


Figure 3.5. Result (bit=1) of distributed reaction diffusion algorithm with random distribution s-elements

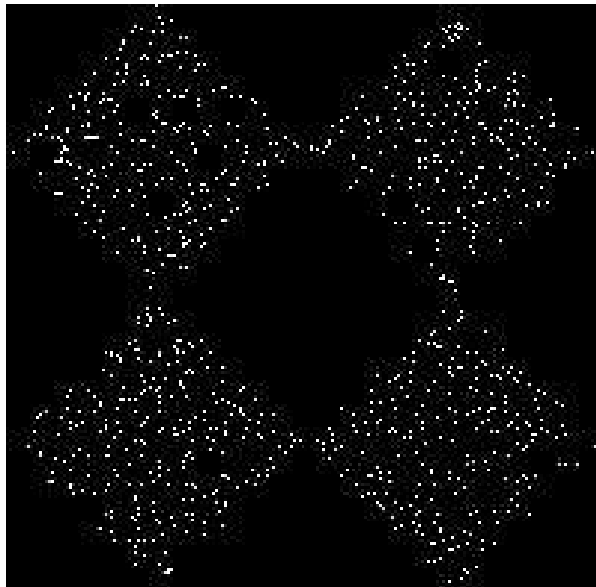


Figure 3.6. Result (bit=0) of distributed reaction-diffusion algorithm with random distribution of s-elements

where u and v are the concentrations of the morphogens, d is the diffusion coefficient and γ is a constant measure of scale. The functions $f(u, v)$ and $g(u, v)$ represent the reaction kinetics. As an example, we have explored the generation of spatial patterns using the Thomas system of equations (Murray[5] describes this system in detail):

$$\begin{aligned} f(u, v) &= a - u - h(u, v), g(u, v) = \alpha(b - v) - h(u, v) \\ h(u, v) &= \frac{\rho uv}{1 + u + Ku^2} \end{aligned} \quad (3.26)$$

where a , b , α , ρ , and K are the positive reaction parameters. They define a domain in which Equations 3.26 become linearly unstable to certain spatial disturbances. This domain is referred to as *Turing space* where the concentrations of the two morphogens will become unstable and result in the patterns shown in Figure 3.8.

The pattern is the result of each s-element running the equations locally while *diffusing* to its neighbors; a stable solution is thresholded to produce a binary value at each sensor, and the total of these gives the pattern in the sensor. [Note: in the example given here, the spatial solution is known analytically and was computed directly.]

For example, the pattern shown in Figure 3.4 makes a very regular framework within which a robot can move, while Figure 3.7 provides a set of fixed distances from some radially aligned point. Such a pattern can be used as an incremental encoder for the determination of distance traveled, or as a mechanism for robot rendezvous.

Generated patterns may also be much less regular (see Figure 3.8) while still having statistically useful properties; e.g., a certain percentage of area covered with *on* bits. The notion is that a mobile robot traveling near a given sensor can determine whether the sensor is on or off (in the final spatial pattern) and use that information to achieve its desired behavior; e.g., follow a stripe in Figure 3.8. Multiple patterns may exist simultaneously for different goals, and any desired pattern can be represented by a combination of some basis set of 2D patterns (e.g., Haar basis functions).

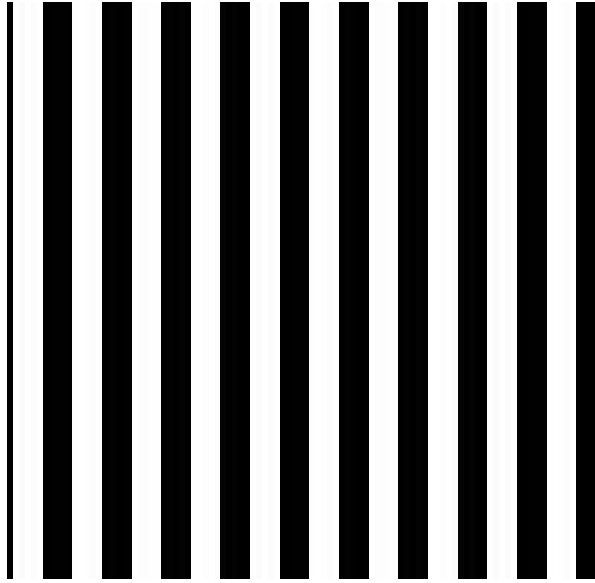


Figure 3.7. Stripe Pattern

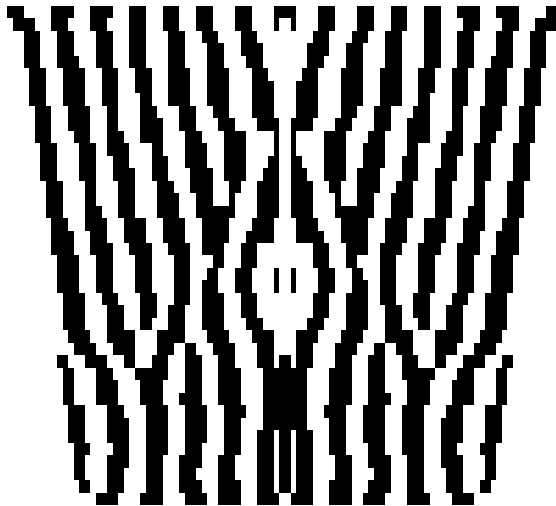


Figure 3.8. Irregular Pattern

3.3 Level Set Method

Level set methods are newly developed techniques for analyzing and computing interface motions. Numerical techniques are used to track complex fronts as they evolve [6].

Suppose there is a closed hyper-surface $\Gamma(t=0)$, propagating along its normal direction with speed F , which is a function of curvature, normal direction, etc. The main idea of the level set methodology is to embed this propagating interface as the zero level set of a higher dimensional function ϕ , defined as $\phi(x,t=0)$. x is a point in \mathbb{R}^n . The equation for the evolving function $\phi(x,t)$ that contains the embedded motion of $\Gamma(t)$ as the zero level set (in the Eulerian formulation) is:

$$\phi_t + F|\nabla\phi| = 0 \quad (3.27)$$

This formula transforms a geometry problem into an initial value partial differential equation, so that numerical techniques can be used to solve the problems of a surface moving in two or three dimensions.

3.3.1 Path Planning

Our focus in this project is on constrained mobile robot navigation; this means that we would like to determine optimal paths from the mobile robot to a target location with the following conditions:

- the computation should be distributed in the s-elements
- the computation should be able to incorporate constraints (e.g., in the form of a speed function tied to the nature of the terrain)
- the computation should be robust.

It is possible to achieve this by using the level set technique proposed by Sethian (p. 181):

As a simple example, imagine a starting point A and a finishing point B in the plane, and a collection of subsets of the plane which

represent obstacles: $\Omega_j, j = 1 \dots N$. The shortest path under a speed function $F(x, y)$ from A to B which avoids these obstacles can be found by computing the solution to the Eikonal equation:

$$|\nabla T| = \frac{1}{F(x, y)}$$

where $F(x, y)$ is reset to a very small number ϵ at those (x, y) which belong to one of the subsets. Once the solution is found, back propagation from B to A along the gradient constructs the optimal path.

The function $F(x, y)$ gives the speed (i.e., maximum possible robot speed) at location (x, y) in the terrain. The function $T(x, y)$ gives the time of arrival of the mobile robot were it to take the fastest possible path to (x, y) from its current position.

In general, the function T depends on the mobile robot's location and must be computed. Once T is known, the gradient of T can be computed, and then the optimal path is found by following the opposite of the gradient of T from the target back to the robot. Our insight is that this computation is easily done by the distributed sensor devices. Let's consider an example.

Figure 3.9 shows a function $F(x, y)$ where speed is greater on roads (white regions), lower off-road (gray regions), and one part of the vertically-oriented road is blocked (black region; $F = 0$ in that region). The mobile robot (light gray square) is in the upper part of the map. The mobile robot communicates to the nearest s-element which sets its T value to zero. The Eikonal equation is then solved (although we use a slightly modified version of the fast marching algorithm which we have found faster and more robust), and the function T is determined (see Figure 3.10). In the figure, intensity is proportional to distance; the blocked zone is white and cannot be entered.

Figure 3.11 shows the contour plot of T . The gradient is reversed and followed from the target to the robot; the computed path is shown overlaid on the F function in Figure 3.12. As can be seen, the robot follows the fastest parts of the terrain where possible.

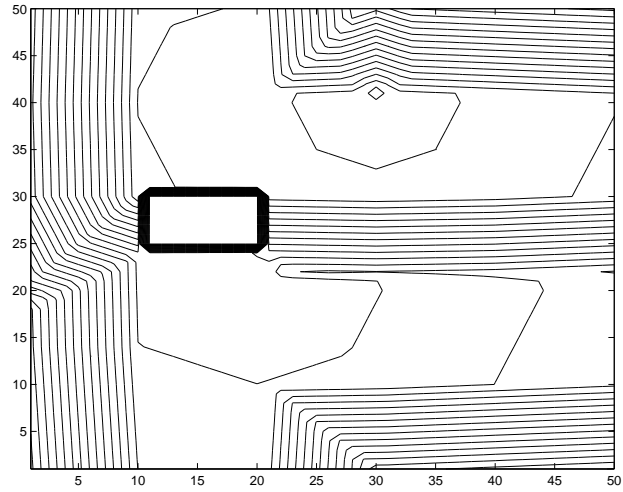


Figure 3.11. Contours of T

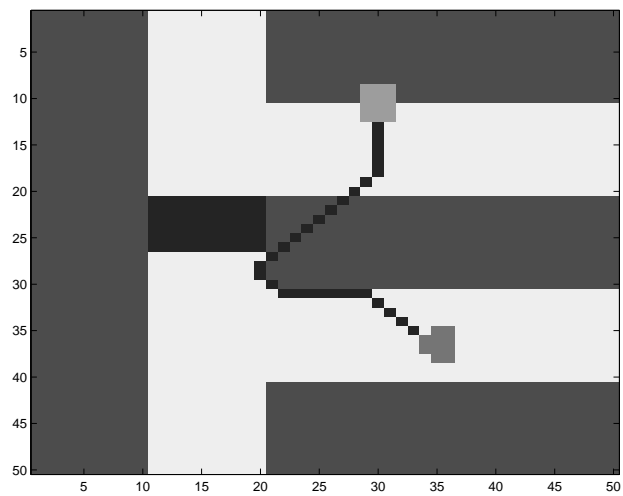


Figure 3.12. Shortest Path

Several issues require further elucidation. First, the distributed Eikonal equation computation is accomplished as follows:

- each node initially is set to a value of infinity
- when all a node's neighbors that are used in the computation of its own value change from infinity to a known value, then the node's computation proceeds
- the node then passes its value to its neighbors which require the value (i.e., we use the *upwind* scheme).

This proceeds until all nodes have a value (this requires time proportional to the diameter of the sensor device grid).

The F values must be set; several possibilities exist:

- registration of local sensor device coordinate frames with a global frame of reference (e.g., a map)
- distributed sensor devices measure the hardness of the local terrain and assign an F value
- as they explore the terrain, the mobile robots set F values in the s-elements.

Our method assumes F is known.

Once T is calculated, ∇T can be computed locally. Reversing the gradient is accomplished by reversing the sign of the gradient. To determine the shortest path, we simply follow the gradient from the target back to the robot. (Our method follows the minimum time of arrival value back to the robot.)

CHAPTER 4

MOBILE ROBOT BEHAVIORS

The mobile robot's moving and turning behaviors are based on information provided by the *S-Net*. As an example, consider the case of temperature sensors. At one extreme, a mobile robot may have four on-board temperature sensors located in different positions, thus providing four different spatial samples. The temperature gradient can be determined from these four values. Equations [4.1] and [4.2] give the temperature function and gradient function, respectively. The temperature gradient can be used to control the heading of the robot (see Figures 4.1 and 4.2).

