BAYESIAN APPROACH FOR DETECTION

CLASSIFICATION

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Anupama Reddy Annapureddy

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

January 2011

Fargo, North Dakota

North Dakota State University
Graduate School

Title

BAYESIAN APPROACH FOR

DETECTION CLASSIFICATION

By

ANUPAMA REDDY ANNAPUREDDY

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Annapureddy, Anupama Reddy, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, January 2011. Bayesian Approach for Detection Classification. Major Professor: Dr. Kendall E. Nygard.

The objective of this paper is to develop and test a software system that uses incomplete information from a collection of sensors to classify different objects present in a particular area with a pre-specified probability. The objects in the study are referred to as vehicles called the Bus and the Truck. Intruding vehicles move across a designated geographical area. Sensors that have been placed in that area detect vehicles and calculate probabilities that a vehicle is of a specific type conditioned on the type of vehicle that is actually detected. The goal is to determine unconditional probabilities that a given detection is of a particular type. The main idea is to find which vehicle is located at a geographical point in a designated area using the Bayesian approach to calculate the probabilities for this detection classification problem. Each sensor tries to detect the vehicle based on its sensing radius, which is nothing but the distance between the sensor and the vehicle. To test the methodology, I assumed that the probabilities vary depending on the color of the vehicles. For example, if a vehicle is red in color, it is assumed to be easier for the sensors to classify than if it is blue. The framework uses Bayesian inference to calculate the probabilities and to distinguish two types of moving vehicles. Experiments are conducted to find the number of sensors that successfully distinguish two types of moving objects with a given probability threshold. In the future the Detection Classification Model can be used to distinguish any number of objects with the mobile sensors and also some obstacles included in a designated geographical area.

ACKNOWLEDGEMENTS

My sincere thanks to Dr. Kendall E. Nygard for his supervision and outstanding support and to Dr. Tariq King, Dr. Simone Ludwig, and Dr. Limin Zhang for serving on the committee. Thanks to my family and friends who encouraged me to achieve this goal.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

1. INTRODUCTION

In the real world, we encounter different objects, such as humans, vehicles, and buildings. People can identify these different objects accurately based on the characteristics or properties they exhibit. The goal in this paper is to develop and test a software system that uses incomplete information from a collection of sensors to classify different objects present in a particular area with a pre-specified probability. The objects in the study are referred to as vehicles called the Bus and the Truck.

The sensors can detect vehicles within a prescribed range and generate probabilities using the Bayes' Theorem. "The sensor networks have the capability to answer the queries even if the information used is partially corrupt" [1]. These sensors make use of the Bayesian networks.

The Bayes' Theorem can be stated as follows:

"$P(s_i \mid e_j, z_k) = [\, p(s_i)\, p(z_k \mid e_j, s_i)] \,/\, [p(z_k \mid e_j)]$

Where s_i = State of nature (outcome) i, i = 1, 2, 3... N

e_j = Available experiment j which provides new information, j = 1, 2, 3, ..., J

z_k = Message k that can be returned from an experiment, k = 1, 2, 3, ..., K

a_l = Available course of action l which can be taken, l =1, 2, 3, ..., L

Posterior Probability that s_i will occur given experiment, e_j and message, z_k =

[(Prior Probability that s_i will occur) (Probability of the message, z_k given experiment, e_j and the true state is, s_i)]

(Probability that the message, z_k will be received given experiment, e_j)" [2]

There are two User Interfaces designed in this paper:

1. User Interface to support classifying the detections and

2. User Interface to support the calculating of the Posterior Probabilities when the Prior Probabilities are given

The User Interface to support the calculating of the Posterior Probabilities when the Prior Probabilities are given is like a foundation to the Detection Classification Model because the equations derived for the Posterior Probability calculation, generation of the Decision Tree are used in the Detection Classification Model as well.

The User Interface to support classifying the detections of the moving objects includes two vehicles, a Bus and a Truck, and 100 static sensors. The paths generated are color coded, i.e., green for one vehicle and orange for the other. Sensors try to detect the vehicle based on the distance from the vehicle and also the color of the vehicle. To model incomplete information, if the distance between the sensor and the vehicle is more, the probability of detecting the vehicle accurately is lower. Similarly, if the distance is smaller, the probability of identifying the vehicles accurately is higher. In our model, if a vehicle is red, the probability to detect and classify it accurately will be much higher than if it is blue in color.

The probabilities are generated by iterations. The iterations are carried alternatively with respect to the location where the vehicles are found. By recording these values and considering threshold values of probabilities for both vehicles, we conclude which vehicle is which with a minimum probability level.

The User Interface to support classifying the detections of the moving objects follows this algorithm:

PRIOR PROBABILITIES AT LOCATION 1 = 0.5

STEP 1: CALCULATE POSTERIOR PROBABILITIES AT LOCATION 1

STEP 2: IF (POSTERIOR PROBABILITY AT LOCATION 1 < THRESHOLD)

THEN

STEP 3: PRIOR PROBABILITIES AT LOCATION 2 = POSTERIOR

PROBABILITY AT LOCATION 1

STEP 4: CALCULATE POSTERIOR PROBABILITIES AT LOCATION 2

STEP 5: IF (POSTERIOR PROBABILITY AT LOCATION 2 < THRESHOLD)

THEN

STEP 6: PRIOR PROBABILITIES AT LOCATION 1 = POSTERIOR

PROBABILITY AT LOCATION 2

ELSE

STEP 7: IF (POSTERIOR PROBABILITY AT LOCATION 1 OR AT LOCATION

2 >= THRESHOLD)

THEN

STEP 8: STOP

ELSE

STEP 9: REPEAT STEPS 1 TO 8

The paper has the following structure. In Chapter 2, a brief review of the literature on vehicle detection based on probabilities is presented. In Chapter 3, the problem statement, the application interface, and Bayes' Theorem and its applications are described. Chapter 4 explains all the classes and the form in detail along with the problem output. The Results are given in Chapter 5, and conclusions are drawn in Chapter 6.

2. LITERATURE REVIEW

Object detection or identification algorithms are based on real-life scenarios. The vehicle detection is based on the Bayesian probabilities generated. The detection logic differs for each vehicle considered. Probabilities generated are based on the distance between the sensor and the vehicle; the colors are also taken into account. The probability to find a vehicle when near the sensor will be much higher than finding vehicles that are far from the sensor. There is also a much higher probability to find an object which is red in color than one that is blue because the red color's wavelength is much higher than blue.

The detection reports are generated in the form of Decision Trees with the possible probabilities included. The paper discusses the "Probabilistic View of Decision Trees" approach where each Tree shows the calculated Posterior Probabilities as output; the calculated average of all these Posterior Probabilities gives the final estimated Posterior Probability [3].

The detection logic for each vehicle differs based on characteristics such as color, speed, height, etc. I adapted the idea of detection of the vehicle based on color from Cuong's paper [4] which describes different kinds of approaches for road extraction with one of those approaches being the color-based approach. In the color-based approach the image sensor is designed to respond to a range of color and hence, different objects have different spectral responses. I used the logic to display different probability values based on the color of the vehicle. If the vehicle is red in color the probability generated by the sensor would be higher than the vehicle which is in blue.

Iterative Bayesian calculations are carried out to find which vehicle is placed at a geographical point in a given area. This idea is drawn from the book discussed in [5] where

there is a method described to find a frequency set using a Bayesian Network in a data-mining process.

The main idea of object detection is drawn from Schneiderman and Kanade [6] where there are Classifiers used for object detection. The Object Detectors make use of the Classifiers where there is a different detection logic defined for each kind of object detected. For instance, if the detection is a face, there are three factors taken into consideration: the front profile, right profile, and left profile poses. Similarly, if the detection is a car, there are 15 views that have to be considered, and the Classifiers are trained according to those particular viewpoints.

The idea to have some thresholds for the probabilities to find a vehicle is based on Lee and Chang's article [7]. The probabilities that reach the threshold help us track the moving objects. In Lee and Chang's article, there are some rules set based on the threshold frequencies.

The probabilities generated based on the distance from the sensor and the moving objects are taken from the concept described in Atiya's paper [8]. In that paper, there is a method called K-nearest neighbor, and also, the weights are estimated on the maximum likelihood procedure. I made use of the probabilities generated based on the distance from the sensor to the vehicle, as the distance between the sensor and the vehicle increases the probability is less and vice-versa.

3. EXTERNAL VIEW

The model described in this paper takes inspiration from Bayes' Theorem. This chapter deals with the external view of the application, and explains the look and feel of the interfaces for the Posterior Probability Calculation and Detection Classification Model in detail. Subsections 3.1, 3.2, and 3.3 deal with the problem definition, functionality, and logic behind the Posterior Probability Calculation Interface, respectively. The problem definition, functionality, and logic behind the Detection Classification Model are explained in subsections 3.5, 3.6, and 3.7, respectively.

3.1. Problem Definition With Respect to Posterior Probability Calculation

The problem is to develop a The User Interface to support the calculating of the Posterior Probabilities when the Prior Probabilities are given. For a given "n" value of Prior Probabilities, there are "n^2" Conditional Probabilities, and the corresponding "n^2" Posterior Probabilities are calculated using this interface. The user can either type in the values or can upload the pre-existing probability values for which the Posterior Probabilities are calculated. The calculated probabilities can be viewed in the form of a Tree. The Visual C# code behind the problem execution can be viewed, and the Posterior Probabilities calculated can be displayed on the interface.

Microsoft's .NET Framework is a new platform built with the Internet in mind, without ignoring the classic desktop application platform. I chose the DOTNET platform as the framework consists of the drag-drop tool to develop the window forms or the interface and the code is automatically generated behind the design. DOTNET is a collection of tools, technologies, and languages that all work together in a single environmental

framework for building and deploying robust applications. The following sections describe how the Posterior Probabilities are calculated.

3.2. Calculation of Posterior Probabilities

In this section, the functionality of the interface for the Posterior Probability calculation is described. The User Interface performs calculations for the Prior Probabilities when the values are entered. The "n" value is always greater than zero (n>0) which is the number of Prior Probability values to be considered. We can either load the corresponding text file with the naming convention "PriorProbability_s.txt" if there are some pre-existing values of Prior Probabilities, or we can re-enter or modify the existing values. For every "n" value of Prior Probabilities, there should be n^2 values entered in the text file with the naming convention "priorprobability _zs.txt". The values are automatically generated in a text file with the naming convention "priorprobability _z.txt". After entering all the necessary values and clicking the "Calculate Posterior Probability" button, the event would generate the calculated n^2 Posterior Probability values in a text file with the naming convention "PosteriorProbability.txt". The Decision Tree can be viewed when clicking the "View Tree" button. The following section shows an instance of how the interface works and also shows the Tree Form of the generated probabilities.

