Tactile-Based Mobile Robot Navigation

by

Xianchao Long

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

In

Robotics Engineering

by

____

May 2013

APPROVED:

____

Professor Eduardo Torres-Jara, Advisor

____

Professor Michael A. Gennert, Committee member

____

Professor Taskin Padir, Committee member

Abstract

This thesis presents an effective approach to study tactile based mobile robot navigation. A Matlab simulator, which can simulate the properties of the tactile sensors, the environment, and the motion of the robot, is developed. The simulator uses an abstraction model of a compliant tactile sensor to represent an array of sensors covering the robot. The tactile sensor can detect normal and shear forces. The simulator has been used by a set of human subjects to drive a robot in an indoor environment to capture data. The details of the implementation and the data collected are presented in this thesis. From the data, some contact features can be extracted. Regarding the features, this thesis uses the Gaussian classifier and Gaussian mixture model to classify the data and build the feature classification model. Comparing the classification results of these two methods, the Gaussian mixture model has better performance. Applying the feature classification model, some contact objects can be detected, such as wall and corner. Based on this classification tool, a simple navigation problem can be solved successfully.

Acknowledgements

I would first like to express my sincerest gratitude to my advisor, Dr. Eduardo Torres-Jara, for all the support and guidance from the onset of my experience at Worcester Polytechnic Institute (WPI). Thank you Eduardo for encouraging me to explore and innovate constantly, helping me to clarify my thinking when I trapped into study confusion, and guiding me to solve the problems in my research work. Discussing with you was always an amazing and exciting experience. I was lucky to work with you.

This thesis would have not been possible without the advice of my committee Prof. Michael A. Gennert and Prof. Taskin Padir. Thank you for your encouragement and insightful comments.

Furthermore, I would also like to express my deepest gratitude to my friends in WPI. Hong, Hanlin and Du, Ruixiang, thank you for discussing with me and sharing some ideas with me. Thanks Ji, Wenzhi for testing my simulator and giving me some valuable advices.

Finally, I would like to thank my parents, Ma, Xiangying and Long, Chengzhong, for supporting me throughout all my studies at WPI. Thank you, Mom and Dad, for your encouragement and understanding which provide me confidence and strength to pursue my goal. Your selfless love from another side of the world always keeps me moving forward.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robot navigation, the problem about guiding a robot to approach a destination, is one of the main research topics in autonomous robot system. In our daily life, the navigation system with which we are most familiar is the GPS. GPS can also be applied in the robot, but it is not sufficient to deal with the problems in robot navigation. For instance, the robot has to perform obstacle detection [1], [2], environment mapping [3], [4], landmark recognition [5], [6] and path planning [7], [8]. A lot of work has been done in this area. In the second competition of the DARPA grand challenge, the winner, Stanley, drove 132 miles off-road terrain in 6hour 53min without manual intervention [9] (see Figure 1.1 (a)). Mars rover, Curiosity, used its navigation camera to find its path to the target (see Figure 1.1 (b)).

However, most of navigation system in robotics used light or other radiation sensors to detect obstacles. This allows a robot to move in open space like a road or a building hall. However, robots based on those sensors have some limitations: the sensors do not well at short distance, the sensors do not cover the robot's body completely, and they depend on the reflective surface.

(a) (b)

Figure 1. 1 (a) Stanley [9] (b) Curiosity Rover [10]

Thus, a robot does not work well when navigating in a cluttered environment such as debris in a disaster area with debris (see Figure 1.2 (a)), a narrow tunnel (see Figure 1.2 (b)), or through door. Animals, and humans, have little problem navigating through cluttered environment. For instance, a cave explorer can move through narrow cave by using not only the vision but also the contact information provided by the sensitive skin (see Figure 1.2 (b)). The skin allows us to be aware of the interaction between its body and its environment, which enables to make inform decisions. Moreover, what it touches is the real physical object which is sensed more reliably.

(a) (b)

Figure 1. 2 (a) Disaster Area [11] (b) Cave Detection [12]

Therefore, in order to have robots that are more autonomous and capable of operating on cluttered environments, we need to cover them with sensitive artificial skin using tactile sensors.

Developing such robots presents multiple challenges. For instance, the appropriate type of tactile sensor needs to be used, and algorithms to process the information of this type of sensor arrays are required to be developed. There are many tactile sensor technologies that used different principles, such as, piezoresistance [13], [14], optics [15], [16], capacitance [17], [18], ultrasound [19], and magnetism [20], [21]. A complete review of other technologies can be found in [22]. These technologies have been used in different type of applications such as collision detection [11], object recognition [23], navigation [24], [25] and human robot interaction [26], [27]. In this thesis, we study the problem of navigating in cluttered environments based on tactile sensing using a simulated robot. The simulation developed has a robot covered with tactile sensors. An abstraction model of a compliant tactile sensor tested in sensitive robotics is used [28]. The contact information obtain is analyzed to learn contact patterns. This information is used to develop basic navigation algorithms.

1.2 Thesis Organization

This thesis is organized as follows:

Chapter 2 describes the simulator in detail. There are five parts in this chapter, such as the simulated model of the tactile sensor, the robot kinematic model, Graphical User Interface (GUI), framework and controller in the simulator. The functions of the simulator are also introduced in Chapter 2.

As a starting point to develop the algorithms we have conducted experiments with human subjects using the simulators. The process of the experiment and the results are presented in Chapter 3.

The data analysis of the information collected from the experimentsis discussed in Chapter 4. Special attention is given to the feature classification. Two methods have been used, Gaussian classifier and Gaussian mixture model.

In Chapter 5, the performance of the feature classification model is evaluated by applying the model to a simple navigation mission.

Conclusion and future works are summarized in Chapter 6.

Chapter 2

Simulator

This chapter describes a 2-D simulator developed to study tactile based navigation. The simulator is capable of driving a mobile robot in an unknown environment using tactile feedback has been built. The robot has 44 tactile sensors around its body to detect contact with the environment. The simulator has a GUI interface for a user to drive the robot based on the tactile feedback.

2.1 Tactile Sensor

The sensor used in this simulator was developed by Torres-Jara et al [20]. This compliant tactile sensor has been successfully used in manipulation [29] (see Figure 2.1 (a)). This sensor has unique characteristics such as the detection of normal and shear forces, which allows it to estimate the contact independently of the angle of incidence. Moreover, the geometry of the sensor helps to detect feature in the object that is touched and its dome-shape silicon rubber surface have shock absorption and buffer ability (see Figure 2.1 (b)). The study of this compliant sensor was done with finite element analysis (FEA) (see Figure 2.2). As you can see, (a) display a result when a normal force acts on the top of the dome surface. (b) show a force acts at 45 degree away from the top of the dome surface. These FEA results show a linear relation between the load and displacement.

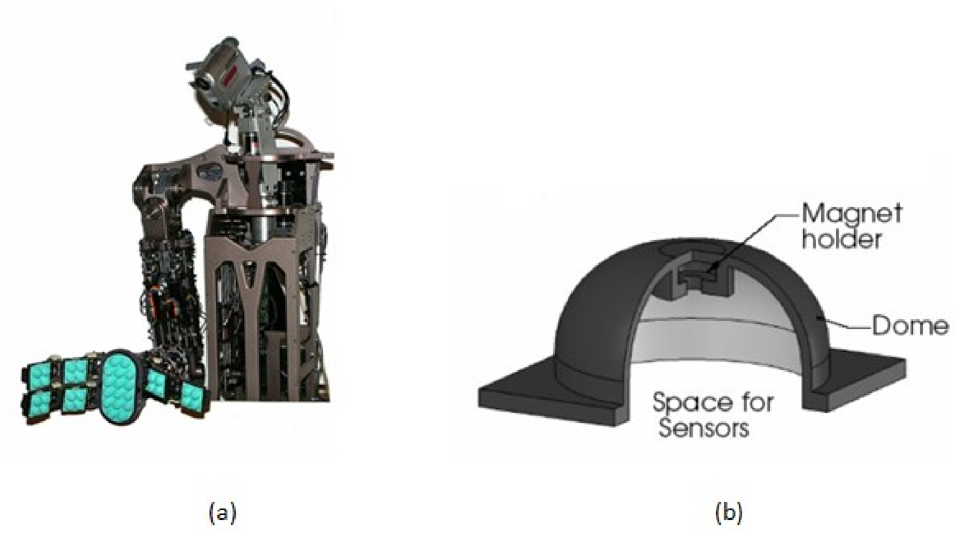


Figure 2. 1 (a) Robot OBRERO (b) Tactile Sensor Shape

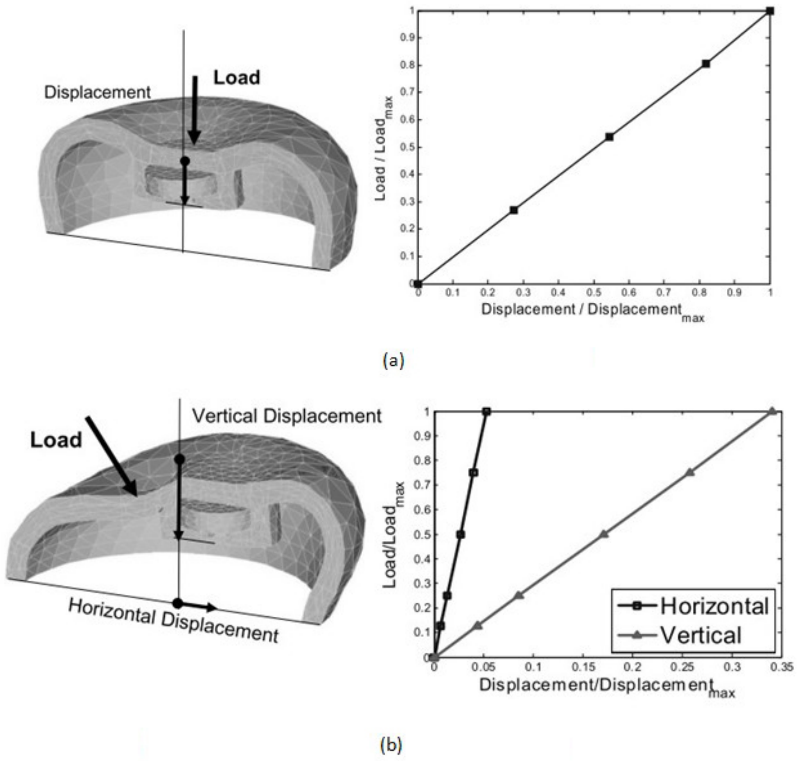


Figure 2. 2 FEA about deforming mesh under (a) a normal force acting on top of the dome and (b) a load acting normal to the surface 45 degrees away from the top of the dome. (The displacements are normalized by maximum vertical displacement of 5 mm, the maximum horizontal displacement of 7.5 mm and the maximum load is 0.147N)

However, FEA is not a feasible alternative when we want to simulate arrays of these sensors. An abstraction model for this compliant tactile sensor has been developed [28] and tested in manipulation. The abstraction model is a rigid sphere connected to a spring and a damper (see Figure 2.3). The basic idea is that: when contacting with the obstacle, the sensor would move backward along the line passing through the center of the half circle and the point in obstacle which is closest to the half circle center; the displacement depends on the distance of this backward movement. In Figure 2.3, dx indicates the horizontal force and dy indicates the vertical force. According to this method, the load/displacement relation is shown in Figure 2.3 (a) and (b).