Figure 4.2 is a simulation of a mobile robot moving in the temperature gradient direction to find the temperature source. The mobile robot starts at the origin and moves to the temperature source (located at (-3,2)), which has the distribution function (4.1). The turning speed is π radians/sec and linear speed is 2 meters/sec. At the other extreme, with the *S-Nets*, the robot will obtain the gradient information from the scattered s-elements.

$$T(x, y) = \frac{C}{\sqrt{(x - x_s)^2 + (y - y_s)^2 + 1}} \quad (4.1)$$

$$\nabla T = \left[\begin{array}{c} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{array} \right] = \left[\begin{array}{c} \frac{-C(x-x_s)}{\sqrt{(x-x_s)^2+(y-y_s)^2} \left(1+\sqrt{(x-x_s)^2+(y-y_s)^2}\right)^2} \\ \frac{-C(y-y_s)}{\sqrt{(x-x_s)^2+(y-y_s)^2} \left(1+\sqrt{(x-x_s)^2+(y-y_s)^2}\right)^2} \end{array} \right] \quad (4.2)$$

When more than one robot is in use and without *S-Nets*, direct cooperation between robots is necessary. A set of algorithms are needed to prevent robots' collision and to improve the operational efficiency (e.g., minimum total distance, etc.). Our goal is to study the various aspects of the relationship between robots

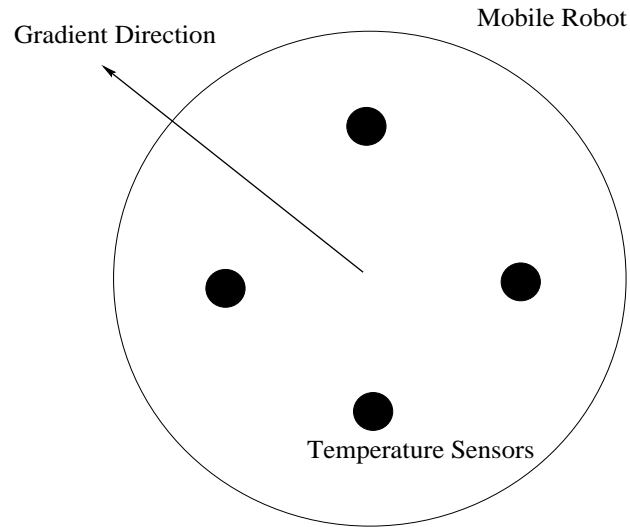


Figure 4.1. Gradient Calculation

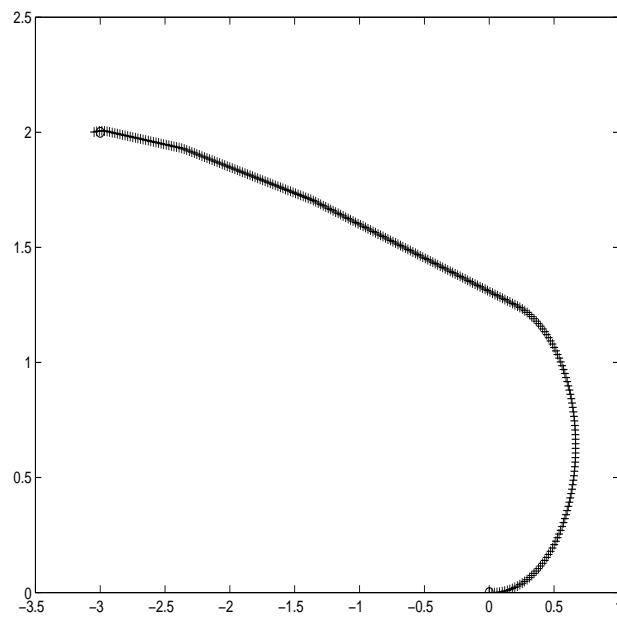


Figure 4.2. Simulation of a Mobile Robot Moving against Temperature Gradient

with and without the *S-Nets*. Variables of interest include: sensor distribution, robot and s-element parameters, and sensor performance.

When using multiple robots and devices, communication between agents plays an important role. The robots should be able to communicate (e.g., current position, speed, etc.) with each other, as well as s-elements and external controllers. A

simple and efficient communication protocol has been described in Section 2.1.

We study robot performance in terms of behaviors developed for various scenarios including both military (e.g., tactical urban settings) and disaster mitigation (e.g., chemical spills, forest fires). Robots can be used to control fire or detect poisonous gas sources; i.e., things that are difficult and dangerous for human beings.

4.1 Goal Achievement

The most important criterion for robot evaluation is the successful achievement of its goals. Assuming that various strategies are all successful, the next level of comparison is achieved in terms of the efficiency of the behavior. A basic set of goals for mobile robots are:

- *Go to geometric destination:* This may involve either absolute or relative locations as well as the ability to maintain relative positions (e.g., follow another robot).
 - *Go to absolute location:* Suppose we know the global position of the mobile robot and the s-elements and all the agents belong to the same global frame, then *go to absolute location* can be implemented directly.
 - *Go to relative location:* In most cases, global frame may not be available. Information of the environment can be obtained only through local frames. The location of destination is relative to the local frame, and the mobile robot needs to exploit the transformation between local frames to reach the final destination.
- *Go to destination related to source:* This involves moving with respect to the sources of interest in the particular problem (e.g., a temperature source). Moreover, this may be based on the known values at the s-elements or interpolated values.

For those behaviors that utilize the *S-Net*, all the information about source comes from s-elements, therefore the mobile robot needs to communicate with

the origins of various s-cliques or even with particular s-elements (e.g., the s-element with highest temperature in some s-clique). To achieve this, the robot:

- Communicates with the origin and other two s-elements in a s-clique frame to decide its current position in that frame.
- After moving a certain distance, the robot again communicates with them to determine the new location.
- Transforms the pertinent information such as the location of the s-element with highest temperature value to its own frame (the robot frame) and moves to that location.
- Repeats the above process until it reaches the s-element with highest value of all s-elements.

In the discussion above, we suppose there are at least two common s-elements between every two local frames. But in reality, disconnected s-cliques may occur, which means some s-cliques may have less than two common s-elements with others. In this case, the mobile robot can only reach the s-element with the highest value in the connected s-clique set. To do better, it is necessary to increase the number of s-elements, or adjust the range of the s-clique broadcast.

For the robot behaviors that do not exploit an *S-Net*, the mobile robot obtains the information about the source (e.g., the temperature gradient) by itself. The mobile robot moves along the gradient towards the source, until the detected value (e.g., temperature) is above some limit.

- *Go along constrained path:* This involves the incorporation of various constraints into the path selection method; for example, the least costly path may be desired (cost may be related to distance, time, energy, etc.) or a path with some constant value in a space of interest (e.g., constant distance from a source), or may involve other desired properties (e.g., avoid collisions).

4.2 Multiple Robot Behaviors

- Multiple mobile robots cooperate and communicate via the *S-Net* to find temperature source and keep certain distance from the temperature source evenly.

When the robots reach a certain distance away from the highest temperature s-element , robots communicate with each other to get the distances. Then two of the mobile robots that are close to each other keep their positions and the farthest one from them compute the positions that be able to form an equilateral triangle, from the two results, the robot choose the close one and move to it.

- Multiple mobile robots go back and forth to the temperature sources, the intensity of temperature source will decrease after each robot's coming, and finally the temperature of whole environment area will be controlled under certain level.

Home is chosen as the origin of the s-clique that sensed the lowest temperature, then stripe patterns are formed along the gradient of the temperature source to *Home*. The straight line of *Home* to temperature source is in the middle of a white stripe, and black stripes alternate spatially with white stripes. The width of each stripe is a constant. The robots will move from *Home* to temperature source along white stripe and follow the black stripe back *Home*. During the procedure, if any robot detects that a collision is about to happen, it will slow down until the collision will not happen.

CHAPTER 5

PERFORMANCE EVALUATION

The whole development of the thesis up to this point has created the framework in which the performance of mobile robots can be compared with respect to using the *S-Net* or not. In this chapter, we compare the performance of mobile robots while solving the following tasks:

- One robot goes to a temperature source.
- Multiple robots surround a temperature source.
- Multiple robots go back and forth to a temperature source.

This set of tasks represents typical mobile robot tasks and can be configured to exploit many of the constraints described in earlier chapters. For example, a robot's path may be required to be the shortest, the gradient may be followed, or patterns in the *S-Net* may be used as road markers. Moreover, the last two tasks provide a setting to use multiple robots, ranging from few to many robots. In addition, robot interactions are necessary, at least as far as avoiding collisions. For each of these tasks, we propose a relevant set of performance measures, as well as a discussion of parameters and their possible values. Finally, we give the performance results and compare the two approaches.

Our goal is, by analyzing the measurement such as time used, distance traveled and the final distance to the temperature source, trying to find out under what conditions, the *S-Net* system can perform better or become more robust and what is the cost. According to the final result, for the first two behaviors, the *S-Net* system does not perform better under ideal situation, which means no noise at all.

But when the noise is added in, it displayed that the *S-Net* system is more robust, especially in rough situation, which means there are lots of noise in the environment. For the third behavior, the *S-net* system not only performs much better under realistic situation, but even under ideal condition, it displays its benefit. For a certain distance of round trip, the *s-Net* system can support more robots on the trace and prevent collisions happen between each other. On the other hand, if there are too many robots on the trace, in the system without the *S-Net*, some robots can not move properly and prevent collision.

Overview of methodology:

- One robot goes to a temperature source
 - The system without the *S-Net*:

The robot use four on-board temperature sensors to detect the temperature and move against temperature gradient toward the temperature source.
 - The *S-Net* system:

The robot communicates with the s-elements around and finds out the closest origin of a s-clique; then the robot communicates with this origin to transform the position of the s-element with highest temperature in the s-clique to its own frame, and move to that position. This procedure repeated until the robot reaches the highest s-element of the entire environment.
- Multiple robots surround a temperature source
 - The system without the *S-Net*:

Three robots begin from different places, and each of them use four on-board temperature sensors to detect the temperature and move against temperature gradient toward the temperature source. While the average temperature detected is above certain value, they stop and try to commu-

nicate with other robots, until all robots have reached. Then on robot, which is closer to the other will maintain its position, and the other robots will move following a constant-valued temperature contour. To do this, they compute the gradient and move perpendicularly to it.

- The *S-Net* system:

Three robots begin from different places, and each of them use local frame transformation move to the s-element with highest temperature, (for details please see description in behavior one). When all robots reach a certain distance away from the highest temperature s-element, two of the robots keep their positions, and the other one which is the farthest from them will move forward to form an equilateral triangle.

- Multiple robots go back and forth to a temperature source

- The system without the *S-Net*:

Each robot move form *Home* using four on-board temperature sensors to detect the temperature and move against temperature gradient toward the temperature source. Then they turned around, using four on-board *Home* sensors to move back to *Home*. And we choose the normal robot behaviors to prevent collision, when robots detect an environment collision, they make a right turn and then try to get back on track again.

- The *S-Net* system:

The stripe patterns are formed along the gradient of the temperature source to *Home*. Each robot begin from the same place and move along the white stripe toward the temperature source and follow the black stripe back *Home*. When a robot detects that a collision is about to happen, it will slow down to prevent the collision, until it can not detect the collision any more.

5.1 One Robot Goes to a Temperature Source

This is the simplest behavior for a robot, and the major focus of this experiment is to analyze the robustness of the system by changing the parameters of the temperature source and controlling the noise parameters in the sensors.

We compare the performance of:

- One robot without on-board sensors utilizing the *S-Net* to reach the closest temperature source.
- One robot using only on-board sensors to reach the closest temperature source.

Examples of these two alternatives are shown in Figures 5.1 and 5.2, in which there is one temperature source (“ \diamond ”) in the environment. Figure 5.1 is an isotherm plot describing the distribution of temperature, in which a robot starts at the origin (0, 0) and follows the temperature gradient to reach the source, which is located at (3, 2). In order to make the isotherm plot more clear, we amplify the unit by 10. Figure 5.2 not only give out the trace of a robot goes to the destination with the *S-Net*, but also provide the formation of local frames. In the figure, each “+” is one s-element, and each “*” is the origin of one local frame, and “— — —” is the range of local frame origin. As we mentioned before, there should be at least two common s-elements between any two close frames in order to transform between each other. In Figure 5.2, a robot also starts at the origin (0, 0) and moves to the temperature source (3, 2) by using local s-clique frames. Since by using the *S-Net*, the robot can only estimate the destination location in its own frame, as well as the fact that the robot has a non-zero turning radius, this causes the robot to go to a place close to the temperature source instead of exactly to the source.

Constants set up for the experiments:

in which the initial location and initial direction of robot, as well as the location of source are all according to home frame of environment instead of any local frame.

In these experiments we test performance time and distance traveled with respect to sensor noise or variance (0 to 25), number of s-elements (100 to 300), and broadcast distance (1 to 2.5) for the s-elements. According to [2, 8, 4], noise of

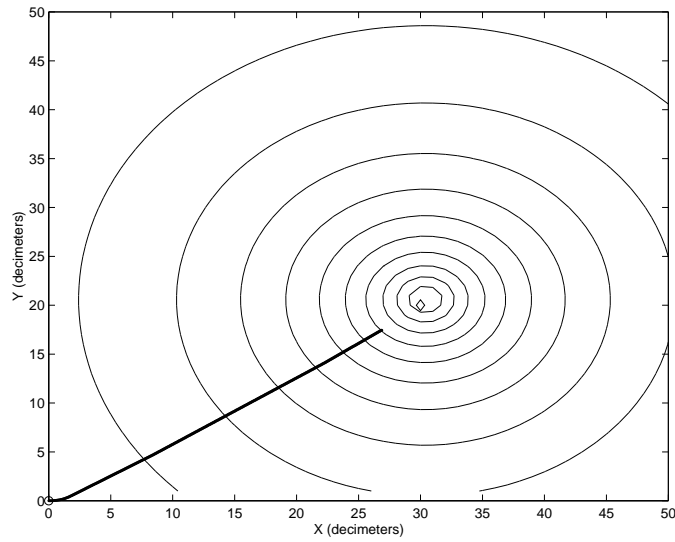


Figure 5.1. Robot goes to destination without *S-Net*

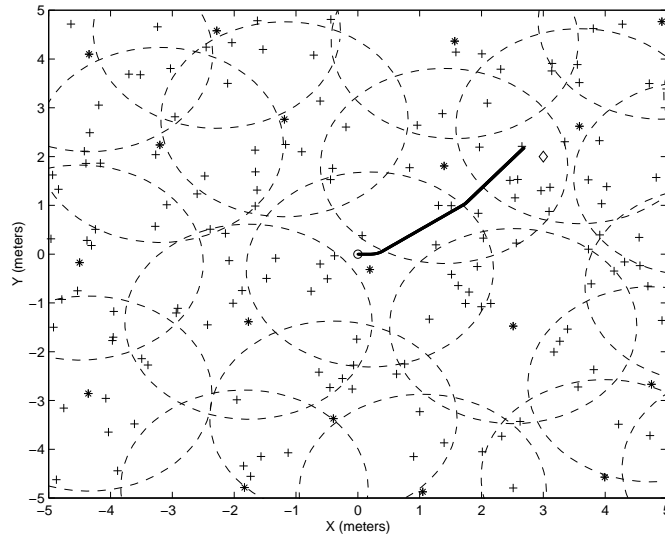


Figure 5.2. Robot goes to destination with *S-Net*

| | |
|-----------------------------------|----------------------------|
| <i>Maximum Linear Speed</i> | 1 m/s |
| <i>Maximum Rotation Speed</i> | π rad/s |
| <i>Initial Location of Robot</i> | (0, 0) ("o" in the figure) |
| <i>Initial Direction of Robot</i> | 0 rad |
| <i>Number of Sources</i> | 1 |
| <i>Location of Source</i> | (3, 2) |

a sensor includes inherent noise, transmitted noise, mechanical noise and seeback noise and so on. The temperature sensor model we choose here have the range of $[0, 1000^{\circ}C]$, and we think that 0.05% is a reasonable tolerance for the temperature sensors. That is why we choose σ^2 ranges from 0 to 25. The number of s-elements is chosen according to the principle that we want to use as little s-elements as possible, but we still want the *S-Net* functions well, which means there should be enough s-elements to satisfy local frame transformation. Generally, the broadcast distance is closely related to the costs of devices. We try to keep the price as low as possible, so we think ranges of 1 meter to 2.5 meters is enough for the local frame establishment.