3.3. The Application User Interface for Posterior Probability Calculation

This section describes how the User Interface application for probability calculation works. The User Interface to calculate the Posterior Probability follows these steps:

1. Enter a value for n (n>0): n
2. Enter the Prior Probability values: $p(s) = s_1, s_2, s_3, \ldots, s_n$

3. Enter n^2 values for the Conditional Probabilities: $p(z_k|e_j, s_i)$
4. The values for the probability that the message, z_k, will be received given the experiment, e_j, $p(z_k|e_j)$, are automatically generated (joint probabilities)
5. The Posterior Probability values, "$P(s_i|e_j, z_k) = [\, p(s_i)\, p(z_k|e_j, s_i)\,]/[\, p(z_k|e_j)\,]$" [2], are calculated
6. The Decision Tree can be drawn using these probabilities
7. The calculated probabilities are displayed on the interface

Figure 3.1 shows the Graphical User Interface to calculate Posterior Probabilities when the Prior Probabilities are given. The input for the Prior Probabilities is either entered manually, or the pre-existing values entered in the corresponding text files can be used.

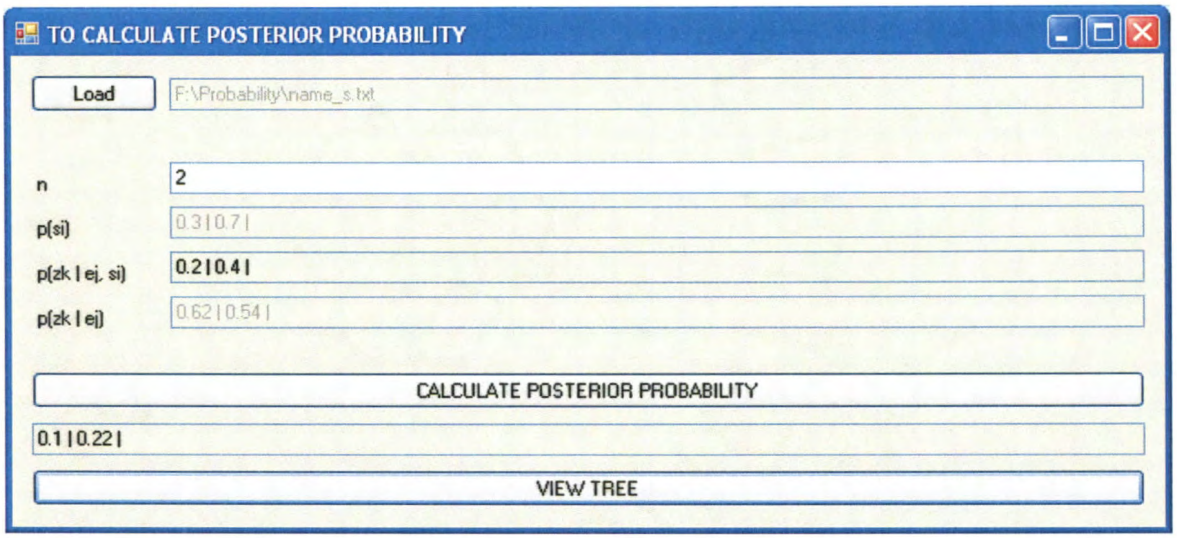

Figure 3.1. Graphical User Interface to Calculate Posterior Probability

The Prior Probabilities, $p(s) = s_1, s_2, s_3 \ldots s_n$, are given as input in a text file "priorprobability_s.txt": $s_1 = p(s_1)$, $s_2 = p(s_2)$, $s_3 = p(s_3)$, …, $s_n = p(s_n)$.

The Conditional Probabilities, $p(z_k|e_j, s_i)$, are n^2 values. These values are given as inputs in a text file "priorprobability _zs.txt":

$p(z_1|e_1, s_1)$, $p(z_2|e_1, s_1)$, $p(z_3|e_1, s_1)$, …, $p(z_n|e_1, s_1)$

$p(z_1|e_1, s_2), p(z_2|e_1, s_2), p(z_3|e_1, s_2), \ldots, p(z_n|e_1, s_2)$

$p(z_1|e_1, s_3), p(z_2|e_1, s_3), p(z_3|e_1, s_3), \ldots, p(z_n|e_1, s_3)$

. . . , ..., .

. . . , ..., .

$p(z_1|e_1, s_n), p(z_2|e_1, s_n), p(z_3|e_1, s_n), \ldots, p(z_n|e_1, s_n)$

The values in each column should add to 1.

The z values are necessary to calculate the Posterior Probabilities, and for this reason, the Joint Probabilities, $p(z_k|e_j) = z_1, z_2, z_3, \ldots, z_n$, are automatically generated in a text file "priorprobability_z.txt". The following equations show how these probabilities are being calculated:

$z_1 = p(z_1|e_1) = p(s_1)*p(z_1|e_1, s_1) + p(s_2)*p(z_1|e_1,s_2) + p(s_3)*p(z_1|e_1, s_3) + \ldots + p(s_n)*p(z_1|e_1, s_n)$

$z_2 = p(z_2|e_1) = p(s_1)*p(z_2|e_1, s_1) + p(s_2)*p(z_2|e_1,s_2) + p(s_3)*p(z_2|e_1, s_3) + \ldots + p(s_n)*p(z_2|e_1, s_n)$

$z_3 = p(z_3|e_1) = p(s_1)*p(z_3|e_1, s_1) + p(s_2)*p(z_3|e_1,s_2) + p(s_3)*p(z_3|e_1, s_3) + \ldots + p(s_n)*p(z_3|e_1, s_n)$

.

.

$z_n = p(z_n|e_1) = p(s_1)*p(z_n|e_1, s_1) + p(s_2)*p(z_n|e_1,s_2) + p(s_3)*p(z_n|e_1, s_3) + \ldots + p(s_n)*p(z_n|e_1, s_n)$

Posterior Probability is calculated using the formula:

"$P(s_i|e_j, z_k) = [\, p(s_i)\, p(z_k|e_j, s_i)\,]/ [\, p(z_k|e_j)\,]$" [2]

The Posterior Probability values are printed in the text file, "PosteriorProbability.txt". The following equations show how the Posterior Probabilities are calculated:

$P(s_1|e_1, z_1) = [\, p(s_1)\, p(z_1|e_1, s_1)\,]/ [\, p(z_1|e_1)\,]$, $P(s_1|e_1, z_2) = [\, p(s_1)\, p(z_2|e_1, s_1)\,]/ [\, p(z_2|e_1)\,]$, ...,

$P(s_1|e_1, z_n) = [\ p(s_1)\ p(z_n|e_1, s_1)\ ]/\ [\ p(z_n|e_1)\ ]$

$P(s_2|e_1, z_1) = [\ p(s_2)\ p(z_1|e_1, s_2)\ ]/\ [\ p(z_1|e_1)\ ]$, $P(s_2|e_1, z_2) = [\ p(s_2)\ p(z_2|e_1, s_2)\ ]/\ [\ p(z_2|e_1)\ ]$,

…,

$P(s_2|e_1, z_n) = [\ p(s_2)\ p(z_n|e_1, s_2)\ ]/\ [\ p(z_n|e_1)\ ]$

$P(s_3|e_1, z_1) = [\ p(s_3)\ p(z_1|e_1, s_3)\ ]/\ [\ p(z_1|e_1)\ ]$, $P(s_3|e_1, z_2) = [\ p(s_3)\ p(z_2|e_1, s_3)\ ]/\ [\ p(z_2|e_1)\ ]$,

…,

$P(s_3|e_1, z_n) = [\ p(s_3)\ p(z_n|e_1, s_3)\ ]/\ [\ p(z_n|e_1)\ ]$

.

.

$P(s_n|e_1, z_1) = [\ p(s_n)\ p(z_1|e_1, s_n)\ ]/\ [\ p(z_1|e_1)\ ]$, $P(s_n|e_1, z_2) = [\ p(s_n)\ p(z_2|e_1, s_n)\ ]/\ [\ p(z_2|e_1)\ ]$,

…,

$P(s_n|e_1, z_n) = [\ p(s_n)\ p(z_n|e_1, s_n)\ ]/\ [\ p(z_n|e_1)\ ]$

The values in each column should add to 1.

Figure 3.2 shows the interface when n equals 3. Figure 3.3 shows the text file "priorprobability_s" where the Prior Probabilities can be entered. All these values should be equal to 1.

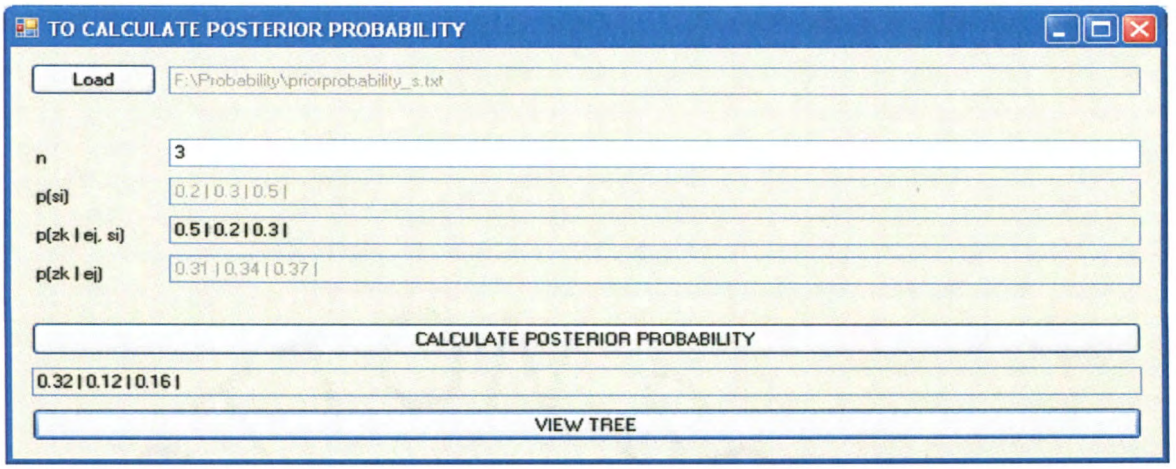

Figure 3.2. Graphical User Interface to Calculate Posterior Probability When n Equals 3

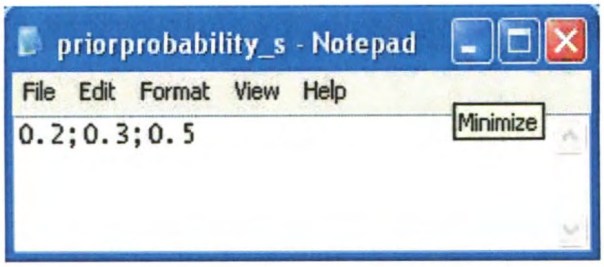

Figure 3.3. Entering s Equals Three Values in priorprobability_s.txt

The values which are shown in Figure 3.3 are given as input for the Posterior Probability calculation. These values always sum to 1. Figure 3.4 shows the text file "priorprobability_zs" where n^2 values are entered; i.e., when n equals 3, there must be nine values entered into this file, and the values in each column should add to 1.