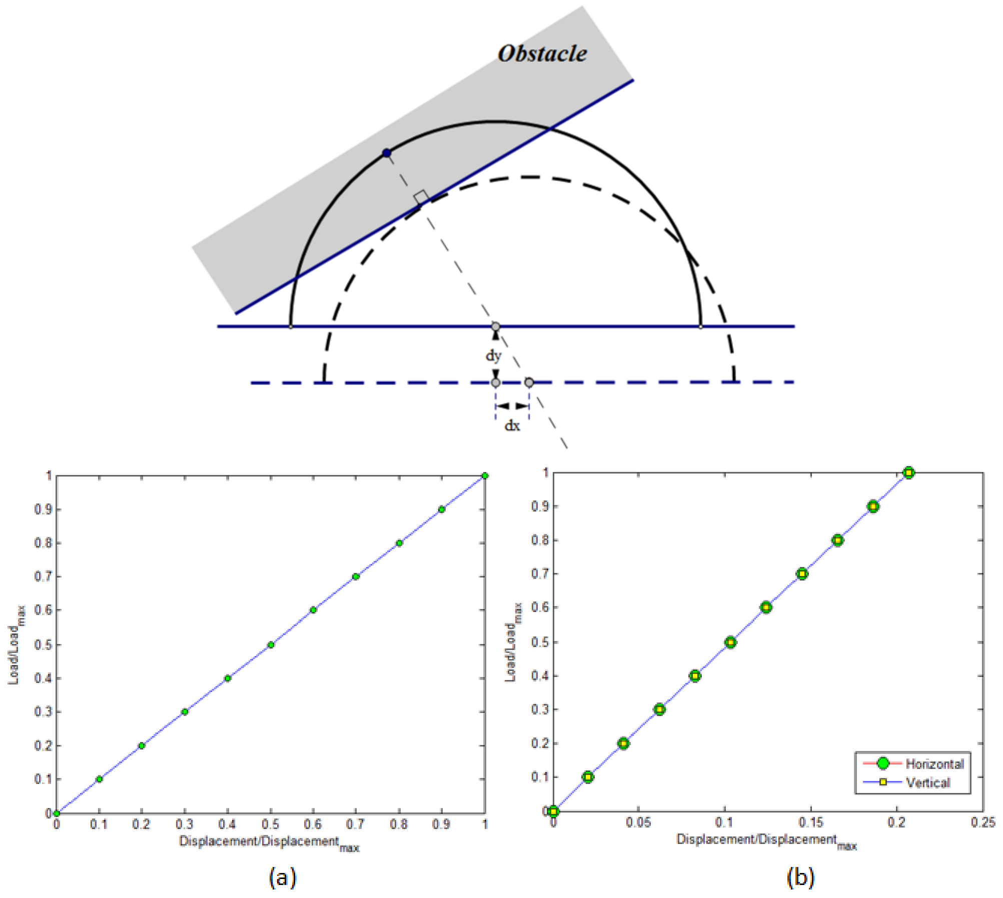


Figure 2. 3 The method used in the simulator to represent the sensors' signal (The load/displacement relation is shown in (a) and (b). (a) indicates a load acts on the top; (b) indicates a load acts at 45 degree away from the top)

Although the load/displacement curves are not completely the same, the changing trend and the linearity characteristic are similar. It is more important that the two main characteristic of the original sensor are preserved in this model: detection of contact independently of the angle of incidence, and capability to detect feature in the object touched. Therefore, this model can be used to simulate the real sensor signal in the simulator.

2.2 Robot

The robot is a car whose shape is a rounded rectangle. The car in this simulator uses the differential drive model, which is shown in Figure 2.4.

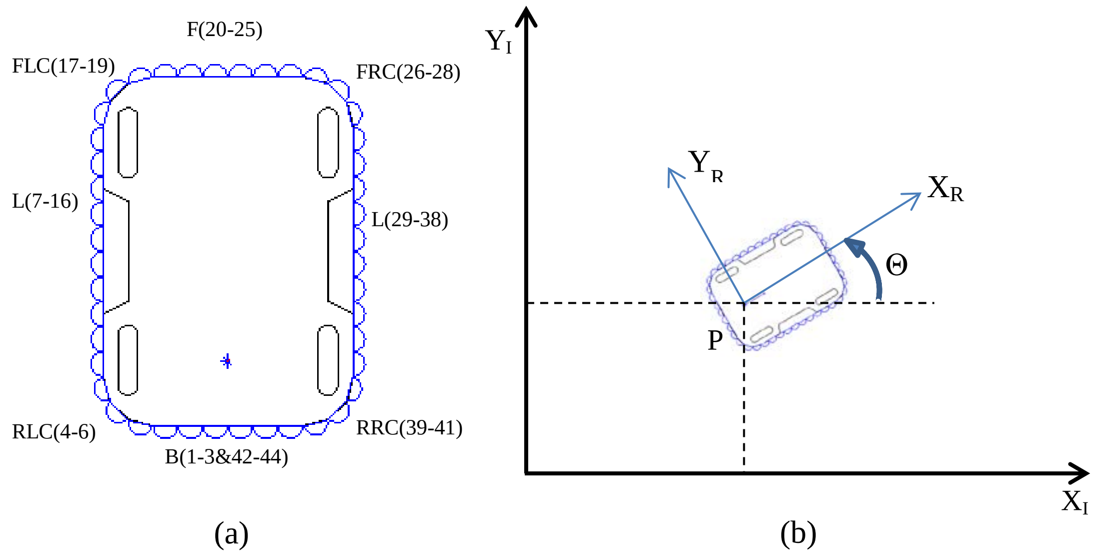


Figure 2. 4 (a) The geometric shape of the robot (the symbols denote the position of the sensors and the numbers denote the index of the sensors) (b) The kinematic model of the robot

The car’s kinematics is that:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = R_I^R(\theta) * \begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta}_I \end{bmatrix}, where\ R_I^R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When controlling the car, the velocity and angular velocity would be set. To draw the car in the map, these parameters need to be converted from the local reference frame of the car to the global reference frame of the plane.

$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta}_I \end{bmatrix} = R_R^I(\theta) * \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix}, where\ R_R^I(\theta) = \ R_I^R(\theta)^{-1} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The kinematics can help me build the animation of the simulation conveniently.

2.3 GUI

There are two frames in GUI (see Figure 2.5), which are the Simulator and Data frames. The Simulator frame is used to display the current status of the robot and the current sensor signal. The whole experiment environment can also be checked in this frame. Data frame is used to record all of the sensor data which are produced during the experiment. Therefore, the motion of the robot can be shown in this frame. In detail, Part 1 is a picture which shows the car’s status. In this picture, the blue arrow expresses the car’s forward direction. The red arrow is pointing to destination. The number at the bottom is the distance between the car and the destination. The sensors filled blue color show that they are contacting with the obstacle. This picture has another mode which can display the whole experiment environment (see Figure 2.6). Part 2 is the current signal of the sensor array (F: front sensors, L: left side sensors, FLC: sensors on the front left corner, RLC: sensors on the rear left corner, R: right side sensors, FRC: sensors on the front right corner, RRC: sensors on the rear right corner, B: rear sensors). Part 3 is a collision alarm flag. Red means the car is blocked, while green means the car run well. Part 4 is a button for expanding the window to show the Data frame. Part 5 is a button for saving the experiment data.