The results are displayed in Figures 5.3 to 5.15. Each data point represents the estimated value of the performance measure of interest, and is the mean of ten simulation experiments. The variance is also shown. The stochastic part of each experiment is the location of the s-elements. The reason why we choose ten simulation experiments is according to the confidence interval. Suppose we want to obtain an approximate 90% confidence interval for the expected average time utilization, which is given by:

$$E(X) = E\left(\frac{\sum_{i=1}^N D_i}{N}\right)$$

From the ten replications we obtain:

$$\bar{X}(10) = 4.95$$

$$S^2(10) = 1.34$$

and the confidence interval is:

$$\begin{aligned} & \bar{X}(10) \pm t_{n-1, 1-\frac{\alpha}{2}} \sqrt{\frac{S^2(n)}{n}} \\ & = \bar{X}(10) \pm t_{9, 0.95} \sqrt{\frac{S^2(10)}{10}} = 4.95 \pm 0.679 \end{aligned}$$

Thus, subject to the correct interpretation to be given to confidence interval, we can claim with approximately 90% confidence that $E(X)$ is contained in the interval $[4.271, 5.629]$ seconds.

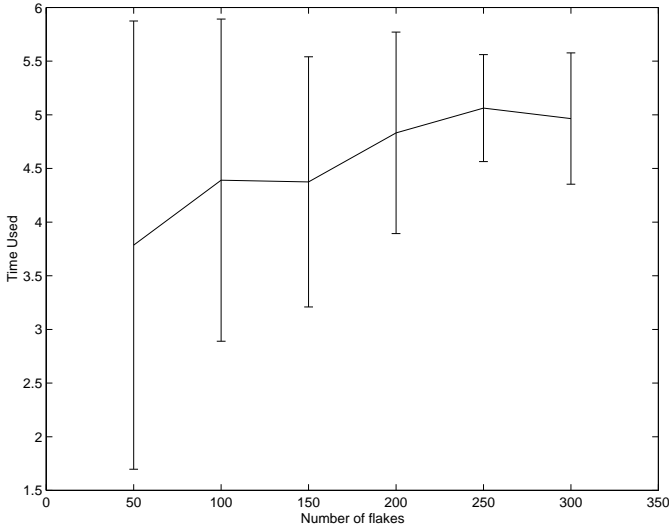


Figure 5.3. Time vs. Number of S-Elements with *S-Net* (Range = 2)

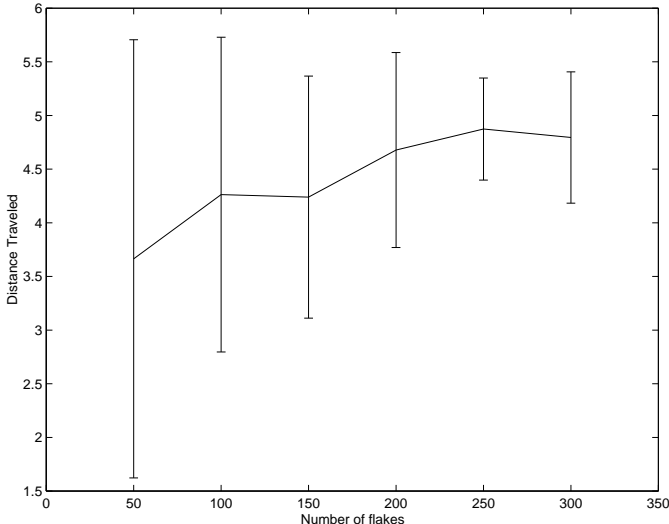


Figure 5.4. Distance Traveled vs. Number of S-Elements with *S-Net* (Range = 2)

The robot first communicates with the s-clique origin with the highest temperature within communication range, then gets the information about the local frame from it. After the robot determines its own position in the s-clique’s frame, it transform the location of the s-element, with the highest temperature value, to its own frame and then goes to that location. This process repeats through s-clique within range until robot reaches the s-element with the highest global temperature.

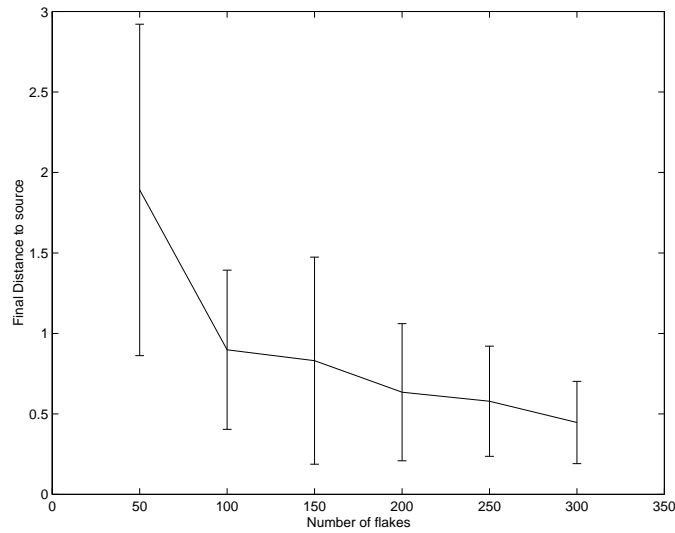


Figure 5.5. Final Distance to Source vs. Number of S-Elements with *S-Net* (Range = 2)

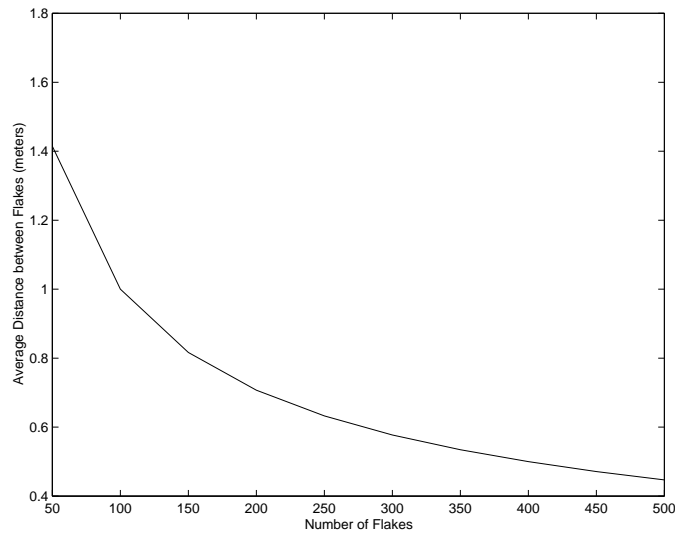


Figure 5.6. Average Distance of S-Elements vs. Number of S-Elements in the Area

Figures 5.3 and 5.4 show that more s-elements will cause an increase in both time used and distance traveled by the robot. This effect is explained below.

According to Figure 5.5, we can see that when the number of s-elements increases, the final distance of the robot from the temperature source decreases dramatically. And when the number of s-elements increases, the variances of both

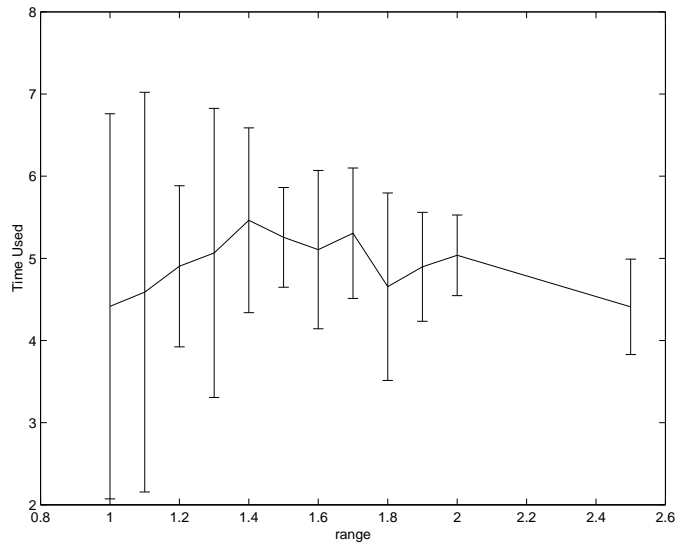


Figure 5.7. Time vs. Range with *S-Net* (Number of S-Elements = 250)

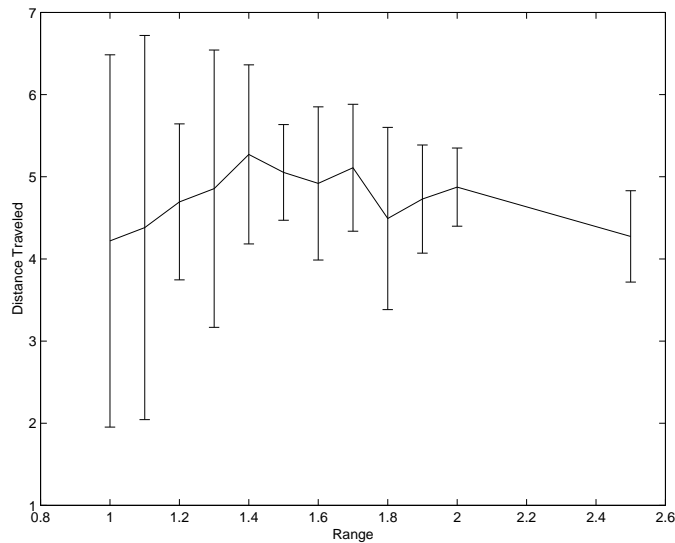


Figure 5.8. Distance Traveled vs. Range with *S-Net* (Number of S-Elements = 250)

time utilized and distance traveled decrease, which means the system becomes more robust. The fact that time used and distance traveled increase with number of s-elements relates to the density of the s-element set and the ability of the *S-Net* to provide adequate spatial resolution. Because the increase of number of s-elements will cause the increase of number of s-cliques, or local frames, formed, so more local

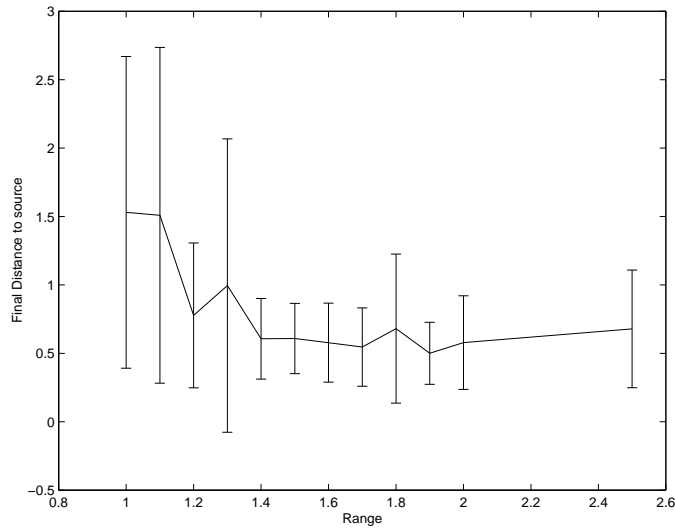


Figure 5.9. Final Distance to Source vs. Range with *S-Net* (Number of S-Elements = 250)

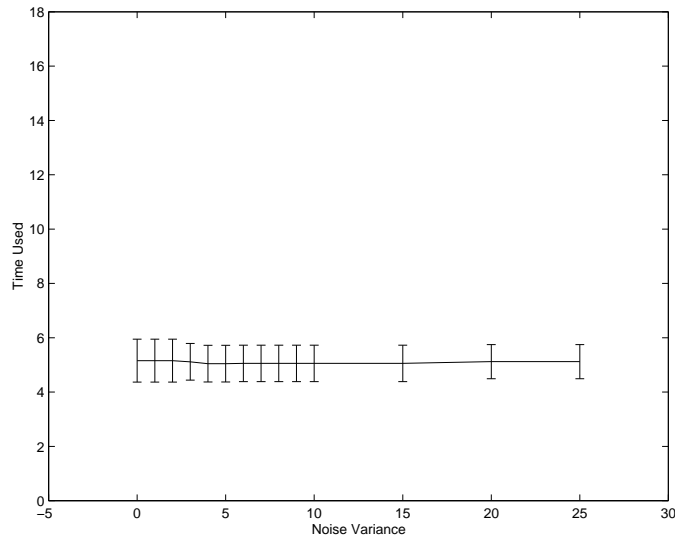


Figure 5.10. Time vs. Variance of Noise With *S-Net* (Number of S-Elements = 300, Range = 2)

frame transformation will be done by robot. When robot tries to reach the highest s-element of each local frame the path becomes more zigzag. But the pay off is that the final distance of robot to the temperature source decreases.