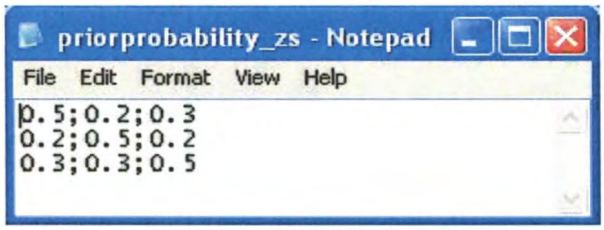

Figure 3.4. Entering n^2 Which Equals Nine Values in priorprobability_zs.txt

The values entered in the "priorprobability_zs.txt" file shown in Figure 3.4 are also given as input for the corresponding s values. For every n Prior Probability values (s values) entered, there are n^2 Conditional Probability values to be entered as input in the "priorprobability_zs.txt" file. The values entered in each column should always sum to 1. Figure 3.5 shows the text file "priorprobability _z.txt" where the values are generated automatically.

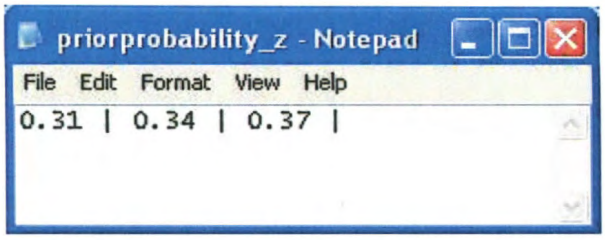

Figure 3.5. Probabilities Generated in priorprobability_z.txt

The values shown in Figure 3.5 are the calculated values for Joint Probabilities and are generated in the "priorprobability_z.txt" text file. When added, these values will always equal 1. Figure 3.6 shows the final calculated Posterior Probabilities for the given Prior Probabilities.

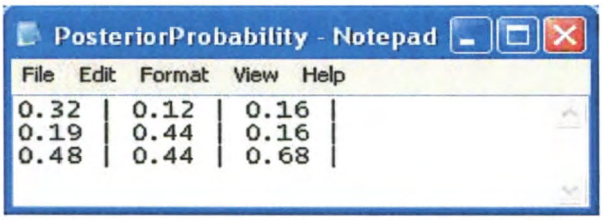

Figure 3.6. Posterior Probabilities Generated for n Equals 3

The Posterior Probabilities calculated with all the necessary inputs given are displayed in a text file called "PosteriorProbability.txt" as shown in Figure 3.6. The values displayed in each column should always equal 1. The Decision Tree can be drawn using these probabilities. The example in Figure 3.7 shows the Decision Tree when n equals 3.

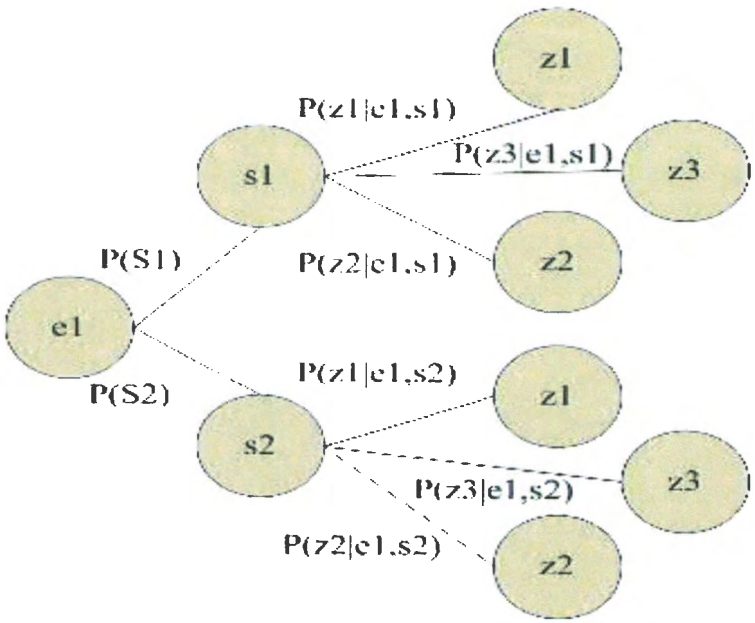

Figure 3.7. Example of Tree Diagram When n Equals 3

The Tree view generated from the calculations for n equals 3 is shown in Figure 3.8 with the Prior and Posterior Probability values in Tree format.

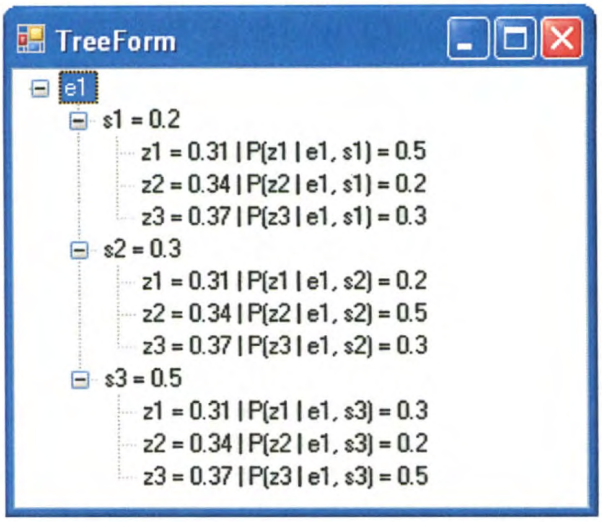

Figure 3.8. Tree View Generated When n Equals 3

The Tree form in Figure 3.8 shows the Tree format of the Posterior Probabilities calculated with the Prior Probabilities given as input. Since, the given input is 3 for the Prior Probabilities there are 9 corresponding Conditional Probabilities and Posterior Probabilities produced.

3.4. The Detection Classification System Design

The Detection Classification System performs motion calculations for the vehicles: Bus and Truck. The major components in the system are the vehicles, the environment, and the communication mechanisms. Vehicles are the dependent entities in the system that has the capability to move according to the speed parameters set.

3.5. Problem Statement With Respect to Detection Classification

The problem is to develop a Detection Classification System which has 100 sensors and 2 moving vehicles, i.e., Bus and Truck. The sensors try to find the probability to detect a vehicle, and whenever there is detection, sensors generate a report. The vehicles move randomly, and for each vehicle move, there is a Tree Diagram set up with the probabilities put in whenever the Posterior Probabilities reaches the assumed thresholds for each

vehicle. The paths generated by the vehicles are green and orange in color. The whole idea is to find which vehicle is at a particular location in the considered area even when there is little or no perfect information available. The vehicle detection is found out using the Bayes' Theorem.

3.6. The Application User Interface for Detection Classification

This section describes, in detail, the layout of the application User Interface for the Detection Classification System. Figure 3.9 shows the User Interface of the Visual C#-based implementation of the artificial Detection Classification System. The grid panel can be divided anywhere from a 20*20 matrix of squares to a 50*50 matrix of squares that contain 100 sensors indicated by an asterisk (*) at different locations. The interface displays the location, i.e., (x, y) coordinates of the sensor, when the cursor points and when the sensors are numbered from 1 to 100. A default grid is loaded at startup with only the "Start" button active. The grid can be reset to default using the "Reset" button. The "Pause" button is used to pause the simulation. In pause mode, the button changes to "Resume", and the simulation can be continued by using the "Resume" button. The "Stop" button stops the simulation, making it possible to input necessary parameter changes followed by restarting the simulation.

The boxes on the left control the execution include the following items:

1. Grid is a component which allows us to select the size of the grid, ranging from 20*20 to 50*50, with X and Y being the axis and the check box "Show Grid." This component also consists of the major buttons "Change Grid Size and Clean," "Populate Sensors," and "Populate Bus/Truck"

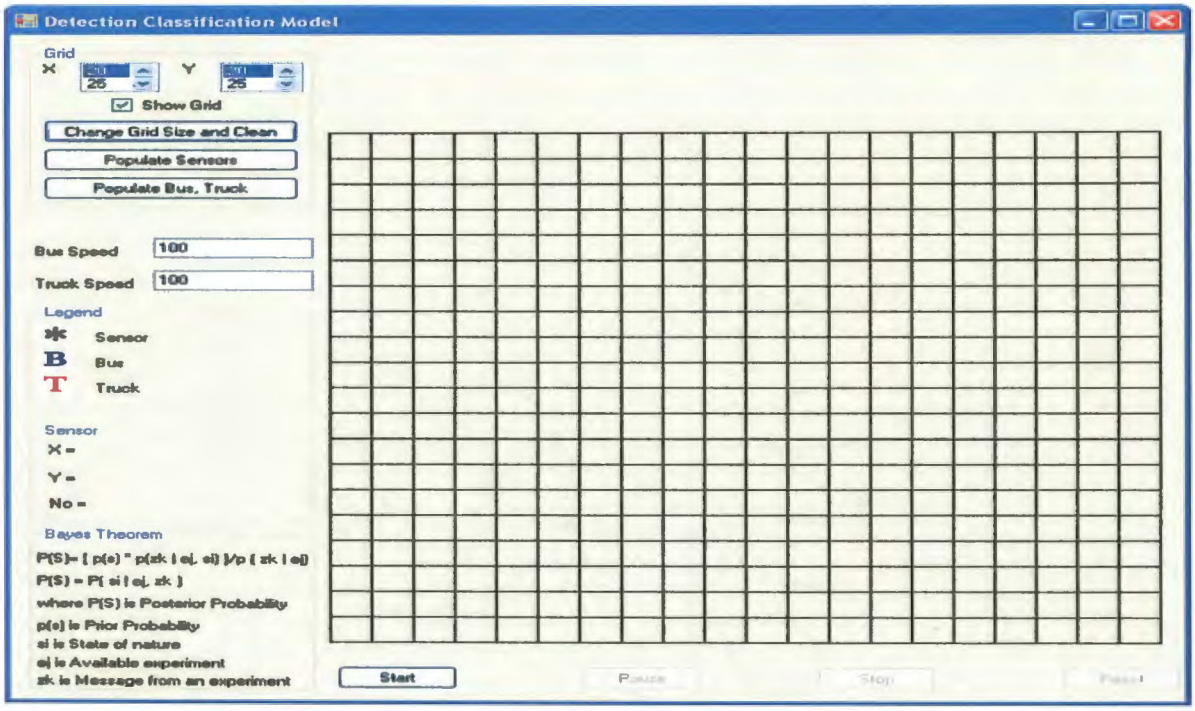

Figure 3.9. Initial Load of the Interface

2. Bus Speed indicates the speed the Bus moves per 100 seconds
3. Truck Speed indicates the speed the Truck moves per 100 seconds
4. Legend is a brief description accompanying an illustration, indicating which are the sensors and the vehicles
5. Sensor grid displays the X, Y coordinates and the index number for the sensor when the mouse hovers over a sensor in the interface
6. Bayes' Theorem shows the logic behind the application developed

The Detection Classification Interface follows these steps:

1. Populate the sensors
2. Populate the vehicles: Bus and Truck
3. Specify the speed parameters
4. Start the simulation
5. The vehicles move randomly

6. At the initial load of the vehicles, the probabilities are generated by the sensors based on distance and color
7. The sensors try to detect the vehicles
8. A Decision Tree is generated in the form of a report for each detection by a sensor

Figure 3.10 shows the User Interface for a Detection Classification System when the simulation is complete. The simulation is run with the vehicles, and the sensors populated on the grid. When a specific speed is set for both the vehicles and the "Start" button is clicked, the vehicles tend to move. For each vehicle move with the "B" image, the path is shown in green, and for each vehicle move with the "T" image, the path is shown in orange. The path generated is shown in the form of arrows to track the direction of the moving vehicle.