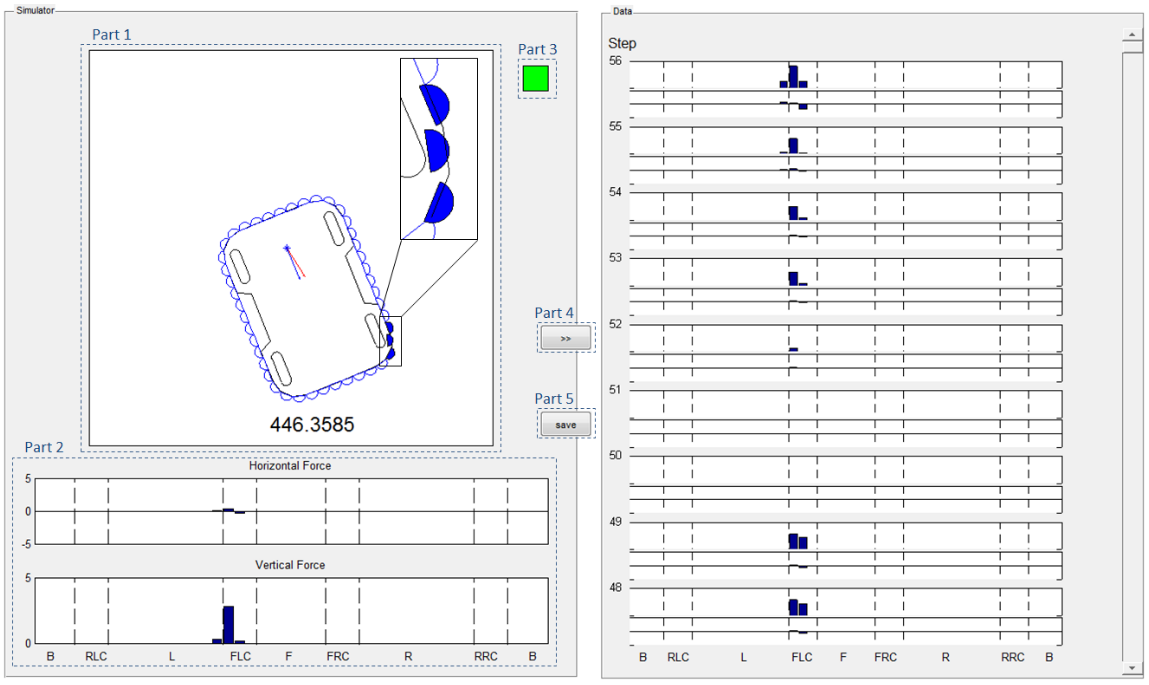


Figure 2. 5 Simulator Window.

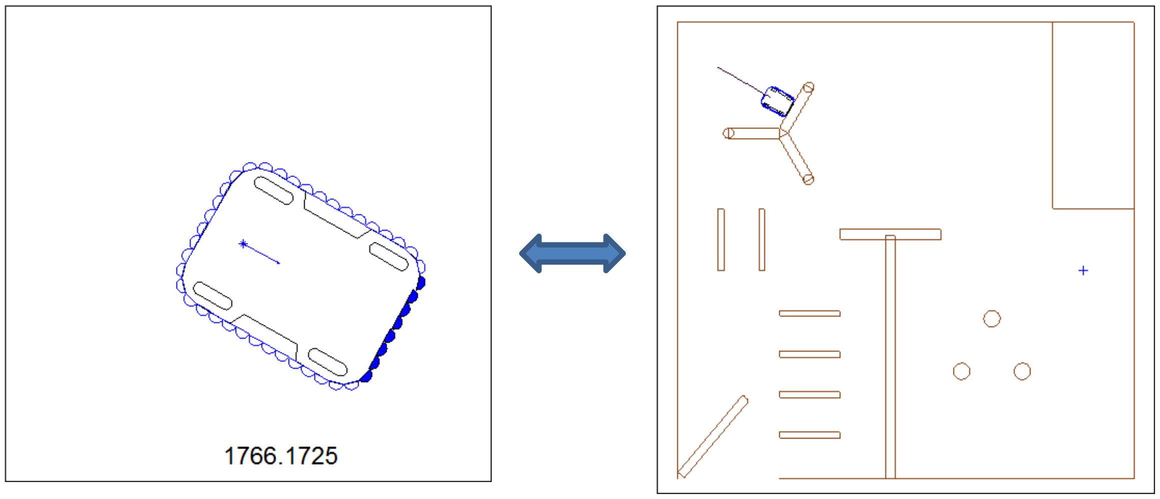


Figure 2. 6 Picture in simulation frame can be altered between the car status picture and the environment map

2.4 Framework

The simulator is built by the Matlab object-oriented programming method and its framework is shown in Figure 2.7.

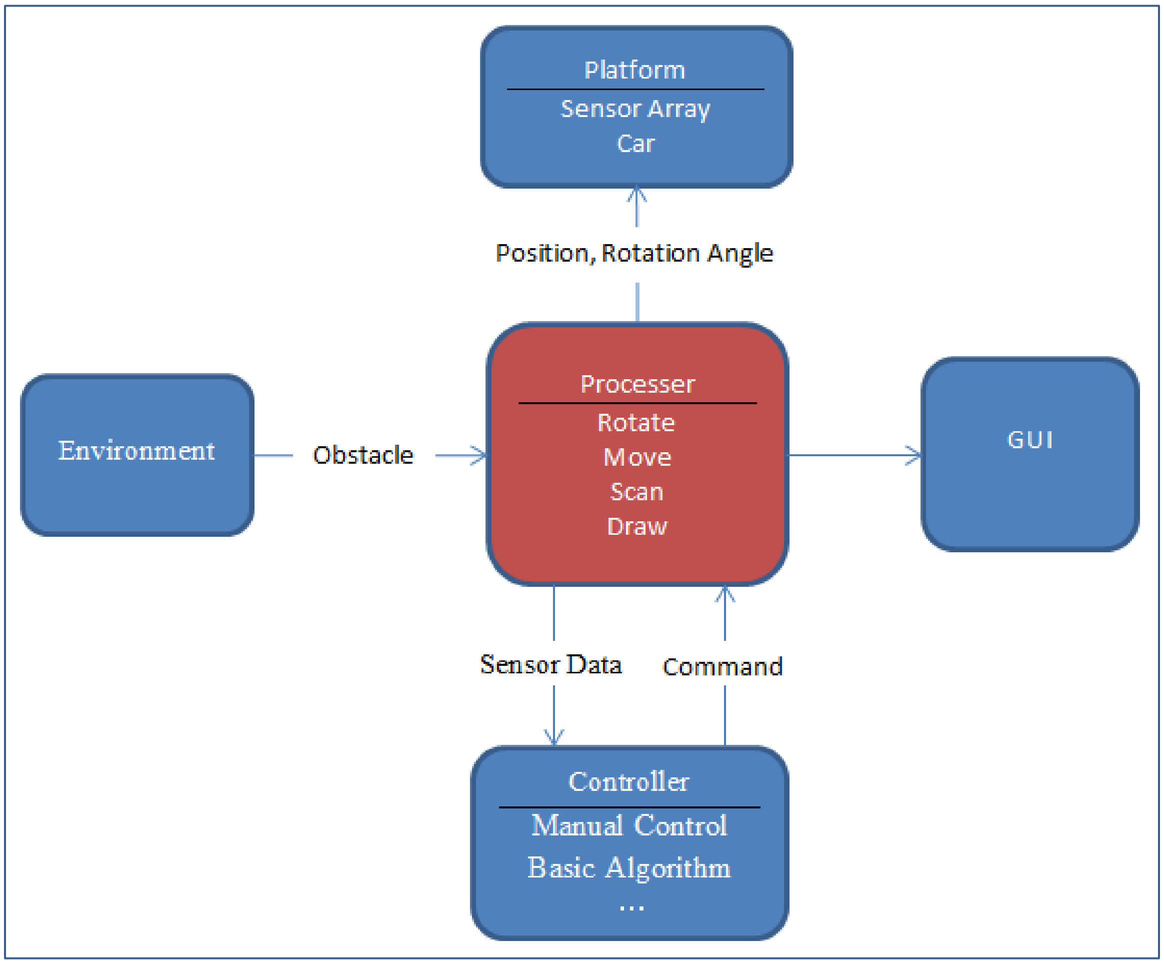


Figure 2. 7 Framework of Simulator

There are five major parts in the framework:

Platform defines the sensor array and car, and provides functions to reset the position and rotation angle;

Environment defines the indoor environment;

Processer contains the functions to rotate and move the platform, generate the sensor data based on the obstacle information which come from environment, and draw the platform and environment in the GUI;

Controller is the place for designing the control algorithm, and whose input is the sensor data and output is the control command;

GUI displays the simulator windows and provides some UI controls.

At the beginning of a step, the controller will give a command to the processer. Then the processer will attempt to rotate and move the platform, and detect the obstacle by tactile sensor array. If there is a collision which means the command is invalid, the temporary movement needs to be cancelled, the platform would keep the previous status, and the collision alarm should be triggered. If not, the new position and rotation angle of the platform will be updated, and the new sensor data need to be recorded. Finally, the GUI will be updated. This is the working process for every step.

2.5 Controller

It is described in the section 2.2 that the kinematics of the car is designed according to the differential-drive robot model. Therefore, the car can move forward and backward, and rotate around the vertical axis which is orthogonal to the plane and passing through the midpoint between the rear wheels. The output command of the controller is the linear velocity and angular velocity of the car. Currently, there is a manual control method in the controller, which is designed for testing the simulator. The functions of this method are listed as follows:

Use “up” and “down” arrow keys to give large scale linear velocity commands, while “W” and “S” keys are used for giving small scale linear velocity commands.

Use “left” and “right” arrow keys to give large scale angular velocity commands, while “A” and “D” keys are used for giving small scale angular velocity commands.

Use “M” to convert the picture in the Simulator frame between car status mode and environment mode.

2.6 Conclusion

The simulator is designed for controlling manually the car only based on the data collected from the tactile sensors. Some experiments can be performed to examine the functionalities of the simulator.

Chapter 3

Experiment

To understand if a robot can be driven to the goal through a room with obstacles only using tactile feedback, we asked 20 human subjects to drive the simulated robot. The subject ages range from 19 to 28.

3.1 Process of Experiment

Step One: Let the subjects learn how to use the simulator. To be familiar with the operation methods, subjects drove the car under the practice mode in which there is a simple practice scene. In this stage, subjects were allowed to check the car's status picture and the practice environment by using the 'M' key (see Figure 3.1).