Figure 5.6 shows quantitatively that as the number of s-elements increases, the average distance from each s-element to the nearest other s-element decreases. So

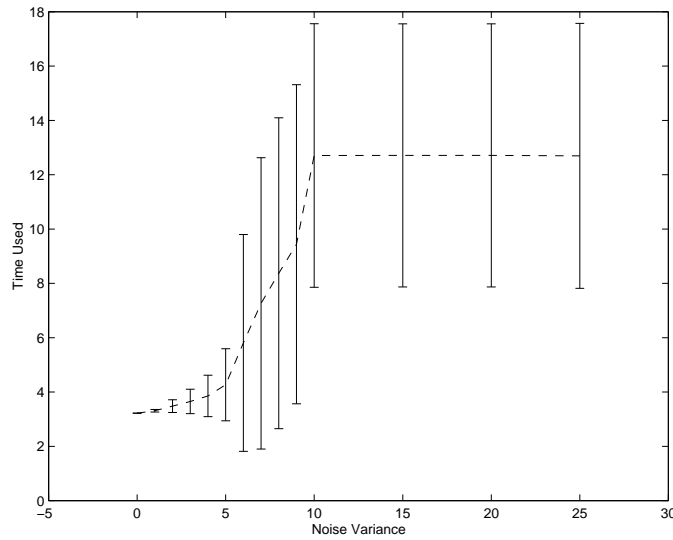


Figure 5.11. Time vs. Variance of Noise Without *S-Net*

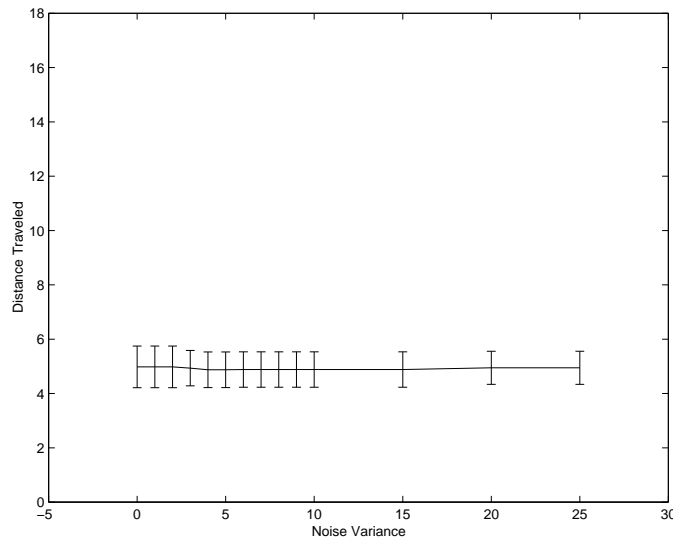


Figure 5.12. Distance Traveled vs. Variance of Noise With *S-Net* (Number of S-Elements = 300, Range = 2)

suppose the robot can reach the highest s-element of the whole system, when given a specified accuracy for the final location with respect to a temperature source, we can determine the number of s-elements required. For example, to guarantee that a mobile robot can reach a temperature source within 1 meter, we need at least 100 s-elements over an area about $10m * 10m$ (the radius of each robot is 0.1m).

According to Figures 5.7, 5.8, and 5.9, we can see that when the radio broadcast

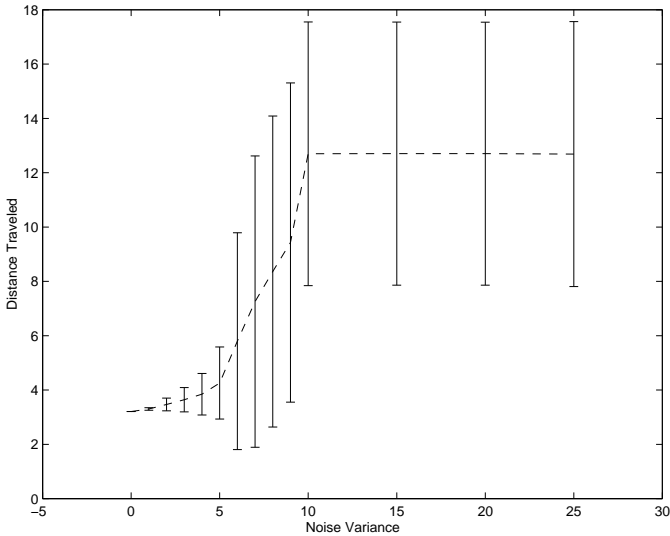


Figure 5.13. Distance Traveled vs. Variance of Noise Without *S-Net*

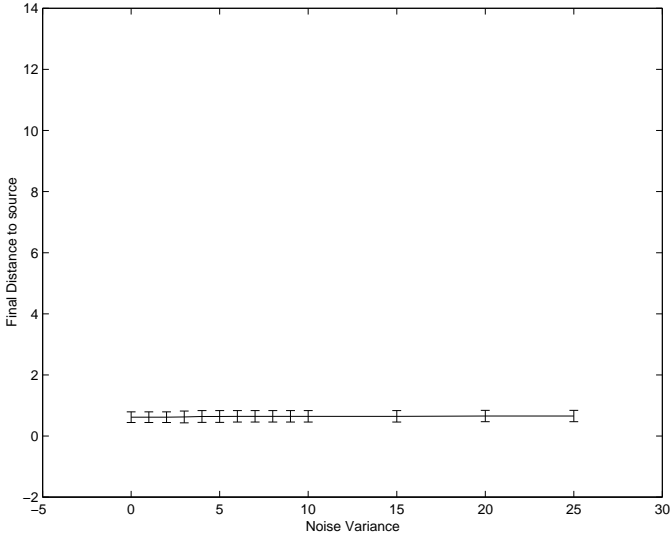


Figure 5.14. Final Distance to Source vs. Variance of Noise With *S-Net* (Number of S-Elements = 300, Range = 2)

range is less than 1.5m, the behavior is less robust (variances are large). When the range is between 1.5 and 2.0m, the variance decreases and the behavior becomes more robust.

Figures 5.10 to 5.15 show the results after adding noise to the sensors, in which “...” is the result of a mobile robot without the *S-Net*, and “—” is the result of the behavior with the *S-Net*. For those with the *S-Net*, we did ten simulation runs

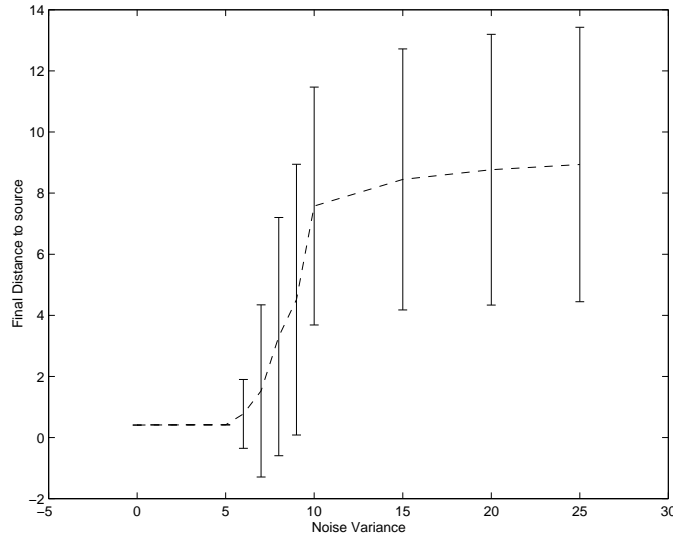


Figure 5.15. Final Distance to Source vs. Variance of Noise Without *S-Net*

by changing the distribution of s -elements, and the mean and standard deviation are shown in each sample point. The values obtained by each sensor (including on-board sensors and all the s -elements) are normally distributed with parameters (μ, σ) , in which μ is the ideal temperature value, and σ is the variance. In fact, each sensor smooth its data value by taking ten samples and returning the average, for both systems with or without the *S-Net*.

From these figures, we can clearly see that, when the noise variance is above 10, for the mobile robot that utilizes the *S-Net*, the result does not change much. But for the mobile robot that uses on-board sensors, it so happens that the robot fails to locate the temperature source correctly (Figure 5.16). The times and distances traveled by the mobile robot depend on this limit; the maximum time allowed for the task is 15 time units. In theory, it might never locate the source. This is because the four on-board sensors are located too close to each other, so the temperatures they report are too noisy to be useful. When noise is added to each sensor, the gradient computed from their values can have large error, which will further change the direction the mobile robot moves. One proposed solution to this problem is to have the mobile robot move to four widely spaced locations and get samples across a greater spatial scale to compute the correct gradient. This will certainly cost

much in time and energy. In fact, it also reduces the accuracy with which the robot can locate the source.

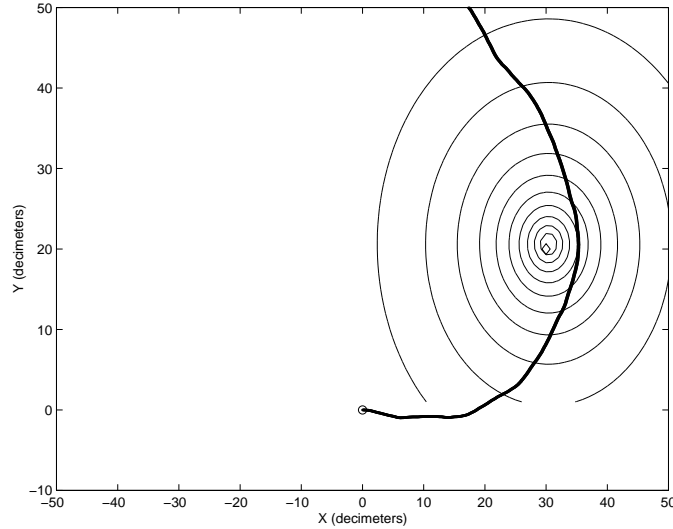


Figure 5.16. Robot goes to destination without *S-Net* (with noise variance = 10)

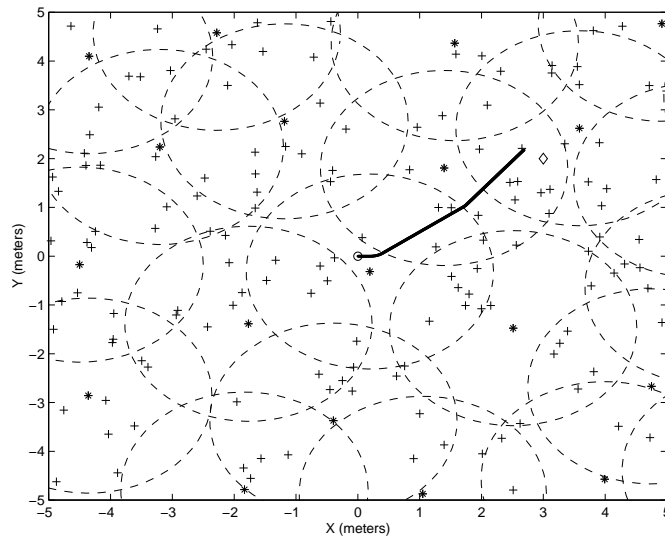


Figure 5.17. Robot goes to destination with *S-Net* (with noise variance = 10)

For some specific cases, e.g., there are two or more temperature sources in the environment, and mobile robot located in the exact middle area of the sources, it is difficult for the mobile robot to locate the source when noise is added to the

sensors. Figures 5.18 and 5.19 show the results of this experiment, with noise variance $\sigma = 8$, the mobile robot with on-board sensors can not figure out the exact location of the temperature source.

From all these measurement and comparison of two systems, we can see that when in ideal situation, which means no noise, the *S-Net* takes more time and distance. Compared to system without *S-Net* (time used = 3.22sec, distance traveled = 3.21m), our cost of time ranges from 3.6(sec) to 5.5(sec), and distance traveled from 3.6m to 5.2m. But when noises are added in, which is more realistic, the *S-Net* system basically does not change much, but the system without the *S-Net* gets much worse. So we conclude that when in real situation, especially tough situation with lots of noise, the *S-net* system will be more robust than the system without the *S-Net*.