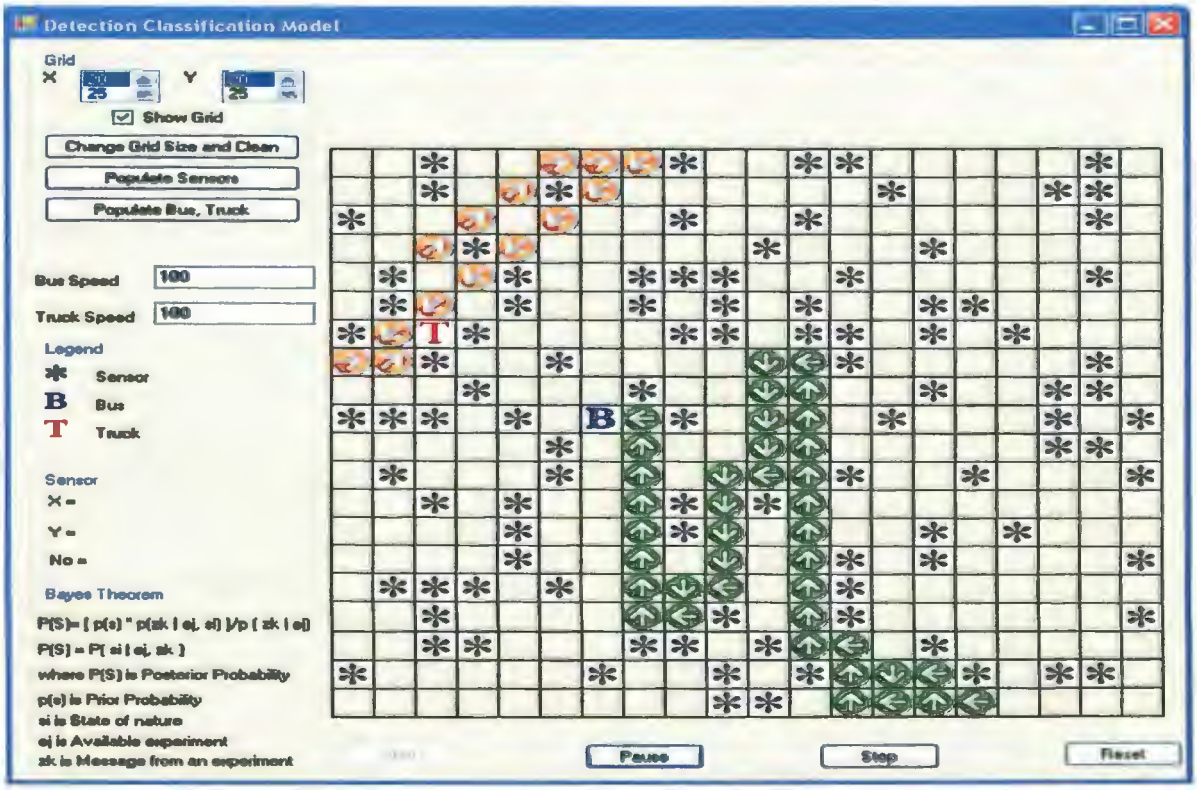

Figure 3.10. Complete Simulation

Figures 3.9 and 3.10 depict the initial and final phases of the application. The following subsection describes the logic behind the object detection.

3.7. Logic Behind the Detection Classification

This section deals with the logic used for the Detection Classification System. The environment for the Detection Classification Model is a finite, rectangular grid and consists of two moving objects in the presence of 100 static sensors. The objects can move to any of the eight neighboring cells which are not occupied by sensors. Even before the simulation starts, the objects that are present are detected by the sensors. The sensor closest to one of the vehicles is taken into consideration, and the Conditional Probabilities are calculated based on the distance and color of the vehicle; the Conditional Probabilities would further help us find the Posterior Probabilities. Upon finding the Posterior Probabilities, the Posterior Probabilities are assumed to be the Prior Probabilities for the other vehicle; hence, the Conditional Probabilities are calculated in a similar method as that mentioned earlier, and also, the Posterior Probabilities are generated. The sensors have a sensing radius by which they produce the probabilities to find a Bus/Truck based on the distance and color of the vehicle.

For instance, if the sensing radius is 9 grid squares and if the vehicle detected is a Bus at a distance, d = 9 grids from the sensor, then the probability, p, would be 0.1; d = 9 grid squares, p = 0.1; d = 8 grid squares, p = 0.2; d = 7 grid squares, p = 0.3; d = 6 grid squares, p = 0.4; d = 5 grid squares, p = 0.5; d = 4 grid squares, p = 0.6; d = 3 grid squares, p = 0.7; d = 2 grid squares, p = 0.8; d = 1 grid square, p = 0.9; and if the detected vehicle is a red Truck at a distance, d = 7-9 grid squares from the sensor, then the probability, p, would be 0.75; d = 4-6 grid squares, p = 0.85; and d = 1-3 grid squares, p = 0.95.

Considering the possible Prior Probabilities for the States of Nature for the vehicle to be a Bus, s_1, and for the vehicle to be a Truck, s_2, to be mutually equal events, i.e., $s_1 = s_2 = 0.5$, the Posterior Probabilities are calculated. For instance, if the sensor 2 cells away from the vehicle detects the vehicle to be a Bus if indeed it is a Bus, then the probability, p $(z_{11}| e_1, s_1)$, is 0.8, and the probability to detect the vehicle as a Truck if indeed it is a Bus, p $(z_{12}| e_1, s_1)$, will be 1-0.8 = 0.2. If the sensor detects the vehicle as a Truck if it is indeed a Truck, then the probability, p $(z_{22}| e_1, s_2)$, is 0.95, and the probability to detect the vehicle as a Bus if it is indeed a Truck, p $(z_{21}| e_1, s_2)$, will be 1-0.95 = 0.05.

The Joint Probabilities can be calculated, i.e., "p $(z_k |e_j)$ = (Prob$\{s_i\}$* Prob$\{z_k | e_j, s_i\}$)" [2].

$p(z_1) = p(s_1)^* p(z_{11}| e_1, s_1) + p(s_2) * p(z_{21}| e_1, s_2)$

$p(z_2) = p(s_1)^* p(z_{12}| e_1, s_1) + p(s_2) * p(z_{22}| e_1, s_2)$. Where $p(z_{12}| e_1, s_1) = 1 - p(z_{11}| e_1, s_1)$ and $p(z_{21}| e_1, s_2) = 1 - p(z_{22}| e_1, s_2)$ and the Posterior Probability can be calculated using the formula

"$P(s_i | e_j, z_k)$, $P(S) = [p(s_i)^* p(z_k | e_j, s_i)] / p(z_k | e_j)$" [2]

Whenever a vehicle is detected by a sensor, a Decision Tree is generated. The Decision Tree can be drawn using these probabilities. The Decision Tree is considered as a form of report for the object detection. The report generated is stored in a text file.

Figure 3.11 shows an example of the Decision Tree for n equals 2. Figure 3.12 depicts the Decision Tree generated from the probabilities calculated with the entire Prior, Joint, and Posterior Probability values included.

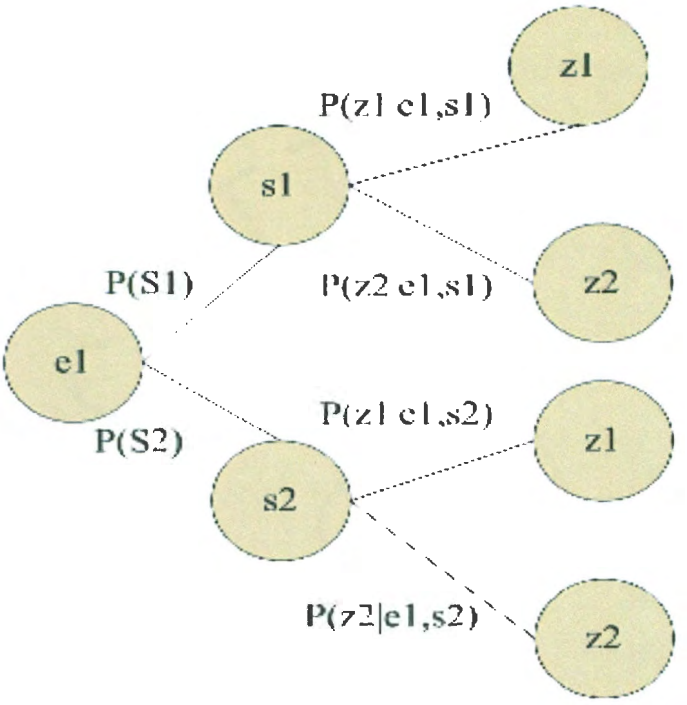

Figure 3.11. Example of Tree Diagram When n Equals 2

Figure 3.11 displays how a Tree Diagram can be drawn using the posterior and Prior Probabilities for an experiment where n is given as 2. The Tree keeps expanding as the number of inputs increases.

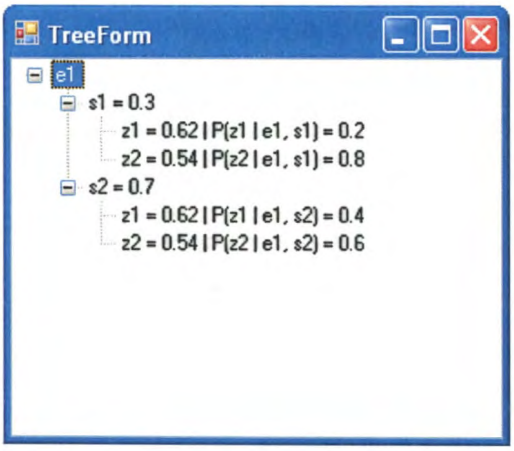

Figure 3.12. Tree View Generated When n Equals 2

The Tree view shown in Figure 3.12 is the form of the output generated with all the probabilities calculated and the same values put in the Tree format. Figure 3.13 shows the

interface when the complete paths for the vehicles have been generated with the green-colored path assuming the vehicle to be a Bus and the orange-colored path assuming the vehicle to be a Truck.