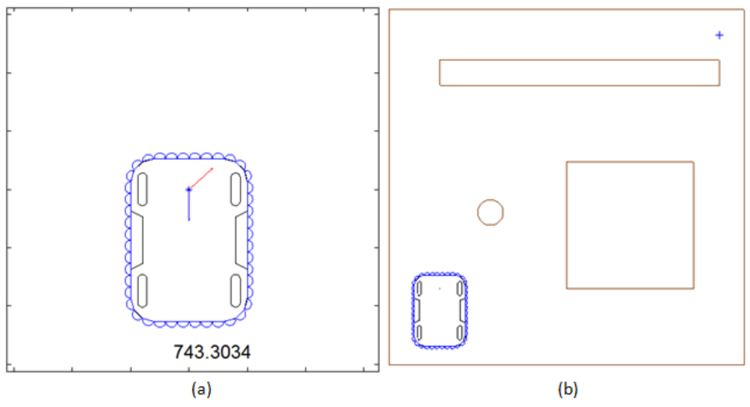


Figure 3. 1 (a) Car's status picture (b) Practice environment picture

Step Two: The subjects are brief on the BUG algorithms [30], [31], in case they want to use it. BUG algorithms are a well-known algorithm to solve the problem about path planning for a point object moving amidst unknown obstacles. Since the algorithms were proved to be feasible by navigating a fictitious point object with position and tactile feedback, it was provided as a reference for the subjects.

The procedure of BUG 1 algorithm is that:

First, head toward goal.

Second, if an obstacle is encountered, circumnavigate it and remember how close you get to the goal.

Third, return to that closest point and continue.

The procedure of BUG 2 algorithm is that:

First, head toward goal on the line from the starting point to the goal (S-G line).

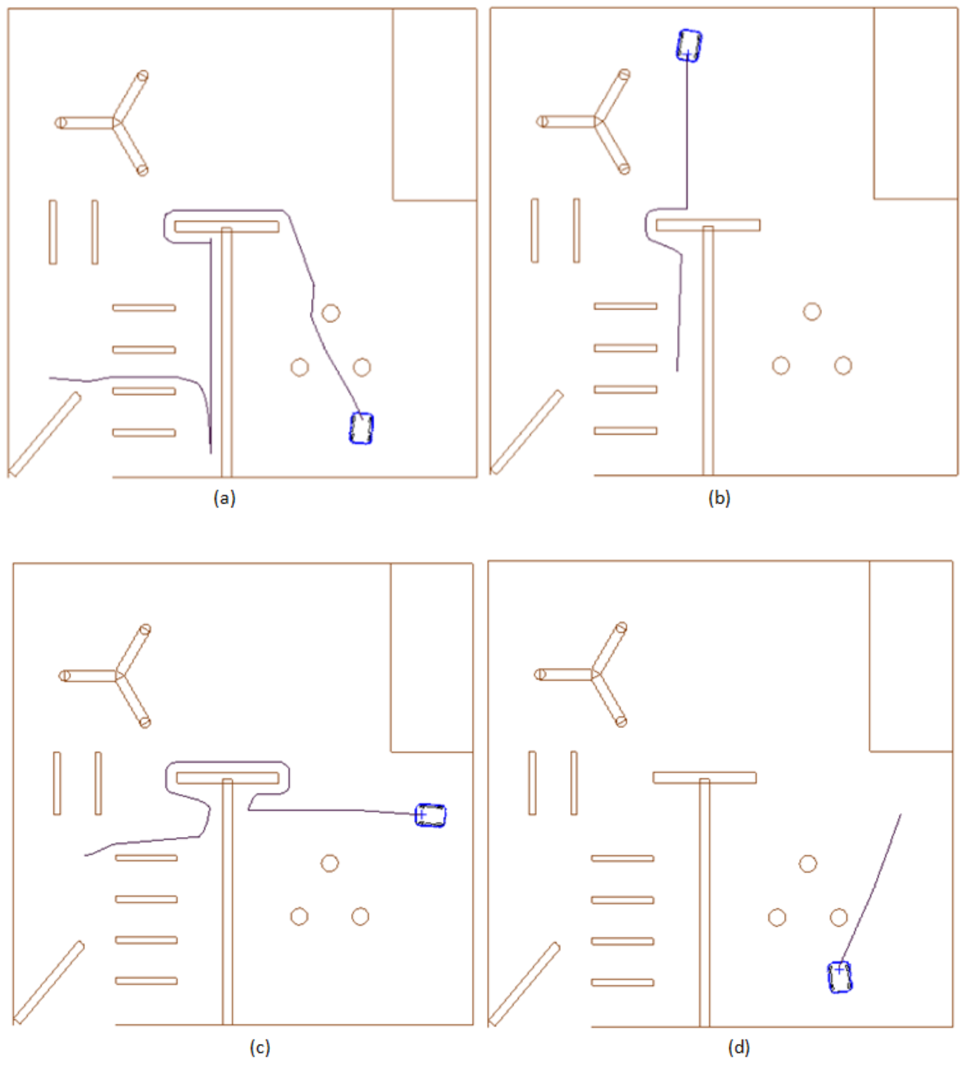
Second, if an obstacle is in the way, follow it until you encounter the S-G line again.

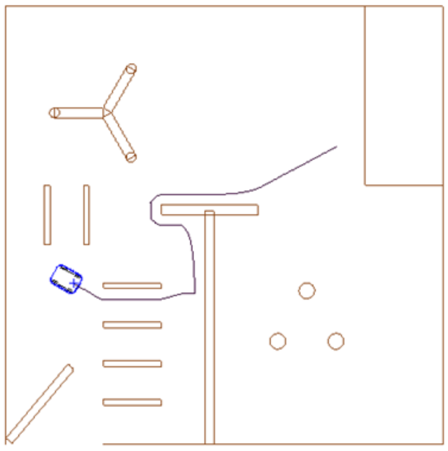
Third, Leave the obstacle and continue toward the goal.

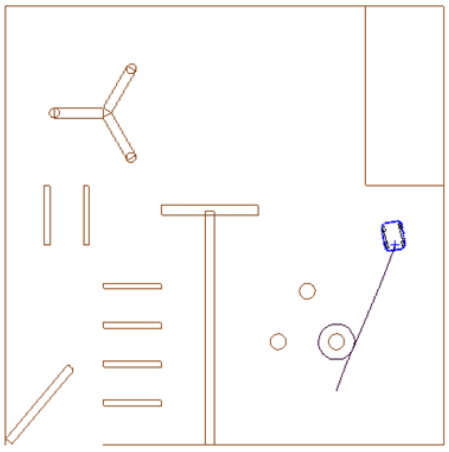
Step Three: The subjects navigated the car to approach the destination only according to the car’s status picture and sensor data. During the experiments, the subjects did not know anything about the environment setting in the simulator. They were not allowed to peep at the environment. The initial position of the robot and the destination were selected randomly. The subjects can stop the experiment when they navigate the car to the destination successfully, they give up the mission or the experiment time is up. The last two situations are seen to be failure results.

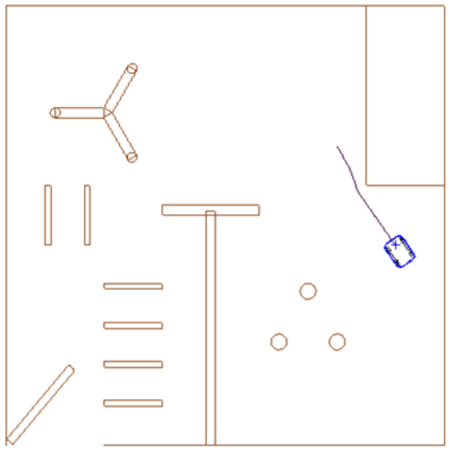
3.2 Experiment Result

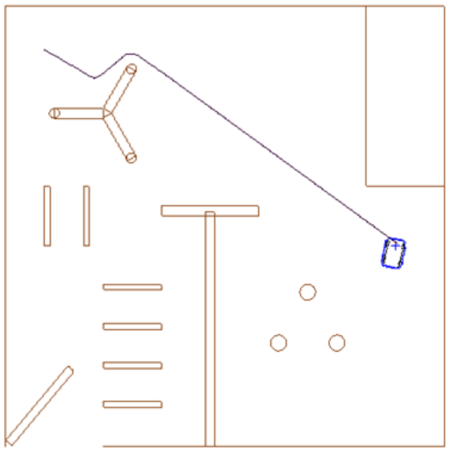
As a result, there were 14 people successfully controlling the car to approach the destination. Some experimental route maps are shown in Figure 3.2. As you can see, some subjects used the BUG algorithms, whose route maps are (b), (c), (f), (i), (l) and (n). These successful cases prove that the path planning methods can be realized in the simulator, and the tactile sensors model using in the simulator can provide enough information to navigate the car.

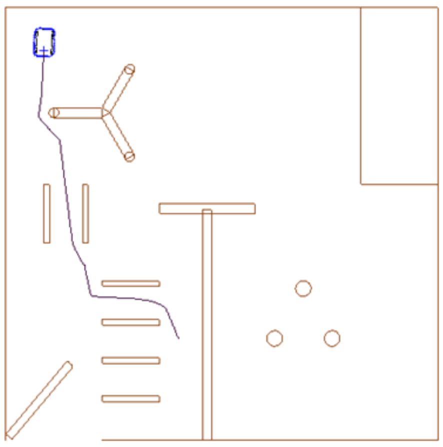


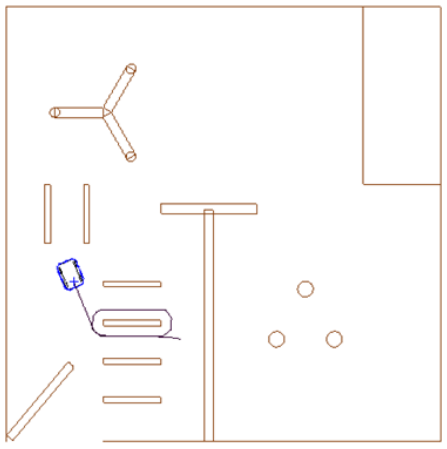

(e)


(f)


(g)


(h)


(i)


(j)