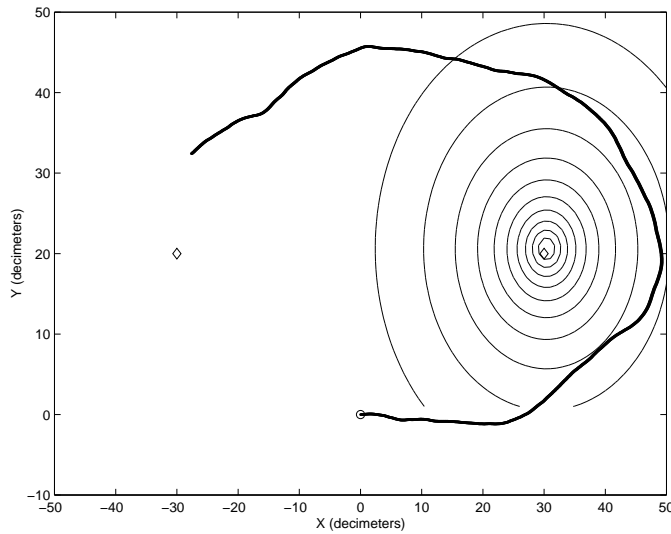


Figure 5.18. Robot goes to destination without *S-Net* (with noise variance = 8)

5.2 Multiple Robots Surround Temperature Source Evenly

This experiment is designed to explore the benefits of using the *S-Net* with regard to multiple mobile robot cooperation. Also, we use the same behavior in all the

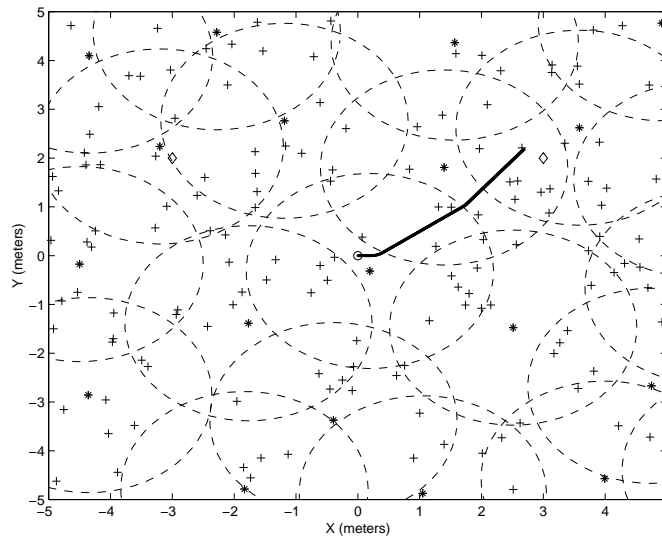


Figure 5.19. Robot goes to destination with *S-Net* (with noise variance = 8)

robots, so that by satisfying the same set of constraints, the robots can achieve the desired final result.

We compare the performance of:

- Three mobile robots without on board temperature sensors utilizing the *S-Net* to surround the s-element with the highest temperature value.
- Three mobile robots using only on-board temperature sensors to surround the temperature source evenly.

Examples of these two alternatives are shown in Figures 5.20 and 5.21, in which three mobile robots originated from different places (“o”), and there is a temperature source (“◊”) in the environment. “*” in Figure 5.20 is the position of the s-element with highest temperature.

Constants set up for the experiments:

In these experiments we test performance time and distance traveled with respect to sensor noise or variance (0 to 25), number of s-elements (100 to 300), and broadcast distance (1 to 2.5) for the s-elements. The reason why we choose ten simulation experiments is according to the confidence interval. Suppose we want to obtain an approximate 90% confidence interval for the expected average time

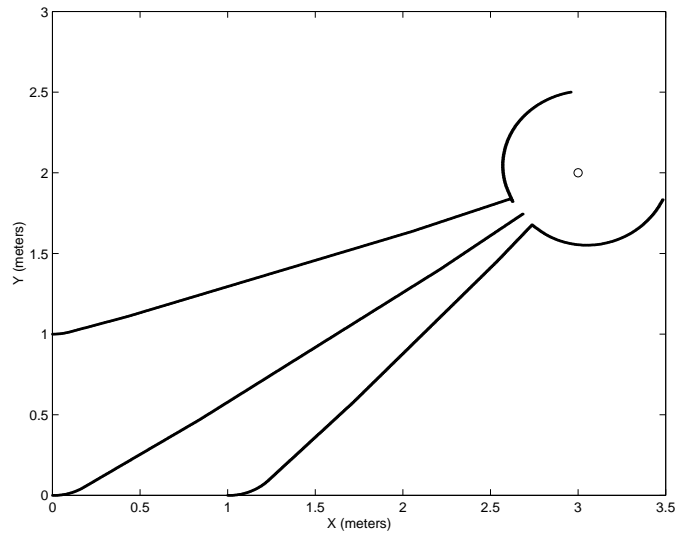


Figure 5.20. Three Robots Surround the Temperature Source Without *S-Net*

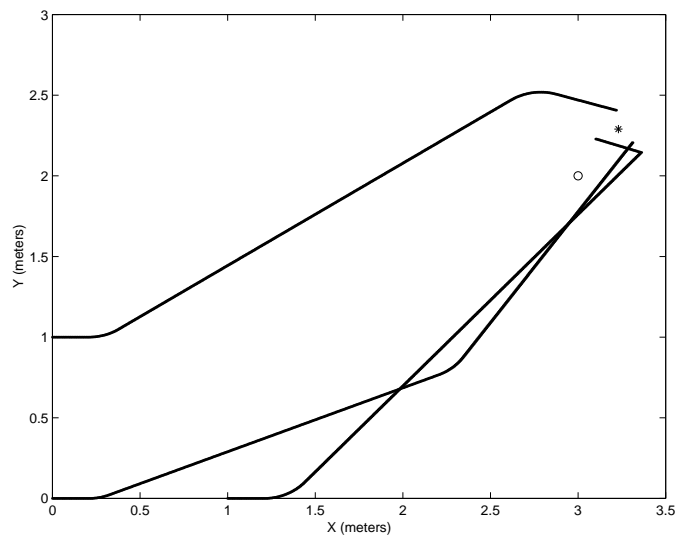


Figure 5.21. Three Robots Surround the Highest S-Element with *S-Net*

| | Robot 1 | Robot 2 | Robot 3 |
|-----------------------------------|-------------|-------------|-------------|
| <i>Maximum Linear Speed</i> | 1 m/s | 1 m/s | 1 m/s |
| <i>Maximum Rotation Speed</i> | π rad/s | π rad/s | π rad/s |
| <i>Initial Location of Robots</i> | (0, 0) | (1, 0) | (0, 1) |
| <i>Initial Direction of Robot</i> | 0 rad | 0 rad | $\pi/2$ rad |
| <i>Number of Sources</i> | 1 | | |
| <i>Location of Source</i> | (3, 2) | | |

utilization, which is given by:

$$E(X) = E\left(\frac{\sum_{i=1}^N D_i}{N}\right)$$

From the ten replications we obtain:

$$\bar{X}(10) = 4.72$$

$$S^2(10) = 0.732$$

and the confidence interval is:

$$\begin{aligned} & \bar{X}(10) \pm t_{n-1, 1-\frac{\alpha}{2}} \sqrt{\frac{S^2(n)}{n}} \\ & = \bar{X}(10) \pm t_{9, 0.95} \sqrt{\frac{S^2(10)}{10}} = 4.72 \pm 0.496 \end{aligned}$$

Thus, subject to the correct interpretation to be given to confidence interval, we can claim with approximately 90% confidence that $E(X)$ is contained in the interval [4.224, 5.216] seconds.

The results are displayed in Figures 5.22 to 5.34. Each data point represents the estimated value of the performance measure of interest, and is the mean of ten simulation experiments. The variance is also shown. The stochastic part of each experiment is the location of the s-elements.

According to Figure 5.24, we can see that when the number of s-elements increases, the final distance of the robot from the temperature source decreases dramatically. When the number of s-elements increases, the variance of time utilized and distance traveled decrease; this means the system becomes more robust. The fact that time and distance traveled increase with the number of s-elements relates to the density of the s-element set and the ability of the *S-Net* to provide adequate spatial resolution.

According to Figures 5.25 to 5.27, we notice that in this particular behavior, range does not affect the results much. This is because the parameters, such as time utilized, distance traveled and final distance to source, depend not only on each specific robot, but also the communication and cooperation of the three robots,

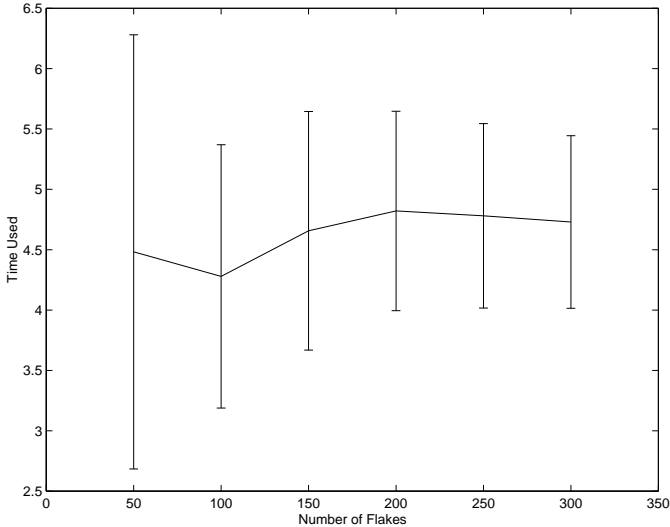


Figure 5.22. Time vs. Number of S-Elements for Robots with *S-Net* (Range = 2)

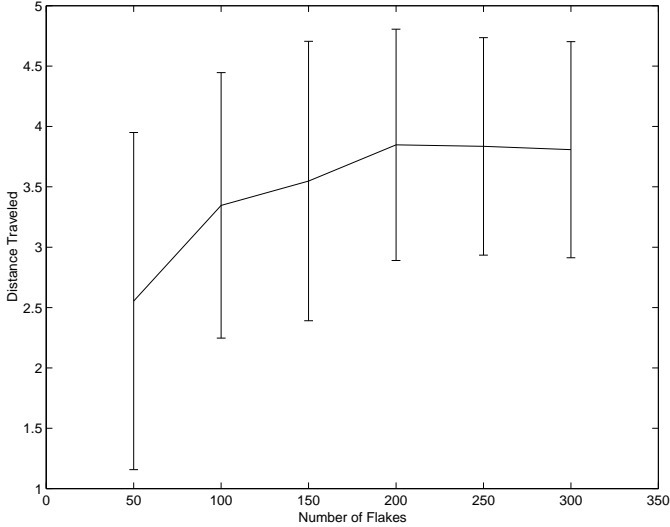


Figure 5.23. Distance Traveled vs. Number of S-Elements for Robots with *S-Net* (Range = 2)

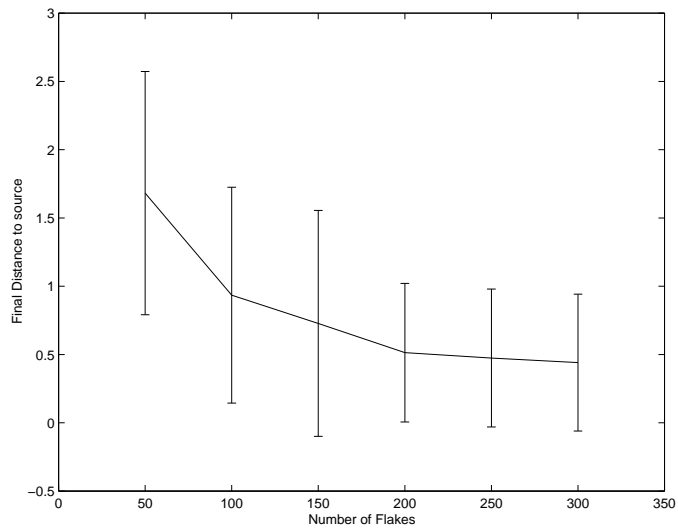


Figure 5.24. Final Distance to Source vs. Number of S-Elements for Robots with *S-Net* (Range = 2)

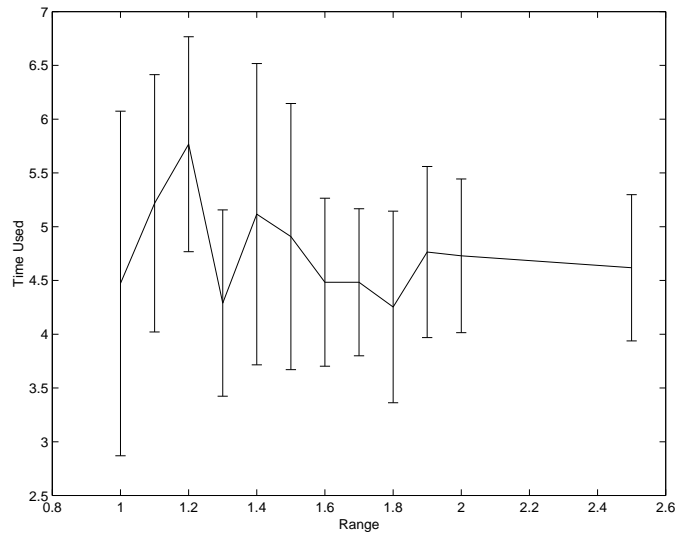


Figure 5.25. Time vs. Range for Robots with *S-Net* (Number of S-Elements = 300)

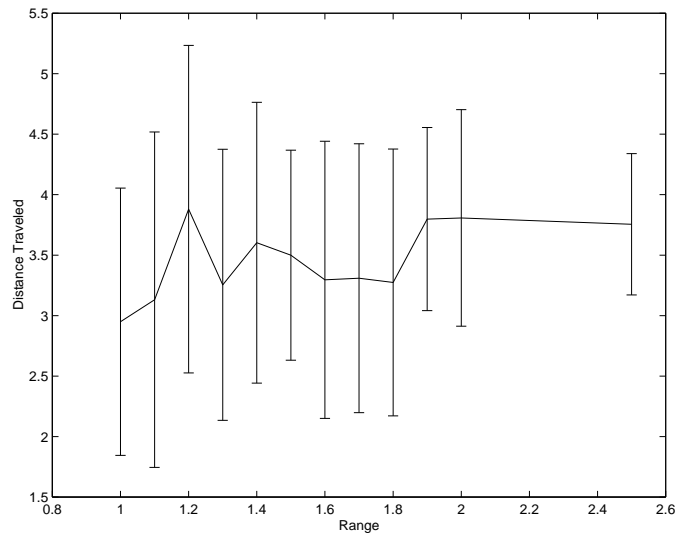


Figure 5.26. Distance Traveled vs. Range for Robots with *S-Net* (Number of S-Elements = 300)

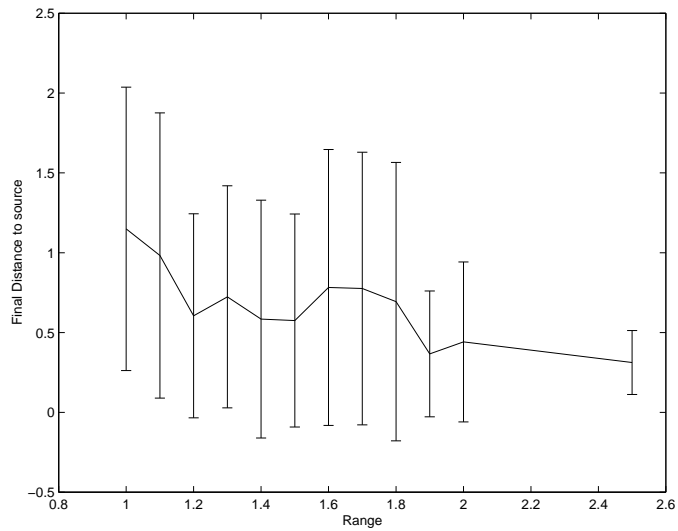


Figure 5.27. Final Distance to Source vs. Range for Robots with *S-Net* (Number of S-Elements = 300)

which decrease the effect of range on the robustness of this behavior. Range can only control the part that each robot try to get close to the temperature source, but does not have any effect on the later part, in which the three robots try to cooperate and surround the highest s-element. However, both parts account for the time used and distance traveled. The only thing we are sure is that the final distance to the source is decreased, and this is majorly because of the effect of the range parameter.