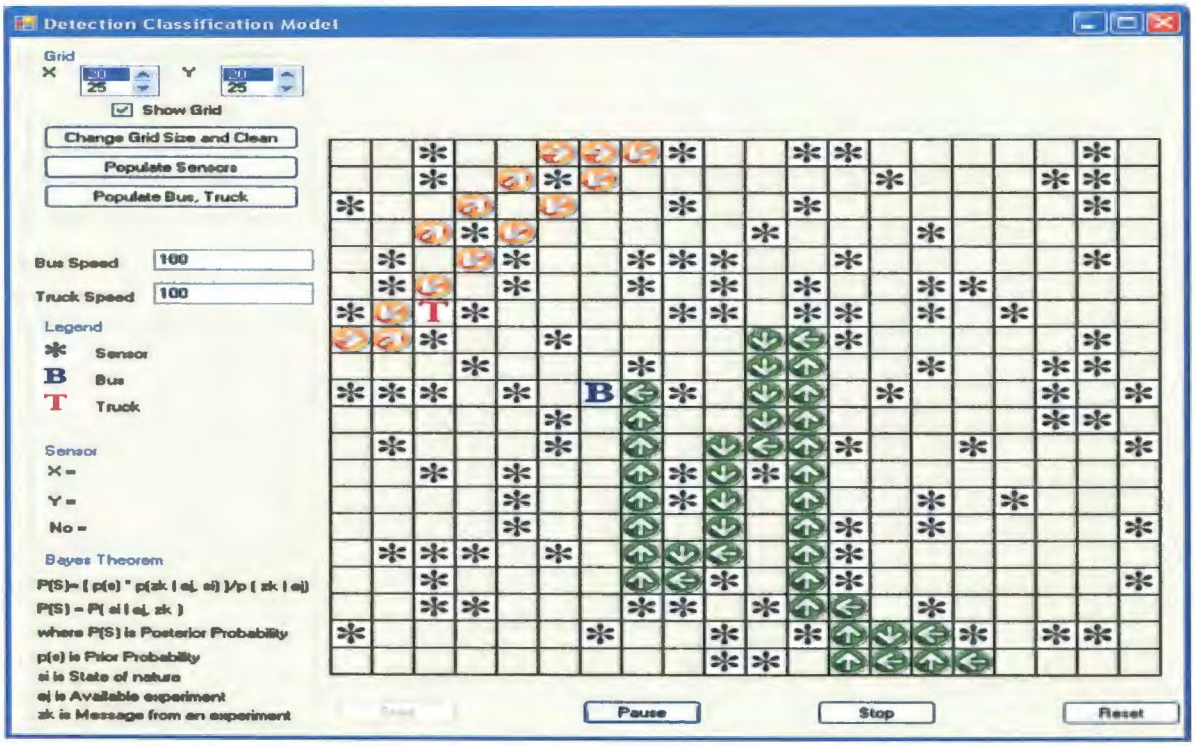

Figure 3.13. Interface With Complete Paths Generated

The Posterior Probabilities generated for the first location, i.e., location 1, where the vehicle is detected by the closest sensor are considered as Prior Probabilities for the next location where the vehicle is detected by the sensor closest to location 2. There are some thresholds considered for each vehicle. When the produced probabilities appear in the thresholds, the iteration is stopped, or this will be an alternative chain process.

4. INTERNAL VIEW

The environment for the Detection Classification Model is a finite, rectangular grid and consists of two moving objects in the presence of 100 static sensors. The objects can move to any of the eight neighboring cells which are not occupied by sensors. Even before the simulation starts, the objects that are present are detected by the sensors. The sensor closest to one of the vehicles is taken into consideration, and the Conditional Probabilities are calculated based on the distance and color of the vehicle; the Conditional Probabilities would further help us find the Posterior Probabilities. Upon finding the Posterior Probabilities, the Posterior Probabilities are assumed to be the Prior Probabilities for the other vehicle; hence, the Conditional Probabilities are calculated in a similar method as that mentioned earlier, and also, the Posterior Probabilities are generated. The goal is to find which object is detected at a particular location in the environment considered. The task of detecting the vehicle or the moving object can be interpreted as a continuous, alternating iterative process for each vehicle.

This chapter describes the sample problem for the interfaces, Posterior Probability Calculation, and the Detection Classification Model with which we worked. The chapter also walks through the class relationships and the key implementation code.

4.1. Internal View of the Posterior Probability Calculation Interface

This section gives an internal view of the interface for Posterior Probability calculation. In this chapter, there are some key classes which are used for the probability calculation. Each class consists of some important methods involved in the Posterior Probability calculation.

4.1.1. Calculation Class

This class contains the methods necessary to calculate the Posterior Probability when the Prior Probabilities are provided using the Bayes' Theorem. The following subsections give a brief description of the methods used in the calculation of the Posterior Probability.

4.1.1.1. Calculateposteriorprobability Method

The calculatePosteriorProbability method uses the formula to determine the Posterior Probability using a formula derived from the Bayes' Theorem.

```
public void calculatePosteriorProbability()
    {
      for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
          post_p[i, j] = Math.Round(p_si[i] * p_zk_si[i, j] / p_zk[j],2);

    }
```

4.1.1.2. Writedata Method

The writeData method writes the calculated Posterior Probabilities to a text file called "PosteriorProbability.txt".

```
public void writeData()
    {
      StreamWriter writer = new StreamWriter("PosteriorProbability.txt");

      string[] value = new string[n];
      for (int s = 0; s < n; s++) value[s] = "";

      for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
          value[i] += post_p[i, j] + " | ";

      for (int w = 0; w < n; w++)
        writer.WriteLine(value[w]);

      writer.Close();
    }
```

4.1.2. Windows Form

The Form class contains all the variables and controls necessary for the calculation of Posterior Probability. It represents a window within the application where we can add controls commonly associated with windows, such as a button, a text box, a check box, or a radio button, and set properties to interact with the controls by altering their appearance and behavior. The form could include dialog boxes as well as Multiple Document Interface (MDI) client and parent windows. The programming logic to form, i.e., the code behind the form, is added to the class, derived from the form, which is created when the User Interface for the application is designed. The event handlers can also be created in the class. The event-handling method is registered in the application when an event is created such as when the user clicks a button, i.e., when the control changes state.

4.1.2.1. Calculation Form

This section describes the variables and controls required for the Calculation form.

```
calculation calc;
string s_path;
string z_s_path;
string z_path;

public Form1()
{
   InitializeComponent();
   calc = new Calculation();
}
```

4.1.2.2. Controls in the Calculation Form

This section deals with all the controls present in the Posterior Probability Calculation Form. The buttons used in the application are Load, Calculate Posterior Probability, and View Tree. The Loadbutton_Click Event validates the input provided in the *_s.txt file and shows the input in the text box provided for the p(si) field. The event

also validates the input provided in "priorporbability_zs.txt" and "priorporbability_z.txt", and shows the input values in the text boxes provided for p(zk|ej, sk) and p(zk|ej).

The CalculatePosteriorProbabilitybutton_Click Event takes all the values provided in the text boxes and calculates the Posterior Probability for the given values. The process displays the values in the text box provided below the "Calculate Posterior Probability" button.

The View Treebutton_Click Event considers all the values provided and calculated. It generates a Tree View of the probabilities for an easy decision.

4.1.2.3. Treeform

This class contains the methods necessary to build a Decision Tree when all the necessary probabilities are calculated.

```
private void buildTree()
    {
      TreeNode node = new TreeNode("e1");
      TreeView1.Nodes.Add(node);

      for (int i = 0; i < calc.getN(); i++)
      {
        TreeView1.Nodes[0].Nodes.Add("s" + (i + 1) + " = " +
Convert.ToString(calc.get_p_si()[i]));

        for (int j = 0; j < calc.getN(); j++) TreeView1.Nodes[0].Nodes[i].Nodes.Add("z"
+ (j + 1) + " = " + Convert.ToString(calc.get_p_zk()[j])+" |
P(z"+(j+1)+"|e1,s"+(i+1)+")="+Convert.ToString(calc.get_p_zk_si()[j,i])+")");
      }
 TreeView1.Refresh();
    }
```

4.2. Internal View of the Detection Classification System

The environment for the Detection Classification Model is a finite, rectangular grid world and consists of a Bus, a Truck, and 100 static sensors. Vehicles move randomly, and the sensors calculate the probability of finding vehicles. From a given square, a vehicle

may move to any of the eight neighbor squares. The goal is to detect vehicles when the vehicles do not collide and when there is no complete information provided by the sensors.

4.2.1. World Class

This class contains the methods necessary to find the detections of a Bus and Truck from one square to another using the algorithm based on the Bayes' Theorem. The class determines the probable detection of the vehicle to be a Bus or a Truck.

Probabilities are considered based on the distance between the sensor and the vehicle:

```
public void calculateNewPosterior(int move_no)

{
  int move = move_no + 1;
  Zone[] z = FindNearBus();
  int bi = -1; int bj = -1;
  int ti = -1; int tj = -1;
  for (int i = 0; i < this.Width; i++)
  {
    for (int j = 0; j < this.Height; j++)
    {
      if (zone[i, j].isBus())
      { bi = i; bj = j; }
      if (zone[i, j].isTruck())
      { ti = i; tj = j; }
    }
  }
  bool foundfirst = false;
  bool foundsec = false;
  Vehicle firstVeh = Vehicle.Bus;
  Vehicle secVeh = Vehicle.Bus;
  double pfirstfound = 0;
  double psecondfound = 0;

  if (z.Length > 0)
  {
    Zone zz = z[0];
    double distBus = this.calculate_distance(zz.getX(), zz.getY(), bi, bj);
    double distTruck = this.calculate_distance(zz.getX(), zz.getY(), ti, tj);
    double pb = 1 - 0.1 * distBus;
```

```
double pt = 0;
if (distTruck >= 1 && distTruck <= 3) pt = 0.95;
if (distTruck >= 4 && distTruck <= 6) pt = 0.9;
if (distTruck >= 7 && distTruck <= 9) pt = 0.85;

//conditional
double pz11e1s1 = pb;
double pz12e1s1 = 1 - pb;
double pz21e1s2 = 1 - pt;
double pz22e1s2 = pt;

//prior
double ps1 = 0.5;
double ps2 = 0.5;

double pz1e1 = ps1 * pz11e1s1 + ps2 * pz21e1s2;
double pz2e1 = ps1 * pz12e1s1 + ps2 * pz22e1s2;

//posterior
double npz11e1s1 = (ps1 * pz11e1s1) / (pz1e1); //P(s1|e1,z1)
double npz12e1s1 = (ps1 * pz12e1s1) / pz2e1; //P(s1|e1,z2)
double npz21e1s2 = (ps2 * pz21e1s2) / (pz1e1); //P(s2|e1,z1)
double npz22e1s2 = (ps2 * pz22e1s2) / pz2e1; //P(s2|e1,z2)
```

4.2.2. Windows Form

The Form class contains all the variables and controls necessary for the calculation of the Posterior Probability. It represents a window within the application where the developer adds controls commonly associated with windows, such as a button, a text box, a check box, or a radio button, and sets the properties to interact with the controls by altering their appearance and behavior. The form could include dialog boxes, and Multiple Document Interface (MDI) client and parent windows. The programming logic to form, i.e., the code behind the form, is added to the class, derived from the form, which is created when the User Interface for the application is designed. The event handlers can also be created in the class. The event handling method is registered in the application when an event is created such as when the user clicks a button, i.e., when the control changes state.