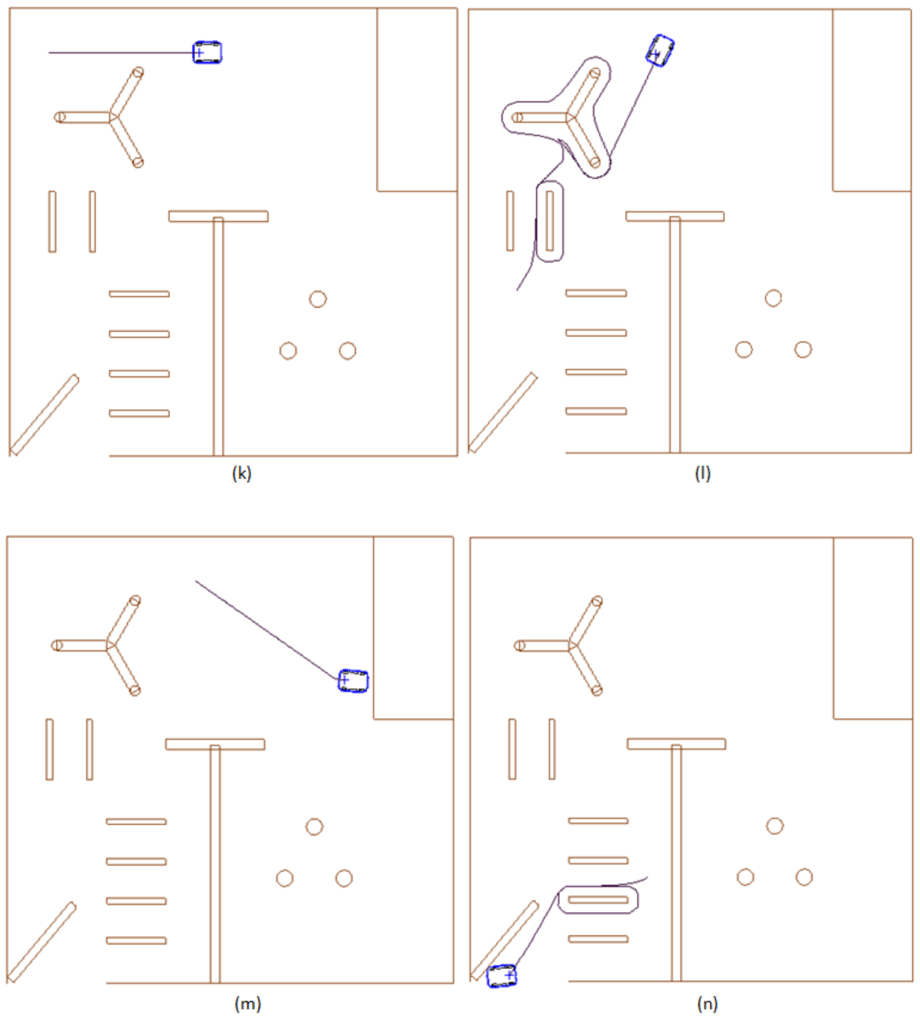


Figure 3. 2 Success Results in Experiment

6 people were unable to complete the mission (see Figure 3.3). The main reason is that the car is not as agile as the point object. Since there are many motion constraints, the car would lose contact when it follows an obstacle under the unskillful control. The results suggest that some basic control methods, which can guarantee the car would keep touch the obstacle when it follow the obstacle’s boundary, are necessary.

These experiments also provide rich data for further study.

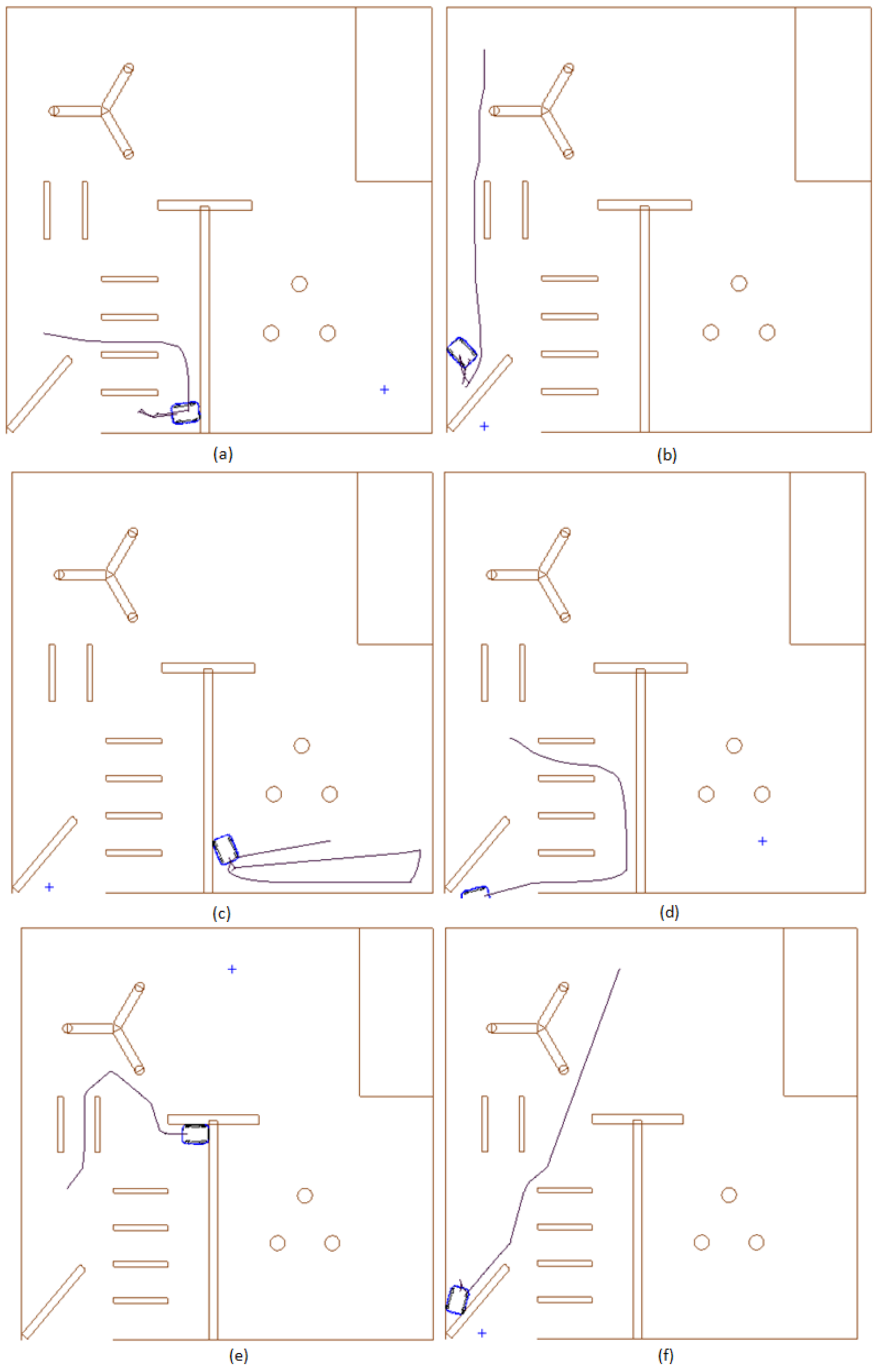


Figure 3. 3 Failure Results in Experiment

Chapter 4

Data Processing

Using the data collected from the experiment described in Chapter 3, we investigate how to classify the information to learn features to recognize the obstacles, map the environment, and plan a path. For example, we want to use the data of the tactile sensor array to determine the driving state of the car. States such as the robot is following a wall, leaving a wall, and passing a corner.

4.1 Simulation Analysis Tool

For the purpose of processing the experimental data, an analysis tool was designed, which is shown in Figure 4.1. The route of the car, the sensor data, the car direction, and the command are available in this tool.

There are three parts in this tool.

The left part displays the experimental environment, the route of the car, the current position and the current sensor signal. The right part contains sensor data set, direction set and command set, which displays the sensor data, the car direction and the control command in each step. All these data matrices are represented by the images. The values of the sensor data which range from 0 to 5 are mapped to the colors which range from blue to red. The color bar in the top of the left part, which indicates the relationship between the values and the colors, can be applied to all the sensor data images. In car direction image, blue color data indicates that the angle of direction is in the first and forth quadrant, while the red color data indicates that the angle is in the second and third quadrant. In the command set, the commands are represented by red color. The symbol "L", "R", "F" and "B" denotes the command turning left, turning right, moving forward and moving backward respectively.