Figures 5.29 to 5.34 show the results of adding noise to sensors, in which “...” is the result of mobile robots without the *S-Net*, and “—” is the result of the behavior with the *S-Net*. For those with the *S-Net*, we did ten simulation runs by changing the distribution of s-elements, so the mean and standard deviation are shown in each sample point. The values obtained by each sensor (including on-board sensors and all the s-elements) are normally distributed with parameters (μ, σ) , in which μ is the ideal temperature value, and σ is the variance. In fact, each sensor smooth the data value by taking ten samples and returning the average.

From these figures, we can clearly see that, when the noise variance is above 10, for the mobile robot that utilizes the *S-Net*, the result does not change much. But for the mobile robot that uses on-board sensors, the robot can not surround the temperature source correctly (Figure 5.28). The results here are similar to those in Section 5.1. In the robot system without the *S-Net*, when robots reach certain distance away from the temperature source, which means when their sensed temperatures are above some level, the robots will begin to cooperate. One robot, which is closer to the others will maintain its position, and the other robots will move following a constant-valued temperature contour. To do this, they compute the gradient and move perpendicularly to it. While tracking the contour, even a small amount of noise will cause them to move away from the contour, and results in their inability to finish the surrounding task. On the other hand, in the *S-Net* system, when robots reach a certain distance away from the highest temperature s-element, two of the robots keep their positions, and the other one which is farthest from them will move forward to form an equilateral triangle. So we conclude that

when in real situations, especially tough situations with lots of noise, the *S-Net* system will be more robust than the system without the *S-Net*.

5.3 Multiple Robots Go Back and Forth to the Temperature Source

This experiment is also designed to explore the benefits of using the *S-Net* with regard to multiple cooperating mobile robots. Also, we would like to use the same behavior in all the robots, so that by satisfying the same set of constraints, the robots can achieve the desired final result.

We compare the performance of:

- Multiple mobile robots without on-board temperature sensors which utilize a stripe pattern formed by the *S-Net* to go back and forth to the temperature source from a home location.
- Multiple mobile robots using only on-board temperature sensors and *Home* sensors to go back and forth to the temperature source from a home location.

Examples of these two alternatives are shown in Figures 5.35 and 5.36, in which mobile robots originated from *Home* (“o” in figures), and there is a temperature source (“◇” in figures) in the environment.

Constants set up for the experiment:

| | Robot |
|-----------------------------------|---------------|
| <i>Maximum Linear Speed</i> | 10 m/s |
| <i>Maximum Rotation Speed</i> | 20π rad/s |
| <i>Initial Direction of Robot</i> | 0 rad |
| <i>Number of Sources</i> | 1 |

In the case that mobile robots use the *S-Net* (500 s-elements), *Home* is chosen as the origin of the s-clique that sensed the lowest temperature; this provides the longest path to the maximum temperature s-element . Then stripe patterns are formed along the gradient of the temperature source to *Home*. The straight line of *Home* to temperature source (located in [10, 20]) is in the middle of the white stripe

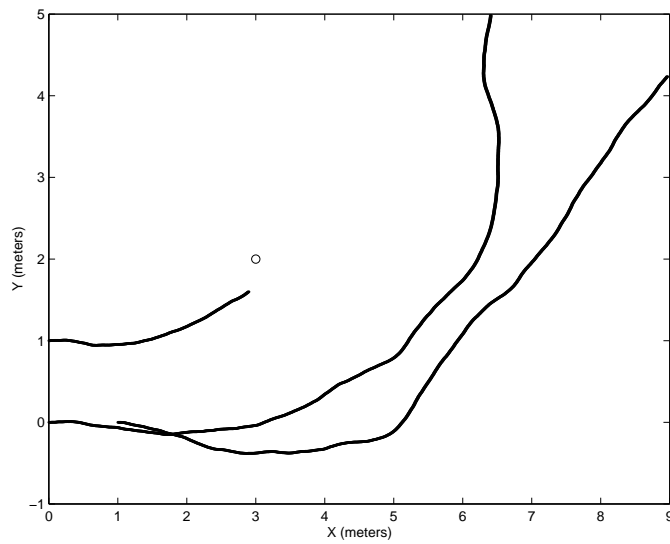


Figure 5.28. Robots Surround the Temperature Source without *S-Net* (with noise variance = 10)

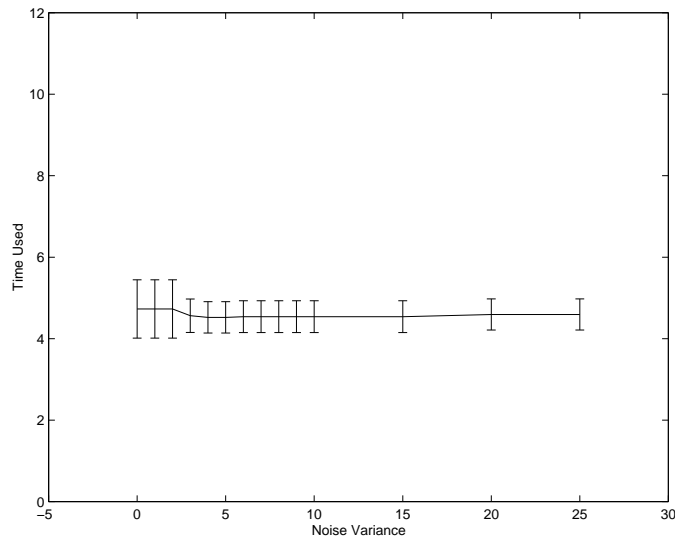


Figure 5.29. Time vs. Variance of Noise for Robots with *S-Net* (Number of S-Elements = 300, Range = 2)

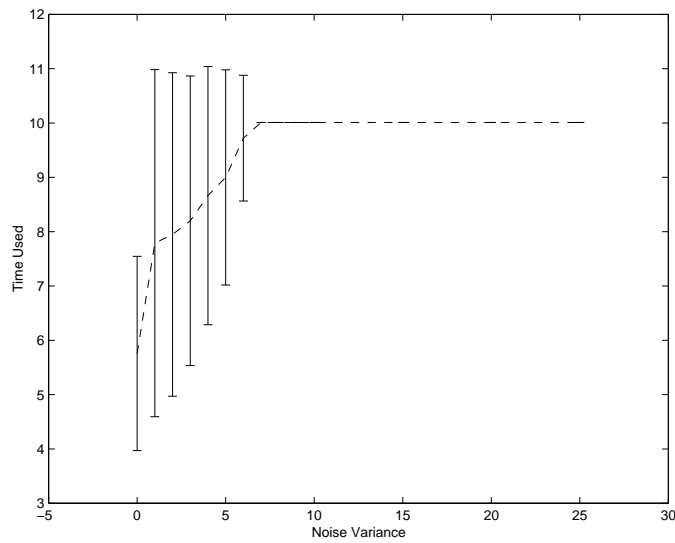


Figure 5.30. Time vs. Variance of Noise for Robots without *S-Net*

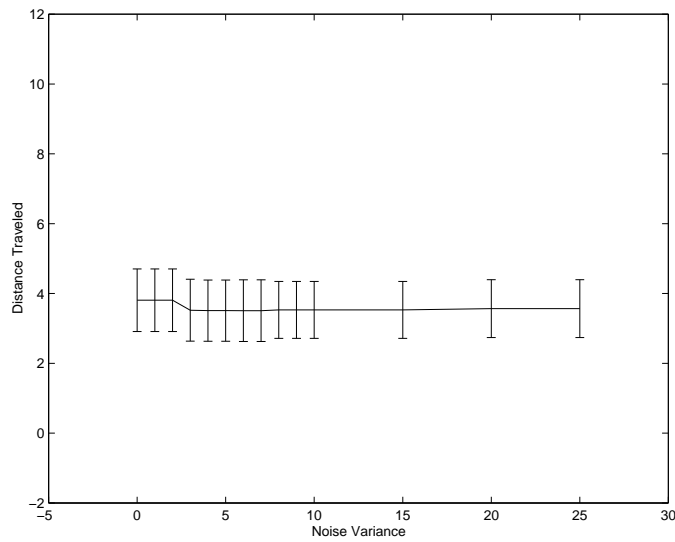


Figure 5.31. Distance Traveled vs. Variance of Noise for Robots With *S-Net* (Number of S-Elements = 300, Range = 2)

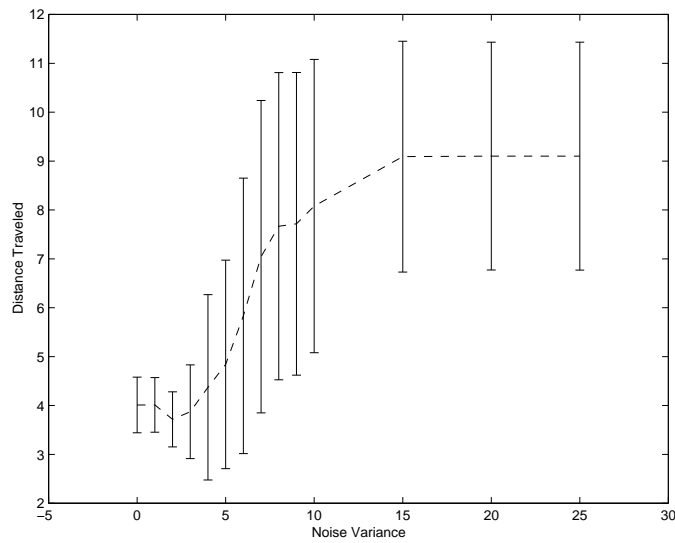


Figure 5.32. Distance Traveled vs. Variance of Noise for Robots Without *S-Net*

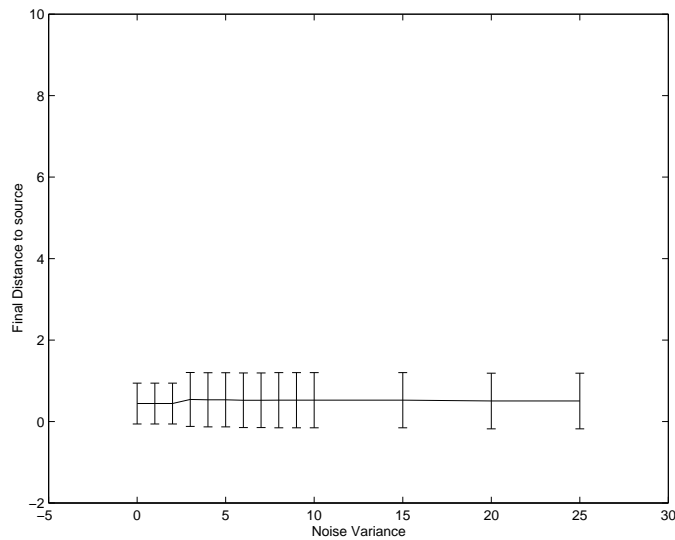


Figure 5.33. Final Distance to Source vs. Variance of Noise for Robots With *S-Net* (Number of S-Elements = 300, Range = 2)

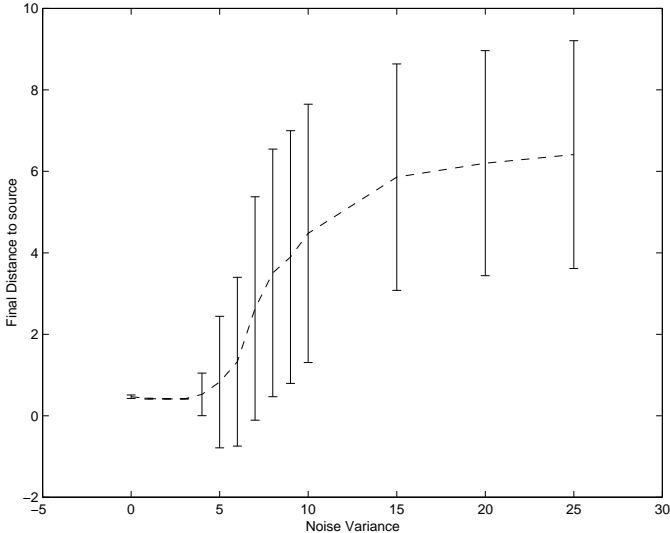


Figure 5.34. Final Distance to Source vs. Variance of Noise for Robots Without *S-Net*

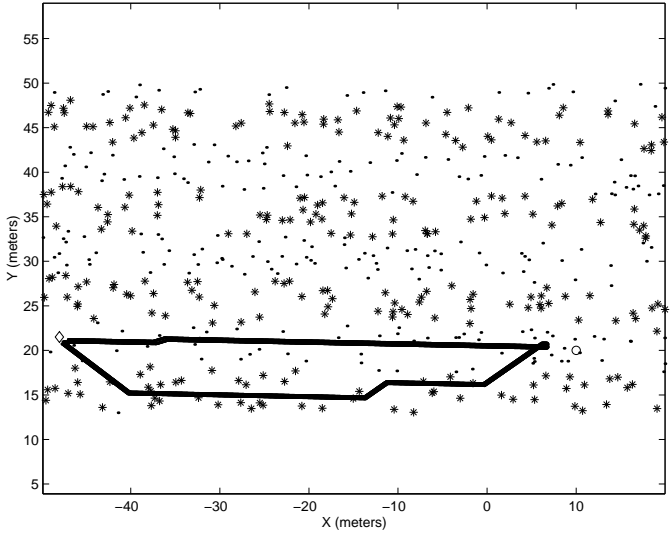


Figure 5.35. Trace of robots going back and forth with *S-Net* (2 robots, 2 round trips)

(pattern value is 1), the width of each stripe is a constant (5m in our case), and black stripes (pattern value is 0) alternate spatially with white stripes. Different stripe patterns are formed for different random streams. The robots will move along the white stripe toward the temperature maximum and follow the black stripe *Home*. When a robot detects that a collision is about to happen, it will slow down to prevent the collision.