4.2.2.1. Detection Classification Form

This section contains all the variables and controls responsible for detecting the vehicles. It represents a window within an application.

```
public Form1()
{
  InitializeComponent();
  this.SetStyle(ControlStyles.DoubleBuffer | ControlStyles.UserPaint |
  ControlStyles.AllPaintingInWmPaint, true);
  this.UpdateStyles();
  this.DoubleBuffered = true;
  this.bac = new SolidBrush(this.BackColor);
  this.MouseMove += new MouseEventHandler(gridPanel_MouseMove);

  listBox1.SetSelected(0, true);
  listBox2.SetSelected(0, true);
  grid_ON = true;
  move_no = 0;

  x_grid = Int32.Parse(listBox1.SelectedItem.ToString());
  y_grid = Int32.Parse(listBox2.SelectedItem.ToString());

  int x_temp = gridPanel.Size.Width / x_grid;
  int y_temp = gridPanel.Size.Height / y_grid;

  if (x_temp < y_temp) rectangle_size = x_temp;
  else rectangle_size = y_temp;

  world = new World(x_grid, y_grid, rectangle_size);
  gridPanel.Hide();
}
```

4.2.2.2. Controls in the Detection Classification Form

This section describes the controls present in the Detection Classification Form.

1. Buttons: The buttons used in the application are as follows: "Change grid size and clean," "Populate Sensors," "Populate Bus, Truck," "Start," "Pause," "Stop," "Reset," and "Change Grid Size and Clean" which allows for a change in the grid size and cleans the previously populated grid

The "Populate Sensors" and "Populate Bus, Truck" buttons are active before the grid is loaded with the sensors and the vehicles and, after you stop or reset the application by pressing the "Start" and the "Reset" buttons, respectively. The "Populate Sensors" and "Populate Bus, Truck" buttons are inactive when the application is either running or paused.

The "Start" button revalidates the input of the text boxes and makes sure that the sensors and vehicles are placed in the grid. Then, the vehicles start to move randomly. Some control availability is changed to prevent misuse. The Timer starts so that the simulation can begin.

The "Pause/Resume" button stops/start the Timer, depending on the situation, so that the simulation can be paused; upon clicking on the "Pause" button, the timer stops and changes the button's text to "Resume." Otherwise, clicking the "Resume" button starts the timer again, changes the text to "Pause," and continues the vehicular movement.

```
if (button_Pause.Text == "Pause")
        {
          timer_Bus.Stop();
          timer_Truck.Stop();
          button_Pause.Text = "Resume";
        }
else
        {

          timer_Bus.Start();
          timer_Truck.Start();
          button_Pause.Text = "Pause";
        }
```

The "Stop" button stops the Timer and disables some controls available to prevent misuse. Upon clicking the "Stop" button, the input text boxes that were read-only are activated and can be used to change the grid dimensions and speed parameters. The notable

difference between the "Pause/Resume" and "Stop" button is that the "Pause/Resume" button pauses the simulation and starts it from the same point where it was stopped, whereas the "Stop" button stops the simulation and starts it all over again.

```
private void button_Stop_Click(object sender, EventArgs e)
    {
      timer_Bus.Stop();
      timer_Truck.Stop();
      Button_ChangeGrid.Enabled = true;
      Button_PopulateSensors.Enabled = true;
      Button_PopulateTruck.Enabled = true;
      button_Reset.Enabled = true;
      button_RefreshDecisionTree.Enabled = true;
      world.calculate_distances();
      this.button1.Enabled = true;

    }
```

The "Reset" button resets the simulation by calling the "StopButton_Click()" method to stop the simulation. "Reset" button enables the "Populate Sensors, Populate Bus and Truck_Click()" method to populate the grid with the sensors and vehicles.

```
{
      button_Reset.Enabled = false;
      button_Stop_Click(sender, e);
      button_Start.Enabled = true;
      button_Pause.Text = "Pause";
      Button_ChangeGrid_Click(sender, e);
      TreeView1.Nodes.Clear();
    }
```

The "Decision Tree" button is used to view the Tree of probabilities generated for each move.

2. Text boxes: The text boxes used in the application are "Truck Speed" and "Bus Speed." The input value for a text box must be an integer value
3. Check boxes: The check box used in the application is "Show Grid." The "Show Grid" check box lets the panel paint if "Show Grid" is checked. When there is a change, it repaints itself to reflect the change

```
{
        if (grid_ON == true) grid_ON = false;
        else grid_ON = true;

        gridPanel.Invalidate();
        this.Invalidate();
    }
```

4.3. Modifying the Simulation

To modify the code, we should use the Microsoft Visual Studio 2010 Environment or later and the Microsoft .NET Framework 2.0. The code has comments indicating which variables for the Bus, Truck, sensors, and paths can easily be changed.

4.4. Output

Simulations are run to check the specifications provided for the vehicular movement scenarios. Figure 4.1 depicts the initial scenario when the grid is loaded with vehicles and the sensors are placed on it. The asterisks (*) represent the sensors; the letter "B" represents the Bus; and the letter "T" represents the Truck.

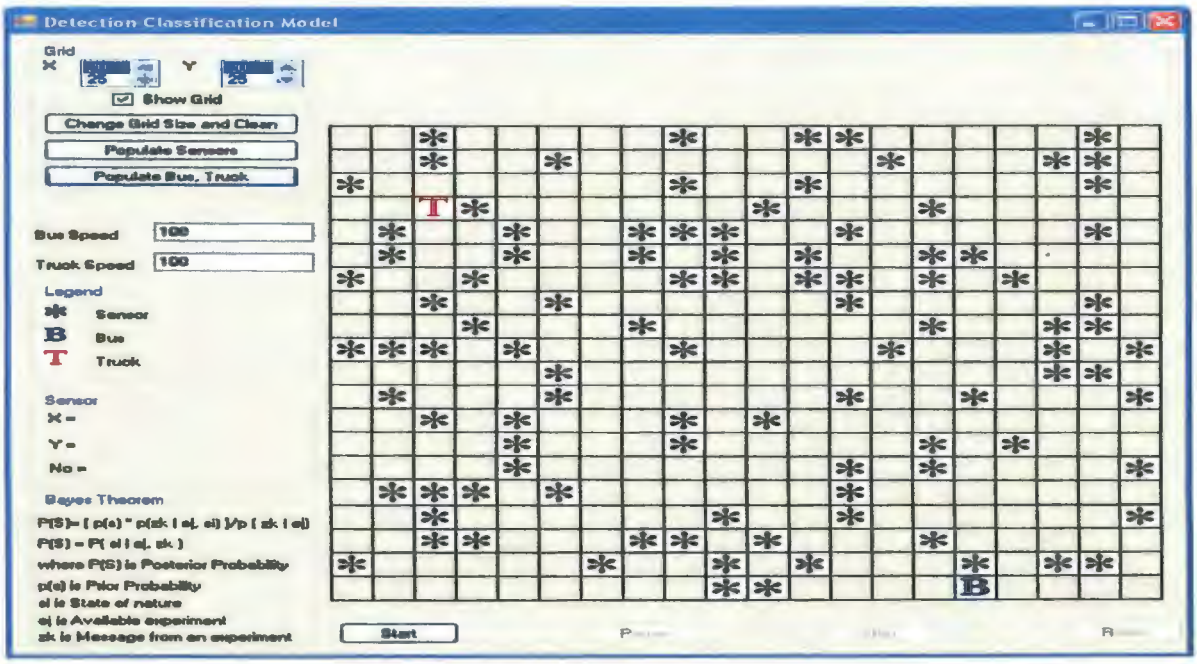

Figure 4.1. Initial Scenario

Figure 4.2 shows the interface when the simulation is started. The sensors show the probabilities when the mouse hovers , i.e., the probability to find a Bus if it is indeed a Bus, p (z11 |e1, s1); the probability to find a Truck if it is indeed a Bus, p (z12 |e1, s1) = 1-p (z11 |e1, s1); the probability to find a Truck if it is indeed a Truck, p (z22 |e1, s2); and the probability to find a Bus if it is indeed a Truck, p (z21 |e1, s2) = 1-p (z22 |e1, s2).

Figure 4.2. Starting the Simulation

When the Bus and Truck are populated, the sensors show the initial probabilities, i.e., before the simulation begins. The vehicles move randomly; the probabilities generated by the sensors are captured; and the Posterior Probabilities are calculated. For each sensor, there are two sets of Conditional Probabilities generated: one set of values with respect to the vehicle for image "B" and the other set of values with respect to the vehicle for image "T." Figure 4.3 shows the hover-over message for each sensor, including the location and also the sensor's index number.