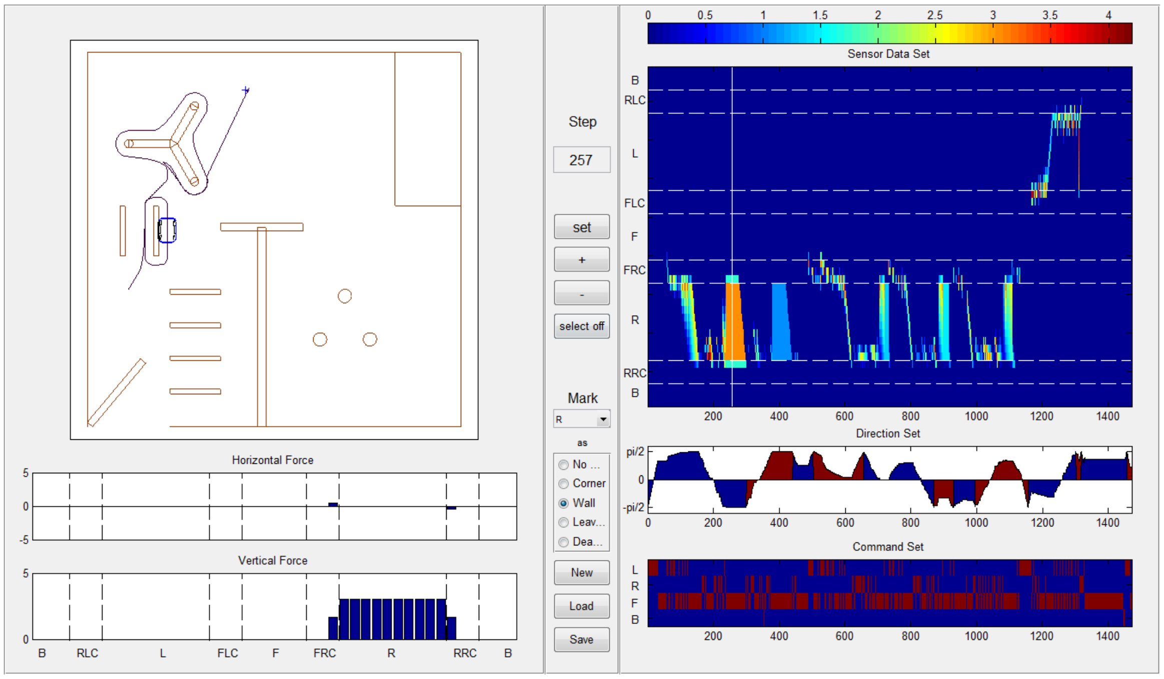


Figure 4. 1 Data Analysis Tool

The middle part is an operation panel. One function of this panel is reviewing the movement of the car. Users can manually set the step they want to check. They can also click on "+" and "-" button to see the next step and previous step. Additionally, if a user is interested in a certain point in the car's route, he/she can open the selection mode and obtain the step number of the point by clicking on it. Then, the user can set the car to that point. This selection mode is also available in the sensor data set. To find out the feature of the obstacle which the car is contacting with, there is a part that allows users to mark the sensor data for each step in this operation panel. Users can choose the part of the sensors they want to mark. Currently, five features have been defined, such as "No

Contact”, “Corner”, “Wall”, “Leaving Wall”(LW) and “Dead Path”. The meanings of these features are shown as follows:

“No Contact” --- there is no obstacle contacting with the car.

“Corner” --- the car is moving round the corner of the object.

“Wall” --- the car is contacting with a wall.

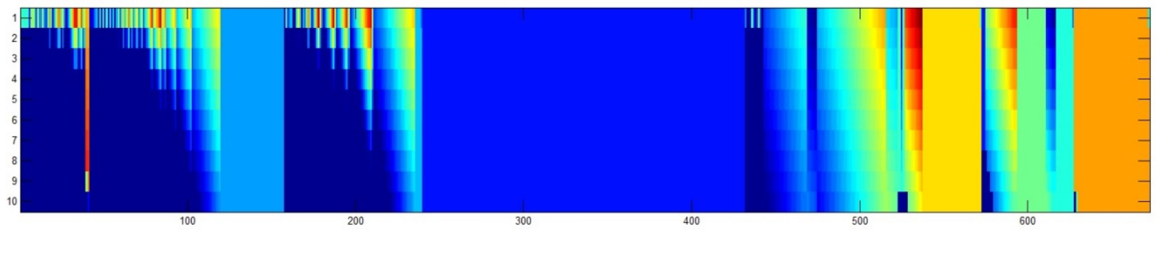
“Leaving Wall” --- the car is leaving a wall.

“Dead Path” --- the car cannot move anywhere but retrace its steps.

The “New”, “Load” and “Save” buttons are used for building, loading and saving the label set.

4.2 Data Set

The database consists of two parts, training data and testing data. Four experiment results were chose as training data and another four as testing data. As stated earlier, the whole sensor array was divided into eight parts, such as front side, rear side, left side, right side, left front corner, right front corner, left rear corner and right rear corner. The results were analyzed part by part. Four kinds of features are marked for the data in the database, which are “No Contact”, “Wall”, “Corner” and “Leaving Wall”. There is no “Dead Path” data in this database. In these features, “No Contact” is obvious, since all sensors data in this situation are zero. “Wall”, “Corner” and “Leaving Wall” (LW) are the features which need to be classified. The data are categorized in Figure 4.2 to Figure 4.8 according to these features. There is no contact signal in the rear side sensors and no “LW” feature data in sensors of front side, right front corner and left front corner.



(a)

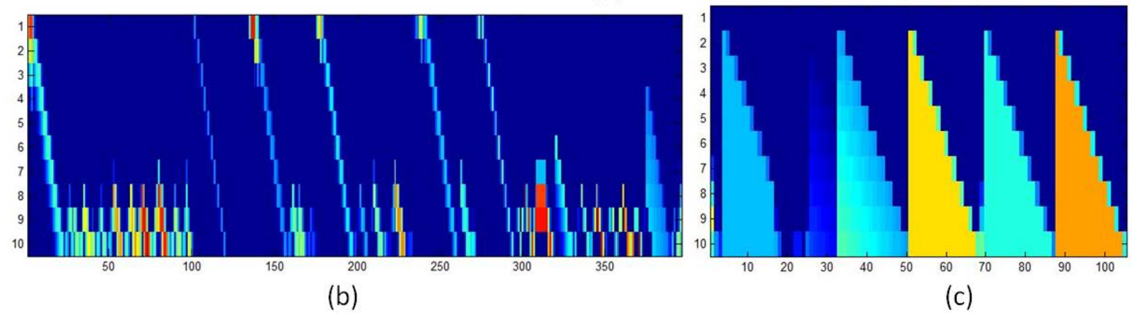


Figure 4. 2 Data collected from the sensor in the left side ((a) data marked as "Wall", (b) data marked as "Corner", (c) data marked as "LW")

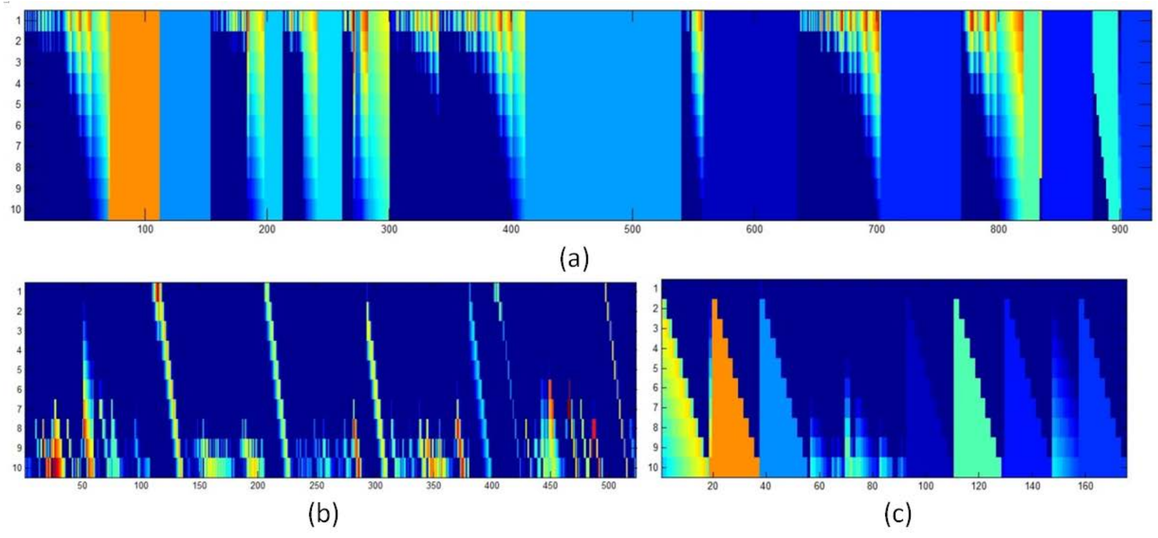


Figure 4. 3 Data collected from the sensor in the right side ((a) data marked as "Wall", (b) data marked as "Corner", (c) data marked as "LW")

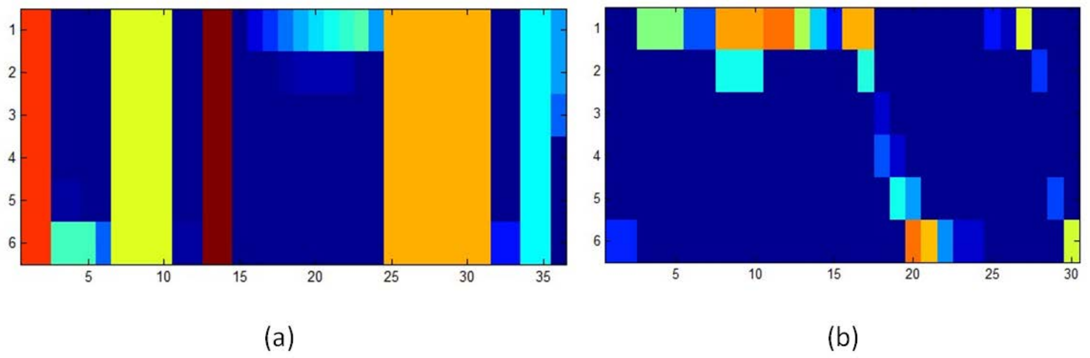


Figure 4. 4 Data collected from the sensor in the front side ((a) data marked as “Wall”, (b) data marked as “Corner”)

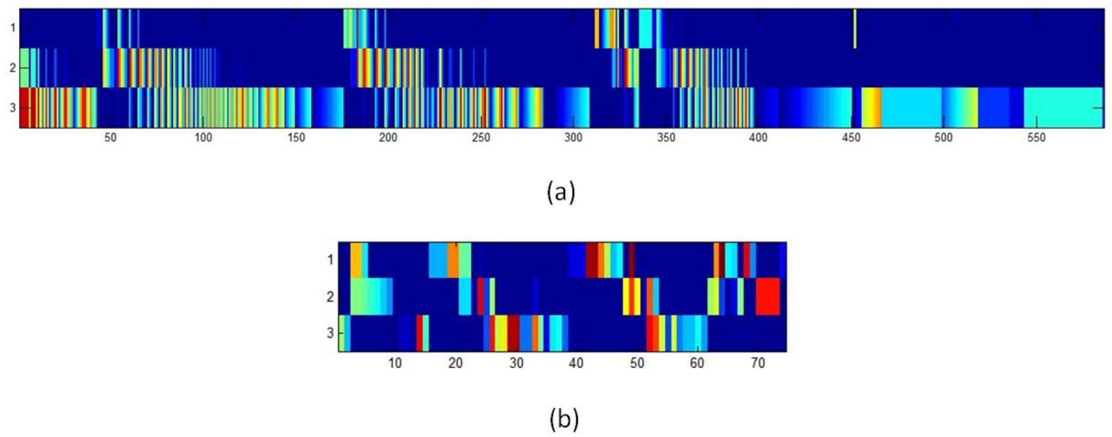


Figure 4. 5 Data collected from the sensor in the left front corner ((a) data marked as “Wall”, (b) data marked as “Corner”)