In the case that mobile robots do not use the *S-Net*, *Home* is arbitrarily located at the origin of environment (0, 0), and the temperature source is located according to the average distance of *Home* to the temperature source in the *S-Net* experiments. The purpose of this is to make sure that the distances of the round trip are basically the same for both setups. We believe that the gradient of temperature source to *Home* does not affect our experiment, so we choose the temperature source located in (40, 46). When robots detect an environment collision, they make a right turn and then try get back on track again. The robots do not use the method of slowing down, because it is difficult and time consuming to determine whether the robot in front is moving toward them or moving in the same direction as they are moving.

In these experiments we test performance time and distance traveled, with respect to the number of round trips (1 to 10) and the number of robots on the path (1 to 10). The results are displayed in Figures 5.37 to 5.40. Each data point represents the estimated value of the performance measure of interest, and is the mean of twelve simulation experiments. The stochastic part of each experiment is the location of the s-elements.

Figures 5.37 and 5.38 give the performance of the *S-Net* system, from which we can see that when the number of robots and trips increase, the average time used and distance traveled by each robot increases linearly, and there are no major deviations from linear. When we take a close look at the data collected, we find that on occasion, due to the particular random number streams, the result is not ideal, which means the robots can not exactly follow the stripe, but get lost looking for the correct stripe. Under detailed analysis, we found that it is caused by some particular distributions of s-elements. Since we use the origin with lowest temperature as the

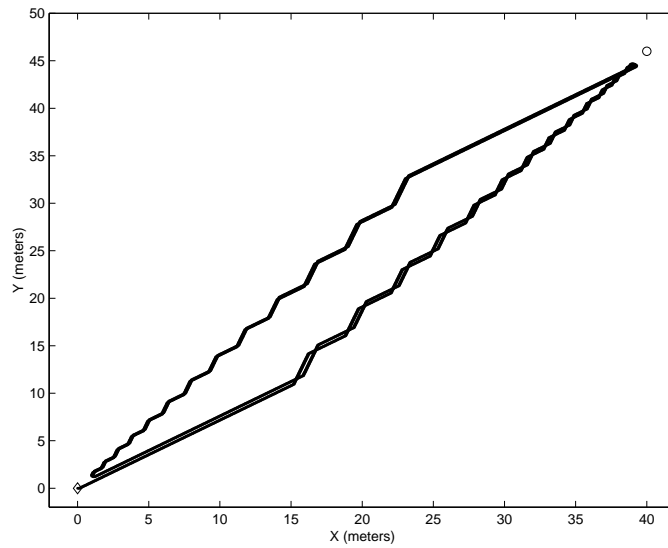


Figure 5.36. Trace of robots going back and forth without *S-Net* (2 robots, 2 round trips)

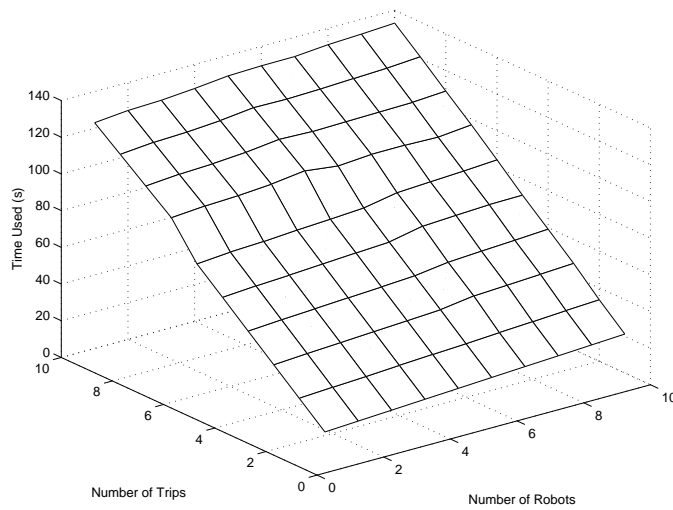


Figure 5.37. Time used by up to 10 robots for up to 10 round trips with *S-net* (number of s-elements = 500)

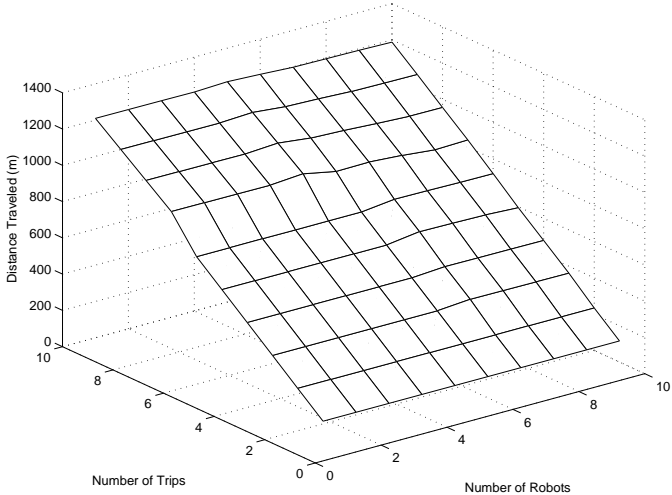


Figure 5.38. Distance traveled by up to 10 robots for up to 10 round trips with *S-net* (number of s-elements = 500)

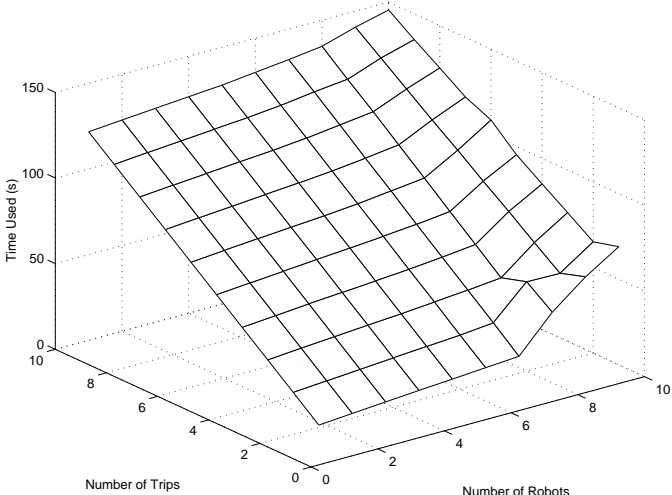


Figure 5.39. Time used by up to 10 robots for up to 10 round trips without *S-Net*

Home, it is sometimes possible that it is on the border of the stripe. There may not be enough s-elements on the border for black stripes, and then when the robots try to follow the stripe to go *Home*, they may not get enough information to keep on the black stripe, and thus move away. This can be solved by making more s-elements on the border or making *Home* far away from borders. But this is just some particular cases, which can be handled accordingly in reality, and is not unsolvable.

Figures 5.39 and 5.40 give the performance of the system without the *S-Net*, from which we can see that when the number of robots and trips increased, the average time used and distance traveled by each robot increase linearly. After a detailed analysis of the data collected, we found that when there are more than eight robots on the same path, several robots may lose control. This is related to the robot behavior chosen. While there are lots of other behaviors, we believe that this is a rather standard collision avoidance algorithm and representative of many implementation in physical systems.

Figures 5.41 and 5.42 show how noise affects the performance in both cases. From these figures, we found that the one using *S-Net* can handle noise very well, when the noise variance is about 10, the performance and trace of robots generally stay the same. But for the case that does not use the *S-Net*, noise variance has a huge effect on the performance of the robots, where even a tiny variance as little as 0.5 can cause the robots to lose control. So we can conclude that, in terms of noise, the *S-Net* performs much better.

The performance measures used to this point have looked at success/failure, time to goal and distance traveled. Another crucial aspect is the more qualitative users' defined cost which is, in general, a function of the physical performance measures. For example, it may be that timeliness is extremely important, and the user may assign an exponential cost to time. Even if the cost is directly proportional, it may be linearly related with a steep slope.

To explore this aspect of performance cost, we have set up two models: (1) linear, and (2) quadratic. The three major terms included are:

- robot cost: we always assume this is linear in the number of robots.

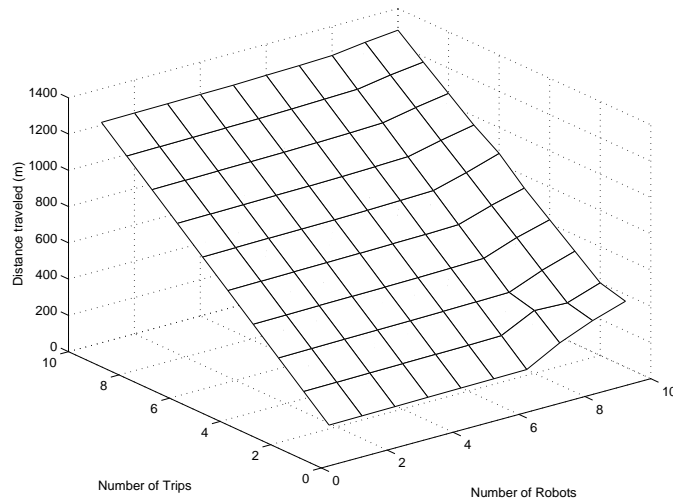


Figure 5.40. Distance used by up to 10 robots for up to 10 round trips without *S-Net*

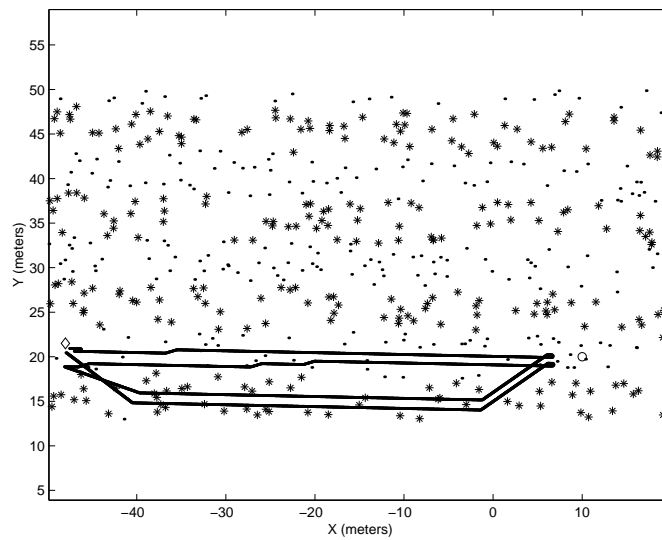


Figure 5.41. Trace of robots going back and forth with *S-Net* (2 robots, 2 round trips, noise = 10)

- s-net cost: we always assume this is linear in the number of s-elements.
- physical quantitative (e.g., time and distance determined from simulation experiments): we apply a linear or quadratic forum to this term.

In order to explore this issue, we examined linear and quadratic cost functions in terms of parameters in the equations in order to determine the existence of various cost requires related to parameter values.

We define the cost relation as:

$$C_l = C_s + C_p$$

where:

- C_l is the total cost
- C_s is the cost of the system infrastructure
- C_p is the cost of performance
- $C_s = N_r * C_r + N_{s-el} * C_{s-el}$
- $C_p = a_t * t^k + a_d * d^k$

in which $k = 1$ in the linear case, and $k = 2$ in the quadratic case.

The performance data with out the *S-Net* are given by Tables T1, and T2 (time and distance, respectively), and performance with *S-Net* is given by Tables T3 and T4. We compare the two systems (without and with *S-Net*) by computing the number of cases for which the *S-Net* system outperforms the non-S-Net system (over the 100 cases of experiment).