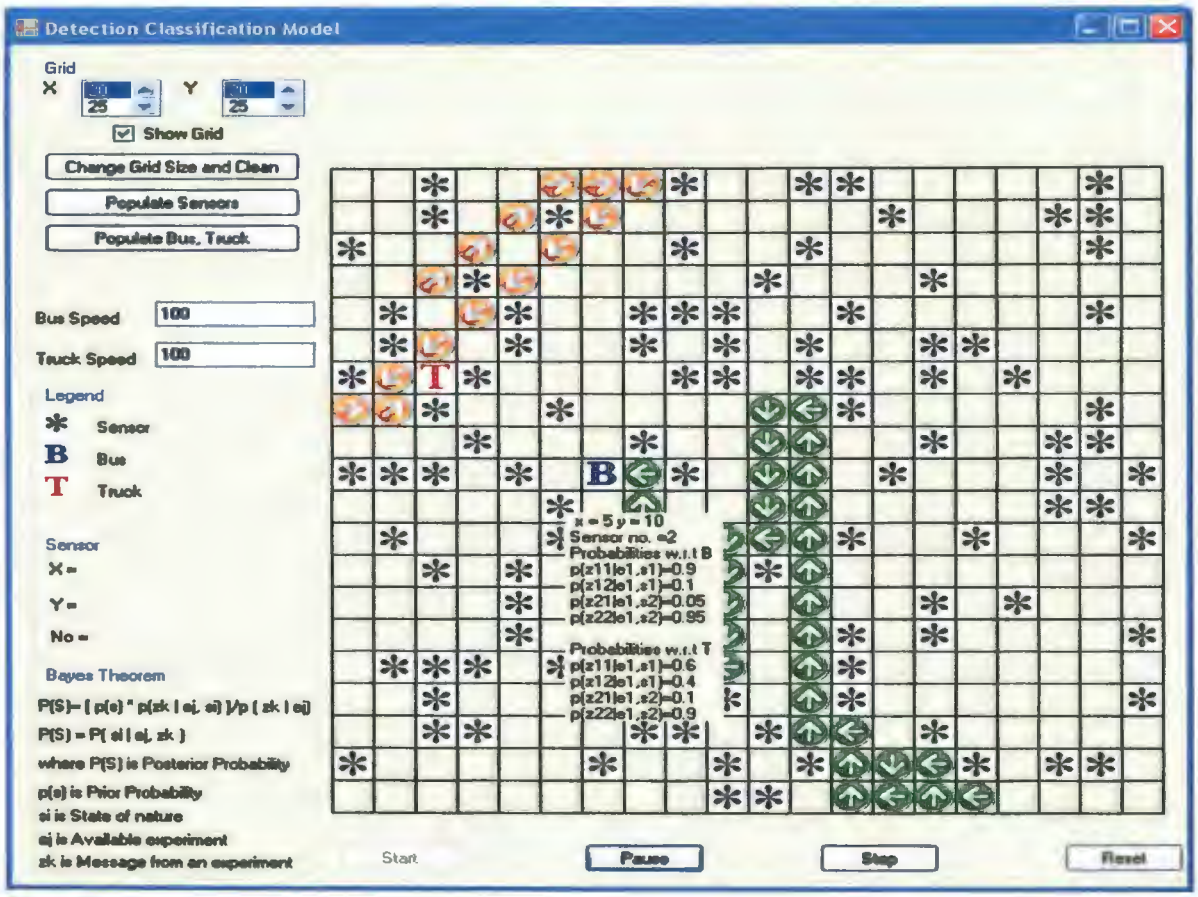

Figure 4.3. Message When the Mouse Hovers Over a Sensor

For each vehicle detection, a report is generated in the form of a Decision Tree. It is shown in APPENDIX A. REPORT GENERATED. Hence, for each iteration in a move, the report is generated in the form of a Decision Tree with all the necessary probabilities calculated and shown. Where e represents the experiment, s1 and s2 are states of nature, z1 and z2 represent the Joint Probabilities, P(z1|e1,s1), P(z2|e1, z2), P(z1|e1,s2) and P(z2|e1,s2) represent the Conditional Probabilities and P(s1|e1,z2), P(s2|e1,z1), P(s2|e1,z2), and P(s1|e1,z1) represent the Posterior Probabilities.

5. RESULTS

The Bayesian Approach was used to solve the Detection Classification problem. I used test cases where Test Case 1 and Test Case 2 help us find the efficiency of the algorithm designed for classifying the detection. The test cases were executed on a Windows XP Operating System.

5.1. Test Case 1

Experiments were carried out to find the number of sensors that successfully distinguish two types of moving objects. With the variable being the number of sensors, we conducted the experiments in decreasing order of sensors. The sensors are placed randomly, the default size of the geographical area is 20x20 grid squares, and the same layout is used for all the differing number of sensors. The movement of the vehicles is fixed with the variable number of sensors. Table 5.1 shows a "√" whenever the threshold values are reached.

Table 5.1. Number of Sensors That Successfully Distinguish Two Moving Objects at Given Thresholds

No. of Sensors	Threshold Value 0.85	Threshold Value 0.9	Threshold Value 0.98
50	√	√	√
45	√	√	√
40	√	√	√
35	√	√	√
30	√	√	√
25	√	√	√
20	√	√	√
15	√	√	√
10	√	√	√
07	√	√	-
05	√	-	-

With the threshold values set at 0.85, 0.9, and 0.98, the experiments revealed that the number of sensors required to successfully distinguish two moving objects reach the threshold values were, respectively, 5, 7, and 10 with the number of sensors ranging from 50 to 5. The experiment was run once for each threshold value, with the number of sensors decreasing in multiples of five. From Table 5.1, we can see the number of sensors to successfully distinguish two moving objects to reach a threshold value of 0.85 is 5. 7 sensors are required to reach a threshold value of 0.95 and at least 10 sensors are required to reach a threshold value of 0.98.

5.2. Test Case 2

Experiments were carried out to find how frequently threshold values are generated to distinguish two types of moving objects with the varying number of sensors. The values in Table 5.2 are displayed for the sensors that are placed randomly, the default size of the geographical area is 20x20 grid squares, and the same layout is used for all the differing number of sensors.

Table 5.2. Threshold Frequency With Varying Number of Sensors With the Threshold Value Set as 0.98

No. of sensors	Frequency to reach the threshold 0.98
10	5
20	8
30	13
40	20
50	25
60	29
70	33
80	38
90	42
100	45

There are 150 iterations simulated. For a given move of the two vehicles, the algorithm is run until the specified threshold is reached. Moves proceed one by one, and

the iterations are cumulative across the vehicle moves. The process stops when 150 iterations are reached, regardless of the number of vehicle moves. Figure 5.1 shows the values for the number of sensors on the x-axis and the number of times the threshold value, 0.98, was reached on the y-axis.

From the following Figure 5.1, we can see that the threshold frequency is linearly increasing with the corresponding number of sensors when the threshold value is 0.98. This finding shows that the number of sensors is directly proportional to the threshold frequency.

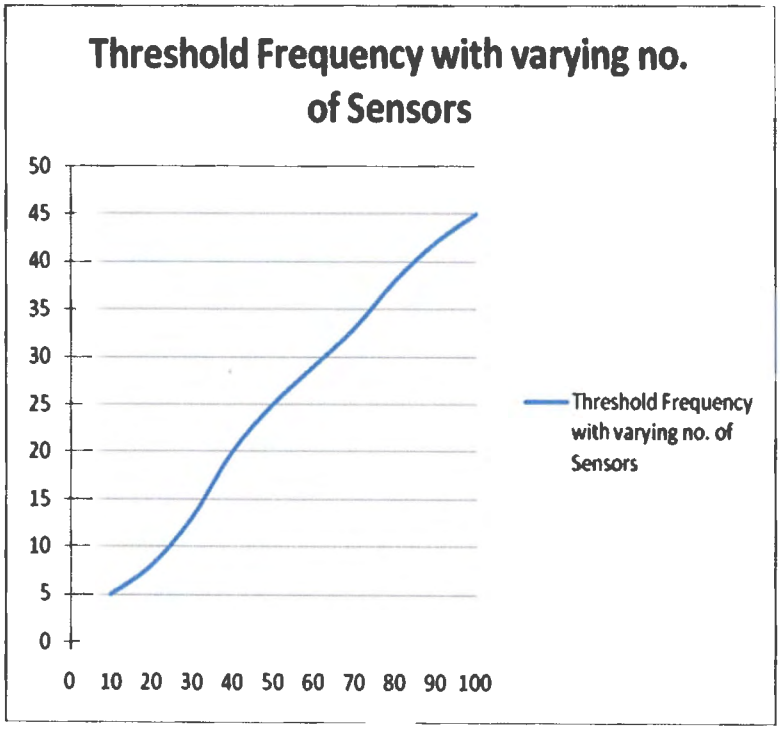

Figure 5.1. Threshold Frequency With Varying Number of Sensors With the Threshold Value Set as 0.98

Table 5.3 shows the frequency with which the variable number of sensors reached the threshold value of 0.9. The values in Table 5.3 are displayed for 150 iterations when all the moves for the simulation are considered with the variable number of sensors and the

threshold value considered is 0.9. Figure 5.2 shows the results for the number of sensors on the x-axis and the number of times the threshold value, 0.9 was reached on the y-axis.

Table 5.3. Threshold Frequency With Varying Number of Sensors With the Threshold Value Set as 0.9

No. of sensors	Frequency to reach the threshold 0.9
7	20
10	25
15	33
20	39
25	45
30	52
40	59
50	65
60	71
70	74
80	78
90	81
100	85

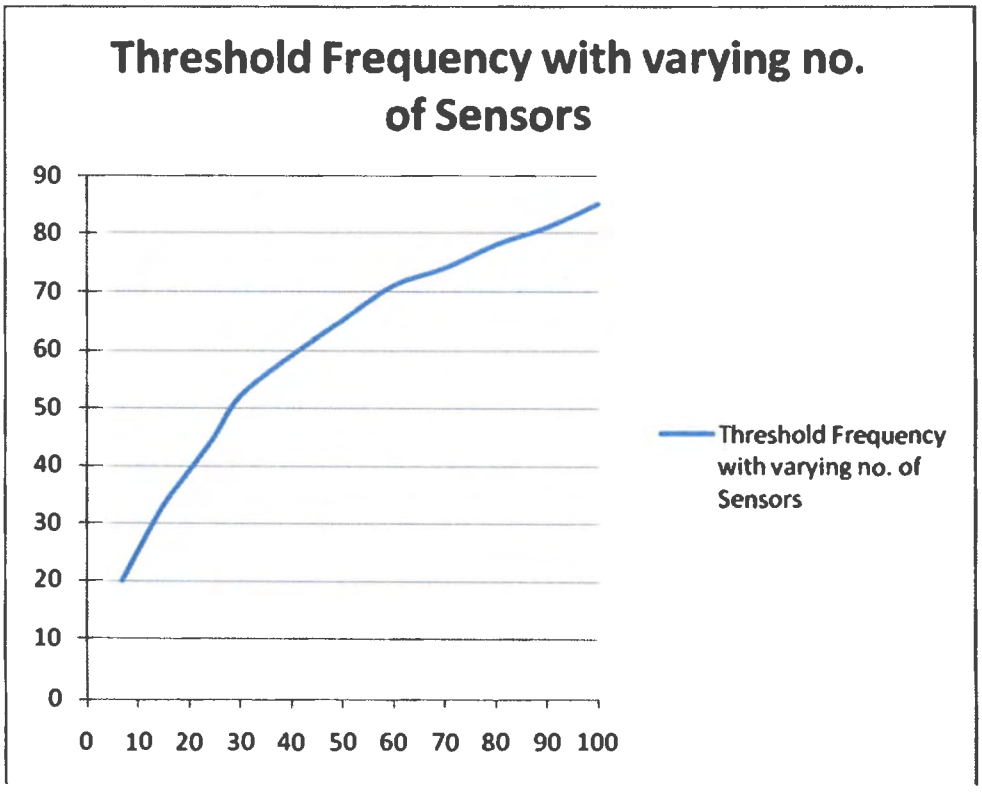

Figure 5.2. Threshold Frequency With Varying Number of Sensors With the Threshold Value Set as 0.9

From the graph in Figure 5.2, we can see that the threshold frequency is linearly increasing with the corresponding number of sensors when the threshold value is set at 0.9. Table 5.4 shows the frequency with which the variable number of sensors reached the threshold value of 0.85. Figure 5.3 shows the values for the number of sensors on the x-axis and the number of times the threshold value, 0.85, was reached on the y-axis. The values in Table 5.4 are displayed for 150 iterations when all the moves for the simulation are considered with the variable number of sensors and the threshold value considered is 0.85.