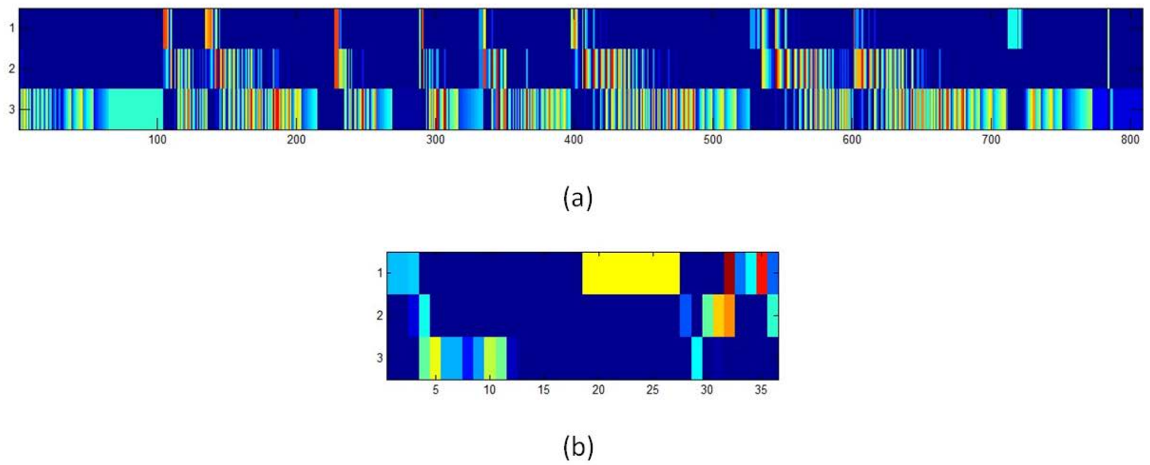


Figure 4. 6 Data collected from the sensor in the right front corner ((a) data marked as “Wall”, (b) data marked as “Corner”)

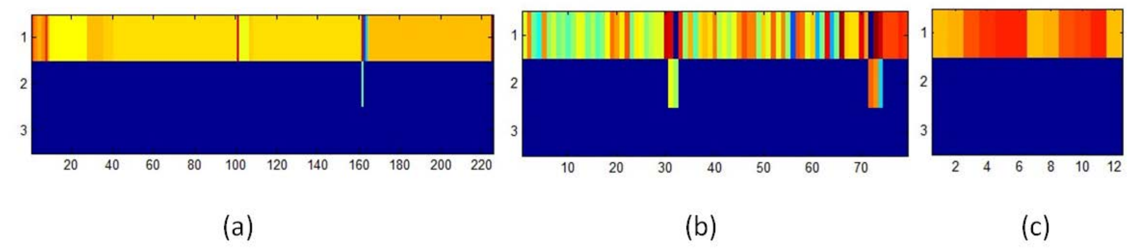


Figure 4. 7 Data collected from the sensor in the left rear corner ((a) data marked as "Wall", (b) data marked as "Corner", (c) data marked as "LW")

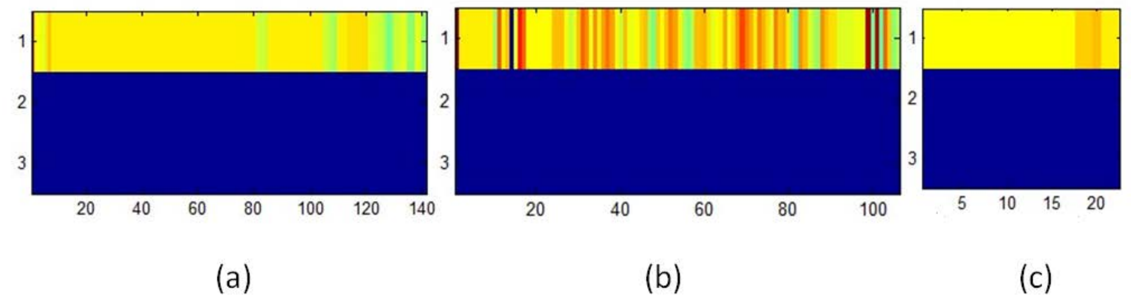


Figure 4. 8 Data collected from the sensor in the right rear corner ((a) data marked as "Wall", (b) data marked as "Corner", (c) data marked as "LW")

As it is shown in these figures, the data collected from some sensors has obvious difference between different features, such as the sensor in the right and left side. However, the situation does not always appear. For example, the data collected from the sensors in the left rear corner and the right rear corner looks like similar. Some methods need to be found to classify the sensor data.

4.3 Features Identification using Gaussian Classifier

4.3.1 Algorithm Description

Gaussian classifier is one of the basic algorithms for classification [32], [33], which is introduced as follows:

For a D-dimensional vector $\vec{x}$, the multivariate Gaussian distribution takes the form:

$$\aleph(\vec{x}|\vec{\mu},\sum) = \frac{1}{(2\pi)^{d/2}|\sum|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1}(\vec{x}-\vec{\mu})\right)$$

with mean vector $\vec{\mu}$ and covariance matrix $\sum$. $|\sum|$ denotes the deteminant of $\sum$.

The Gaussian parameters $\vec{\mu}$ and $\sum$ are estimated using the training data $(\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \cdots, \overrightarrow{x_N})$:

$$\vec{u} = \frac{1}{N}\sum_{n=1}^{N} \overrightarrow{x_n}$$

$$\sum = \frac{1}{N-1} \sum_{n=1}^{N} (\overrightarrow{x_n} - \vec{u})(\overrightarrow{x_n} - \vec{u})^T$$

After building the Gaussian distribution for training data, the features of the testing data can be predicted by the Bayes theorem. Take the “Wall” data for example.

$$L = \ln \frac{P(\text{Wall}|X = \vec{x})}{P(\text{NoWall}|X = \vec{x})}$$

$$= -\frac{1}{2}((\vec{x} - \overrightarrow{\mu_W})^T \textstyle\sum_W^{-1} (\vec{x} - \overrightarrow{\mu_W}) - (\vec{x} - \overrightarrow{\mu_{NW}})^T \textstyle\sum_{NW}^{-1} (\vec{x} - \overrightarrow{\mu_{NW}}) + \ln(|\textstyle\sum|_W) - \ln(|\textstyle\sum|_{NW})) + \ln P(W) - \ln P(NW)$$

If assume $P(W) = P(NW) = 0.5$,

$$L = -\frac{1}{2}((\vec{x} - \overrightarrow{\mu_W})^T \textstyle\sum_W^{-1} (\vec{x} - \overrightarrow{\mu_W}) - (\vec{x} - \overrightarrow{\mu_{NW}})^T \textstyle\sum_{NW}^{-1} (\vec{x} - \overrightarrow{\mu_{NW}}) + \ln(|\textstyle\sum|_W) - \ln(|\textstyle\sum|_{NW}))$$

Where W indicates “Wall”, NW indicates “NoWall”.

If L>0, the feature of vector $\vec{x}$ is “Wall”. If L<0, the feature of vector $\vec{x}$ is “NoWall”.

4.3.2 Classification Results

According to the algorithm described above, the results of predicting the features of the testing data are:

Table 4. 1 "Wall" classifier of the data collected from the sensor in the front side

	Wall (true)	No Wall (true)
Wall (predict)	34	5
No Wall (predict)	2	25

*** In these table, # (predict) raw is filled in the number of the data whose features are predicted as #. # (true) column is filled in the number of the data whose features are true #.**

Table 4. 2 "Corner" classifier of the data collected from the sensor in the front side

	Corner (true)	No Corner (true)
Corner (predict)	25	2
No Corner (predict)	5	34

Table 4. 3 "Wall" classifier of the data collected from the sensor in the right front corner

	Wall (true)	No Wall (true)
Wall (predict)	672	8
No Wall (predict)	136	24

Table 4. 4 "Corner" classifier of the data collected from the sensor in the right front corner

	Corner (true)	No Corner (true)
Corner (predict)	24	136
No Corner (predict)	8	672

Table 4. 5 "Wall" classifier of the data collected from the sensor in the left front corner

	Wall (true)	No Wall (true)
Wall (predict)	480	28
No Wall (predict)	106	46

Table 4. 6 “Corner” classifier of the data collected from the sensor in the left front corner

	Corner (true)	No Corner (true)
Corner (predict)	46	106
No Corner (predict)	28	480

Table 4. 7 “Wall” classifier of the data collected from the sensor in the left side

	Wall (true)	No Wall (true)
Wall (predict)	656	107
No Wall (predict)	16	394

Table 4. 8 “Corner” classifier of the data collected from the sensor in the left side

	Corner (true)	No Corner (true)
Corner (predict)	331	43
No Corner (predict)	65	734

Table 4. 9 “LW” classifier of the data collected from the sensor in the left side

	Leaving Wall (true)	No LW (true)
Leaving Wall (predict)	18	13
No LW (predict)	89	1055

Table 4. 10 “Wall” classifier of the data collected from the sensor in the right side

	Wall (true)	No Wall (true)
Wall (predict)	523	101
No Wall (predict)	11	324

Table 4. 11 “Corner” classifier of the data collected from the sensor in the right side

	Corner (true)	No Corner (true)
Corner (predict)	291	19
No Corner (predict)	98	551