The time used for system without *S-Net* is:

| trips | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| robots | | | | | | | | | | |
| 1 | 12.29 | 13.82 | 15.30 | 16.68 | 18.06 | 19.43 | 20.77 | 41.25 | 56.67 | 69.01 |
| 2 | 24.45 | 25.98 | 27.43 | 28.83 | 30.21 | 31.60 | 33.53 | 52.23 | 51.88 | 65.00 |
| 3 | 36.61 | 38.20 | 39.67 | 41.08 | 42.49 | 44.05 | 45.75 | 47.63 | 62.92 | 75.14 |
| 4 | 48.76 | 50.32 | 51.90 | 53.31 | 54.73 | 56.20 | 58.08 | 60.18 | 74.19 | 85.06 |
| 5 | 60.90 | 62.53 | 64.05 | 65.58 | 67.06 | 68.65 | 70.38 | 72.64 | 85.03 | 95.15 |
| 6 | 73.04 | 74.68 | 76.23 | 77.70 | 79.09 | 80.82 | 83.04 | 85.31 | 96.28 | 108.7 |
| 7 | 85.20 | 86.81 | 88.35 | 89.89 | 91.34 | 93.11 | 95.30 | 97.78 | 107.1 | 115.3 |
| 8 | 97.36 | 99.0 | 100.6 | 102.1 | 103.5 | 105.4 | 107.9 | 110.3 | 118.3 | 125.3 |
| 9 | 109.52 | 111.2 | 112.7 | 114.3 | 115.7 | 117.6 | 120.2 | 122.3 | 129.5 | 135.0 |
| 10 | 121.68 | 123.3 | 124.9 | 126.4 | 127.9 | 130.1 | 132.2 | 134.7 | 140.6 | 145.4 |

The distance traveled for system without *S-Net* is:

| trips | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| robots | | | | | | | | | | |
| 1 | 122.7 | 124.6 | 126.7 | 128.5 | 130.4 | 132.2 | 133.8 | 232.1 | 305.3 | 362.8 |
| 2 | 244.1 | 246.0 | 247.9 | 249.8 | 251.6 | 253.4 | 257.1 | 346.3 | 339.0 | 400.1 |
| 3 | 365.5 | 367.6 | 369.7 | 371.5 | 373.5 | 375.7 | 378.9 | 382.6 | 454.3 | 510.6 |
| 4 | 486.8 | 488.8 | 491.0 | 492.9 | 494.8 | 497.2 | 500.8 | 505.5 | 570.5 | 620.0 |
| 5 | 608.0 | 610.4 | 612.6 | 614.7 | 616.9 | 619.6 | 622.6 | 627.9 | 684.8 | 730.4 |
| 6 | 729.2 | 731.7 | 733.9 | 735.8 | 737.7 | 741.0 | 745.8 | 751.2 | 800.9 | 857.9 |
| 7 | 850.6 | 853.0 | 855.2 | 857.3 | 859.4 | 862.8 | 867.4 | 873.5 | 915.1 | 950.5 |
| 8 | 972.0 | 974.5 | 976.7 | 978.7 | 980.7 | 984.9 | 990.2 | 996.2 | 1031 | 1060 |
| 9 | 1093 | 1096 | 1098 | 1100 | 1102 | 1106 | 1112 | 1117 | 1147 | 1169 |
| 10 | 1215 | 1217 | 1219 | 1222 | 1224 | 1229 | 1233 | 1239 | 1263 | 1280 |

The time used for system with *S-Net* is:

| trips | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| robots | | | | | | | | | | |
| 1 | 12.18 | 13.18 | 14.18 | 15.23 | 16.26 | 17.25 | 18.26 | 19.25 | 20.24 | 21.24 |
| 2 | 24.09 | 25.17 | 26.17 | 27.29 | 28.31 | 29.27 | 30.38 | 31.42 | 32.52 | 33.51 |
| 3 | 35.99 | 37.16 | 38.17 | 39.28 | 40.28 | 41.35 | 43.91 | 44.90 | 45.92 | 46.94 |
| 4 | 47.93 | 49.20 | 50.17 | 51.28 | 52.30 | 53.43 | 55.87 | 56.99 | 57.99 | 59.16 |
| 5 | 59.83 | 61.18 | 62.11 | 63.28 | 64.36 | 65.41 | 69.14 | 70.11 | 71.02 | 71.27 |
| 6 | 71.75 | 73.20 | 74.08 | 75.39 | 76.41 | 77.60 | 80.99 | 82.03 | 82.96 | 84.14 |
| 7 | 90.43 | 91.87 | 92.63 | 93.95 | 96.28 | 94.06 | 95.43 | 95.53 | 95.03 | 96.81 |
| 8 | 101.3 | 102.8 | 103.6 | 104.9 | 107.1 | 106.4 | 106.8 | 107.2 | 107.5 | 108.8 |
| 9 | 112.15 | 113.8 | 114.5 | 115.8 | 117.9 | 117.6 | 118.2 | 119.0 | 119.8 | 121.1 |
| 10 | 123.0 | 124.8 | 125.4 | 126.9 | 128.8 | 129.4 | 129.5 | 131.1 | 131.9 | 133.0 |

The distance traveled for system with *S-Net* is:

| trips | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| robots | | | | | | | | | | |
| 1 | 116.8 | 116.7 | 116.7 | 116.7 | 116.7 | 116.7 | 116.7 | 116.7 | 116.7 | 116.7 |
| 2 | 234.0 | 234.0 | 234.0 | 234.0 | 234.0 | 234.0 | 233.9 | 234.0 | 234.0 | 234.0 |
| 3 | 351.2 | 351.2 | 351.2 | 351.2 | 351.3 | 351.2 | 365.5 | 364.0 | 362.9 | 361.9 |
| 4 | 468.8 | 468.8 | 468.8 | 468.8 | 468.8 | 468.8 | 481.6 | 480.4 | 479.3 | 478.4 |
| 5 | 585.9 | 586.0 | 586.0 | 585.9 | 586.0 | 586.0 | 610.5 | 607.9 | 604.4 | 594.9 |
| 6 | 703.4 | 703.6 | 703.7 | 703.6 | 703.7 | 703.6 | 725.0 | 722.9 | 719.7 | 718.6 |
| 7 | 888.3 | 887.7 | 886.9 | 885.9 | 899.4 | 863.6 | 865.6 | 853.0 | 835.1 | 840.4 |
| 8 | 995.4 | 994.7 | 994.2 | 993.2 | 1005 | 983.1 | 975.9 | 965.3 | 956.2 | 954.8 |
| 9 | 1102 | 1102 | 1101 | 1100 | 1110 | 1092 | 1086 | 1078 | 1075 | 1073 |
| 10 | 1209 | 1209 | 1208 | 1207 | 1215 | 1207 | 1196 | 1194 | 1192 | 1187 |

To establish C_s , we investigated mobile robot costs and a reasonable projection for s-element costs. For a given number of robots and s-elements, these costs are fixed and the cost variation comes from the C_p term. Rather than look at particular fixed a_t and a_d , we have assumed they are equal. Separate them out vary a_t independent of a_d . Figures 5.43 and 5.44 show the percentages of times the *S-Net* out-perform the *non-S-Net*s a function of the coefficient value ($a_t = a_d$). As can be seen, for both the quadratic and linear cost function, there are thresholds below which the *non-S-Net* out-performs the *S-Net*. This indicates that for any particular implementation, a specific detailed analysis should be done to determine which is preferred.

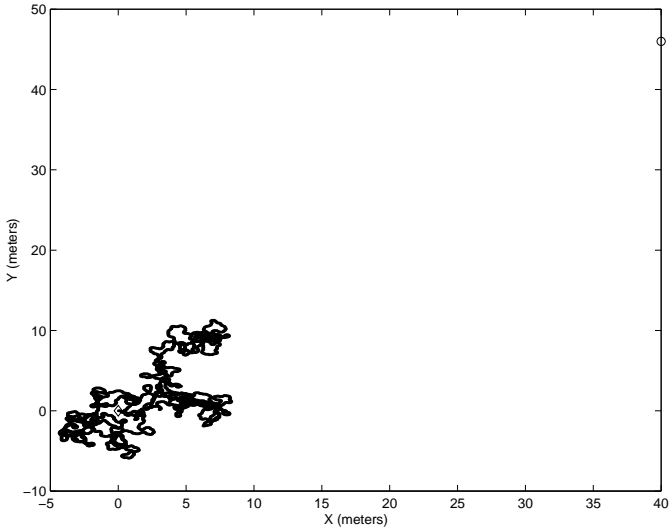


Figure 5.42. Trace of robots going back and forth without *S-Net* (2 robots, 2 round trips, noise = 0.5)

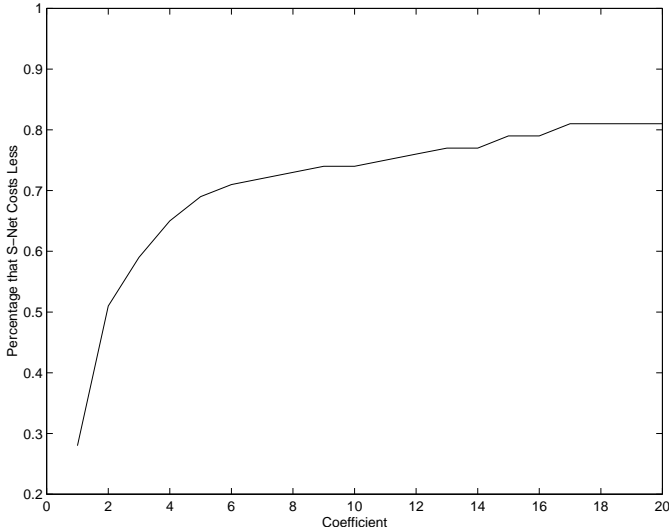


Figure 5.43. System cost comparison vs. coefficient in quadratic distribution

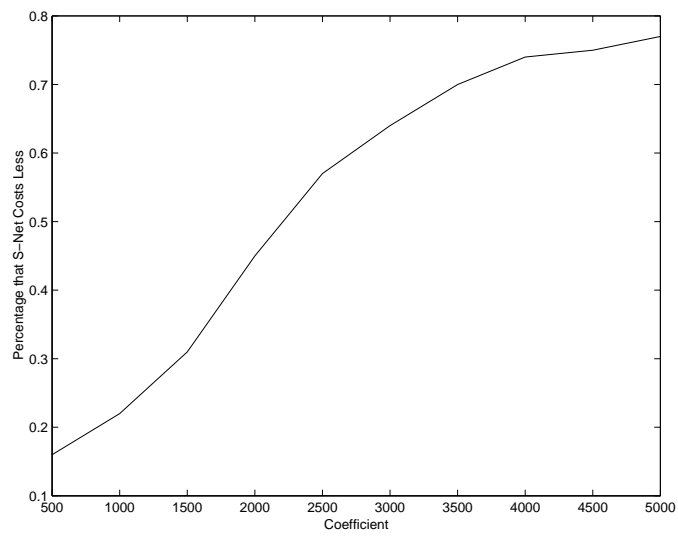


Figure 5.44. System cost comparison vs. coefficient in linear distribution

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

We have presented a formal basis for the use of a large number of spatially distributed sensors, to help multiple mobile robots cooperate and perform tasks. We developed the mobile robot model including its own local frame, primitive and high-level behaviors. The communication protocols are set up for mobile robots to get information about the environment from the s-elements in the *S-Net* system, or to communicate with other mobile robots to cooperate and prevent collision.

Algorithms are developed for the *S-Net* in order to:

- produce local frames with the sensors,
- produce useful patterns in the sensor set, and
- exploit the sensors to achieve more locally-tuned and robust robot behaviors.

Finally, the behaviors are developed for mobile robots, including:

- One mobile robot goes to a temperature source
- Multiple mobile robots surround a temperature source
- Multiple mobile robots go back and forth to a temperature source

By analyzing the measurement such as time used, distance traveled and the final distance to the temperature source, the results show that the *S-Net* system out-performs standard mobile robots (i.e., with on-board sensors), and remains robust even in the presence of heavy noise.

Future work includes:

- Physical implementation of the *S-Net* system, including the s-elements and mobile robots which exploit them.
- Optimize communication message code or layout, and study the communication errors, such as bad bits or lost message problems.
- Develop more useful patterns to help multiple mobile robots cooperation. For example, to find the shortest path regarding to time or energy.
- Develop more mobile robot behaviors that can utilize the advantage of *S-Net* system.

REFERENCES

- [1] BARES, J. E., AND WETTERGREEN, D. S. Dante ii: Technical description, results, and lessons learned. *The International Journal of Robotics Research* 18, 7 (July 1999), 621–649.
- [2] FRADEN, J. *AIP Handbook of Modern Sensors*. American Institute of Physics, New York, 1993.
- [3] HENDERSON, T. C., DEKHIL, M., MORRIS, S., CHEN, Y., AND THOMPSON, W. B. Smart sensor snow. *IEEE Conference on Intelligent Robots and Intelligent Systems* (October 1998).
- [4] H.R.EVERETT. *Sensors for Mobile Robots Theory and Application*. A K Peters, Ltd., Massachusetts, 1995.
- [5] MURRAY, J. *Mathematical Biology*. Springer-Verlag, Berlin, 1993.
- [6] SETHIAN, J. A. *Level Set Methods*. Cambridge University Press, New York, 1996.
- [7] SMITH, R., FROST, A., AND PROBERT, P. A sensor system for the navigation of an underwater vehicle. *The International Journal of Robotics Research* 18, 7 (July 1999), 697–710.
- [8] S.S.IYENGAR, L. P., AND MIN, H. *Advances in Distributed Sensor Integration*. Prentice-Hall Inc., New Jersey, 1995.
- [9] TURING, A. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B237* (1952), 37–72.