Table 5.4. Threshold Frequency With Varying Number of Sensors With the Threshold Value Set as 0.85

No. of sensors	Frequency to reach the threshold 0.85
5	12
10	28
20	35
30	48
40	64
50	72
60	80
70	89
80	97
90	104
100	110

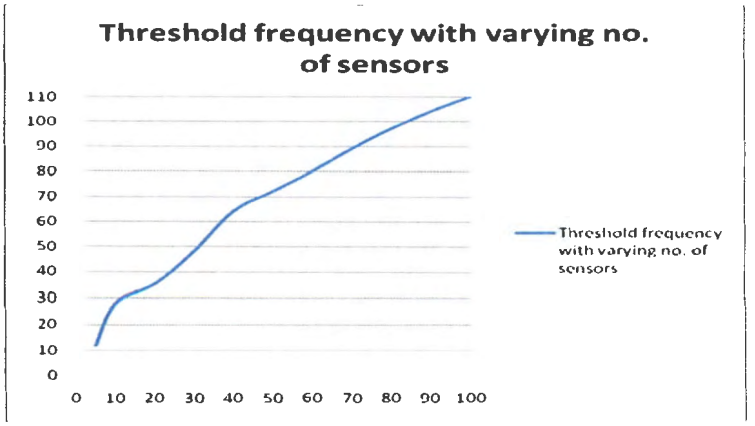

Figure 5.3. Threshold Frequency With Varying Number of Sensors With the Threshold Value Set as 0.85

From the graph in Figure 5.3, we can see that the threshold frequency is linearly increasing with the corresponding number of sensors. The experiment shows that the number of sensors is directly proportional to the threshold frequency.

6. CONCLUSIONS AND FUTURE WORK

A Posterior Probability Calculation Interface and a Detection Classification Model were developed and implemented and tested. In the Detection Classification Model where multiple sensors with sensing radii detected the two vehicles, Bus and Truck, the strategy is to explore an unknown area, added with some intelligence, with sensors that can detect the vehicles. This paper presented a Bayes' Theorem-based strategy where the sensors used different detection logics to classify the moving objects. Although the geographical area considered in this model was a finite, rectangular grid, the algorithm could be applied to any area geographically connected by neighborhood relations and to any moving objects. This model can be used to design multi-vehicular movement systems which involve probability-oriented navigation methods in the presence of obstacles.

While this implementation meets all of the essential requirements of the Bayes' Theorem, there are ways in which the solution could be augmented and improved. Use of techniques like sensor communication and movement can help find better solutions by exploring the solution space far more thoroughly. But the .NET-based implementation demonstration in this article can reach that level of sophistication since they are essentially just refinements of the basic algorithm.

This paper described a model that is limited to only two objects and static sensors. The application could be developed to accommodate multiple objects and different types of sensors. With dynamic targets, i.e., the movement of sensors with time, new things could be tried, and also, a special case, a point of coincidence for the objects with obstacles present in the geographical area, can also be considered. New languages can be developed

to describe scenarios, such as the geographical area of interest, location of targets, etc. Work can be done in scaling up the system with more and different types of grid variables.

With the experiments conducted, we can come to a conclusion that a particular vehicle can be found at a certain location in a designated area with the probability greater than or equal to a threshold value. The Bayes' Theorem is completely based on probabilities; we cannot really conclude that a particular vehicle is found at a location with an exact value.

REFERENCES

1. R. Biswas, S. Thrun, L. J. Guibas, "A Probabilistic Approach to Inference with Limited Information in Sensor Networks," Stanford University, California, 2004.
2. K. E. Nygard, "Bayesian Decision Analysis," Dept. of Computer Science, North Dakota State University, unpublished notes, 2009.
3. W. Fan, E. Greengrass, J. McCloskey, P. S. Yu, K. Drammey, "Effective Estimation of Posterior Probabilities: Explaining the Accuracy of Randomized Decision Tree Approaches," IBM T. J. Watson Research, pp. 154-161, Hawthorne, 2005.
4. D. S. T. Cuong, "Road Detection Using Intrinsic Colors in a Stereo Vision System," Dept. of Engineering in the Faculty of Engineering, National University of Singapore, 2009.
5. C. Faure, S. Delprat, J. F. Boulicaut, A. Mille, "Iterative Bayesian Network Implementation using Annotated Association Rules," Berlin, 2006.
6. H. Schneiderman, T. Kanade, "Object Detection Using the Stats of Parts," International Journal of Computer Vision, 56(3), 151-177, 2004.
7. C. Y. Lee, S. G. Chang, "A Probability-Based Location Management Strategy for the Next Generation Communication Systems," Dept. of Industrial Engineering, Korea Advanced Institute of Science and Technology, Korea, 2000.
8. A. F. Atiya, "Estimating the Posterior Probabilities Using the K-Nearest Neighbor Rule," Dept. of Computer Engineering, Cairo University, Egypt, 2004.

APPENDIX A. REPORT GENERATED

The detection report is generated in the form of a Decision Tree with all the necessary probabilities calculated and shown. Where e represents the experiment, s1 and s2 are states of nature, z1 and z2 represent the Joint Probabilities, P(z1|e1,s1), P(z2|e1, z2), P(z1|e1,s2) and P(z2|e1,s2) represent the Conditional Probabilities and P(s1|e1,z2), P(s2|e1,z1), P(s2|e1,z2), and P(s1|e1,z1) represent the Posterior Probabilities.

Decision.txt

Move No. 0

e1

s1 = 0.5

z1=0.42| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.82

z2=0.57| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.26

s2 = 0.5

z1=0.42| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.18

z2=0.57| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.74

e2

s1 = 0.82

z1=0.6| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.96

z2=0.4| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.62

s2 = 0.18

z1=0.6| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.04

z2=0.4| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.38

e3

s1 = 0.26

z1=0.29| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.62

z2=0.71| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.11

s2 = 0.74

z1=0.29| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.38

z2=0.71| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.89

e4

s1 = 0.96

z1=0.68| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.99

z2=0.32| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.88

s2 = 0.04

z1=0.68| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.01

z2=0.32| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.12

e5

s1 = 0.62

z1=0.49| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.88

z2=0.51| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.37

s2 = 0.38

z1=0.49| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.12

z2=0.51| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.63

e6

s1 = 0.62

z1=0.49| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.88

z2=0.51| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.37

s2 = 0.38

z1=0.49| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.12

z2=0.51| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.63

e7

s1 = 0.11

z1=0.21| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.37

z2=0.79| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.04

s2 = 0.89

z1=0.21| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.63

z2=0.79| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.96

e8

s1 = 0.88

z1=0.64| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.97

z2=0.36| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.73

s2 = 0.12

z1=0.64| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.03

z2=0.36| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.27

e9

s1 = 0.37

z1=0.35| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.73

z2=0.65| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.17

s2 = 0.63

z1=0.35| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.27

z2=0.65| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.83

e10

s1 = 0.37

z1=0.35| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.73

z2=0.65| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.17

s2 = 0.63

z1=0.35| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.27

z2=0.65| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.83

e11

s1 = 0.04

z1=0.17| P(z1|e1,s1)=0.7| P(s1|e1,z1)=0.17

z2=0.83| P(z2|e1,s1)=0.3| P(s1|e1,z2)=0.02

s2 = 0.96

z1=0.17| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.83

z2=0.83| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.98

Move No. 1

e1

s1 = 0.5

z1=0.38| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.8

z2=0.62| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.32

s2 = 0.5

z1=0.38| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.2

z2=0.62| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.68

e2

s1 = 0.8

z1=0.51| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.94

z2=0.49| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.65

s2 = 0.2

z1=0.51| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.06

z2=0.49| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.35

e3

s1 = 0.32

z1=0.29| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.65

z2=0.71| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.18

s2 = 0.68

z1=0.29| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.35

z2=0.71| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.82

e4

s1 = 0.94

z1=0.57| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.98

z2=0.43| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.88

s2 = 0.06

z1=0.57| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.02

z2=0.43| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.12

e5

s1 = 0.65

z1=0.44| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.88

z2=0.56| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.47

s2 = 0.35

z1=0.44| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.12

z2=0.56| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.53

e6

s1 = 0.65

z1=0.44| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.88

z2=0.56| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.47

s2 = 0.35

z1=0.44| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.12

z2=0.56| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.53

e7

s1 = 0.18

z1=0.23| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.47

z2=0.77| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.09

s2 = 0.82

z1=0.23| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.53

z2=0.77| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.91

e8

s1 = 0.88

z1=0.55| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.97

z2=0.45| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.78

s2 = 0.12

z1=0.55| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.03

z2=0.45| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.22

e9

s1 = 0.47

z1=0.36| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.78

z2=0.64| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.29

s2 = 0.53

z1=0.36| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.22

z2=0.64| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.71

e10

s1 = 0.47

z1=0.36| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.78

z2=0.64| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.29

s2 = 0.53

z1=0.36| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.22

z2=0.64| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.71

e11

s1 = 0.09

z1=0.19| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.29

z2=0.81| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.05

s2 = 0.91

z1=0.19| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.71

z2=0.81| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.95

e12

s1 = 0.78

z1=0.5| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.93

z2=0.5| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.63

s2 = 0.22

z1=0.5| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.07

z2=0.5| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.37

e13

s1 = 0.29

z1=0.28| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.63

z2=0.72| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.16

s2 = 0.71

z1=0.28| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.37

z2=0.72| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.84

e14

s1 = 0.29

z1=0.28| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.63

z2=0.72| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.16

s2 = 0.71

z1=0.28| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.37

z2=0.72| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.84

e15

s1 = 0.05

z1=0.17| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.16

z2=0.83| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.02

s2 = 0.95

z1=0.17| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.84

z2=0.83| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.98

e16

s1 = 0.63

z1=0.43| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.87

z2=0.57| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.44

s2 = 0.37

z1=0.43| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.13

z2=0.57| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.56

e17

s1 = 0.16

z1=0.22| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.44

z2=0.78| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.08

s2 = 0.84

z1=0.22| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.56

z2=0.78| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.92

e18

s1 = 0.16

z1=0.22| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.44

z2=0.78| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.08

s2 = 0.84

z1=0.22| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.56

z2=0.78| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.92

e19

s1 = 0.02

z1=0.16| P(z1|e1,s1)=0.6| P(s1|e1,z1)=0.08

z2=0.84| P(z2|e1,s1)=0.4| P(s1|e1,z2)=0.01

s2 = 0.98

z1=0.16| P(z1|e1,s2)=0.15| P(s2|e1,z1)=0.92

z2=0.84| P(z2|e1,s2)=0.85| P(s2|e1,z2)=0.99