Table 4. 12 “LW” classifier of the data collected from the sensor in the right side

	Leaving Wall (true)	No LW (true)
Leaving Wall (predict)	18	18
No LW (predict)	69	854

Table 4. 13 “Wall” classifier of the data collected from the sensor in the right rear corner

	Wall (true)	No Wall (true)
Wall (predict)	4	22
No Wall (predict)	221	69

Table 4. 14 “Corner” classifier of the data collected from the sensor in the right rear corner

	Corner (true)	No Corner (true)
Corner (predict)	66	227
No Corner (predict)	13	10

Table 4. 15 “LW” classifier of the data collected from the sensor in the right rear corner

	Leaving Wall (true)	No LW (true)
Leaving Wall (predict)	0	7
No LW (predict)	12	297

Table 4. 16 “Wall” classifier of the data collected from the sensor in the left rear corner

	Wall (true)	No Wall (true)
Wall (predict)	8	8
No Wall (predict)	133	120

Table 4. 17 “Wall” classifier of the data collected from the sensor in the left rear corner

	Corner (true)	No Corner (true)
Corner (predict)	95	154
No Corner (predict)	11	9

Table 4. 18 “Wall” classifier of the data collected from the sensor in the left rear corner

	Leaving Wall (true)	No LW (true)
Leaving Wall (predict)	22	211
No LW (predict)	0	46

4.3.3 Performance Evaluation

To evaluate the performance of the Gaussian classifier algorithm, the correct rates are computed as the percent of correctly identifications over all predictions (see Table 4.19). The cells with green background show that the Gaussian classifier algorithm does not perform well in these cases.

Table 4. 19 Performance of the Gaussian Classifier Algorithm

	Wall	No Wall	Corner	No Corner	LW	No LW
Front Side	87.1%	92.6%	92.6%	87.1%		
Right front corner	98.8%	15%	15%	98.8%		
Left front corner	94.5%	30.3%	30.3%	94.5%		
Right Side	83.8%	96.7%	93.9%	84.9%	50.0%	92.5%
Left Side	86.0%	96.1%	88.5%	91.9%	58.1%	92.2%
Right rear corner	15.4%	23.8%	22.5%	43.5%	0%	96.1%
Left rear corner	50.0%	47.4%	38.2%	45%	9.4%	100%

Some results show the situation about over-occupancy. For example, “Wall” and “No Wall” is a pair of features. In the right front corner case, the reason of high correct rate of “Wall” is that most of the data in the cross area of “Wall” and “No Wall” are determined as “Wall”. In other word, the high correct rate of “Wall” is obtained by sacrificing the correct rate of “No Wall”. The basic reason behind this appearance is that the Gaussian distribution is intrinsically unimodal. If the data labeled as “A” follow a Gaussian distribution, as well as the data labeled as “B”, the Gaussian classifier model would work well (see Figure 4.9 (a)). However, if there are many clumps in the data labeled as a feature, a single Gaussian distribution cannot be fitted to the data well and would fail to capture the clumps in the data (see Figure 4.9 (b)). If two Gaussian distributions are close, over-occupancy would appear (see Figure 4.9 (c)).

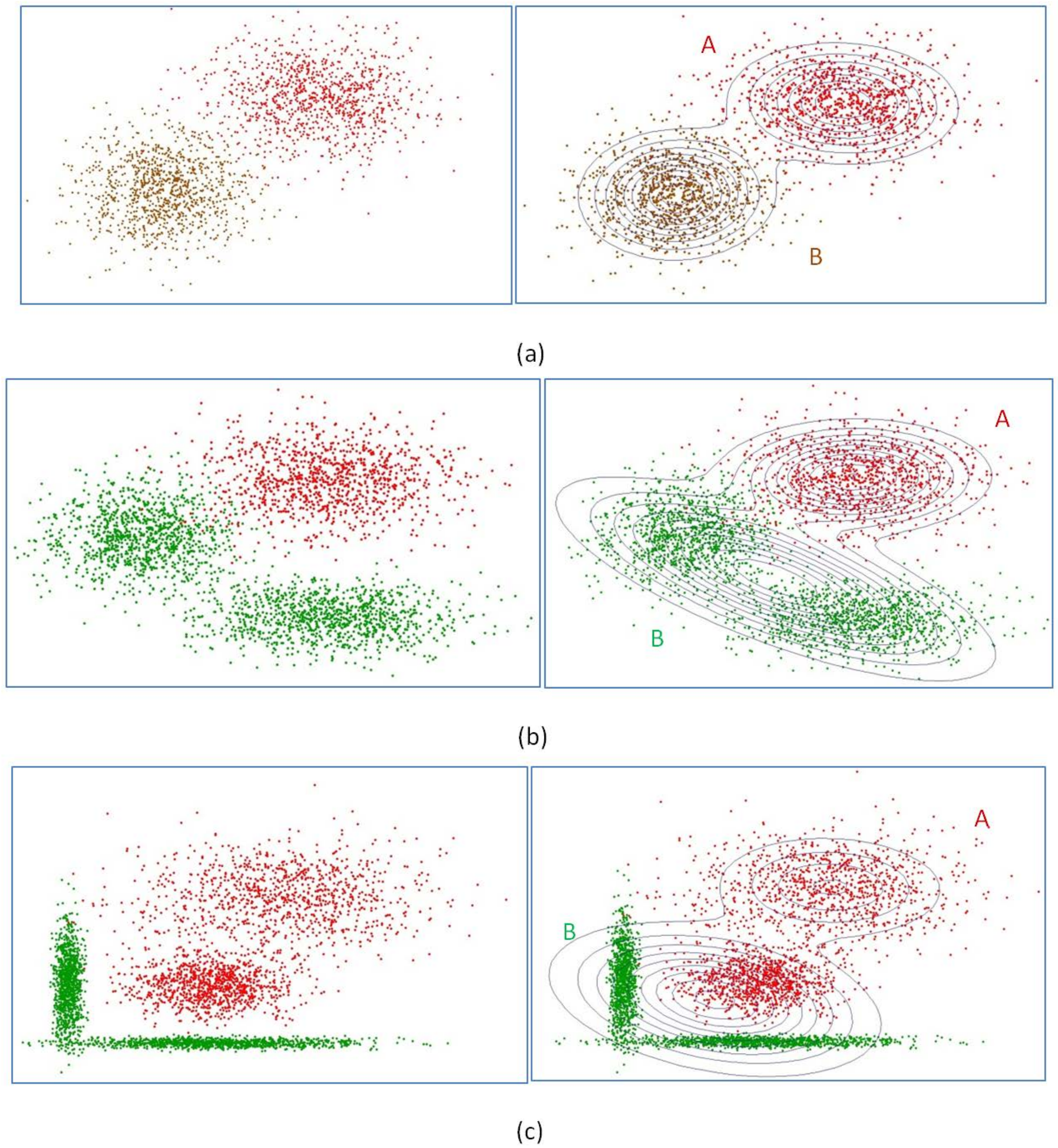


Figure 4. 9 Over-Occupancy in Gaussian Classifier (left part is raw data, right part is classified data)

The data of the sensor in the left front corner are shown in Figure 4.10.

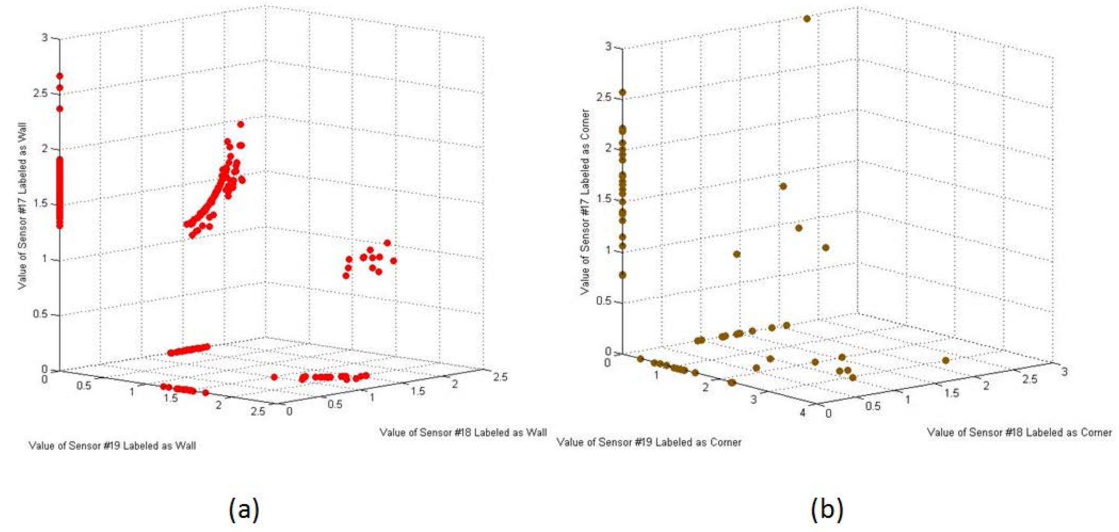
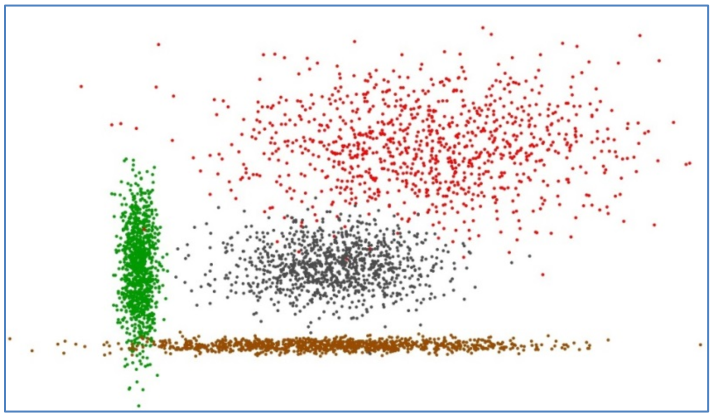
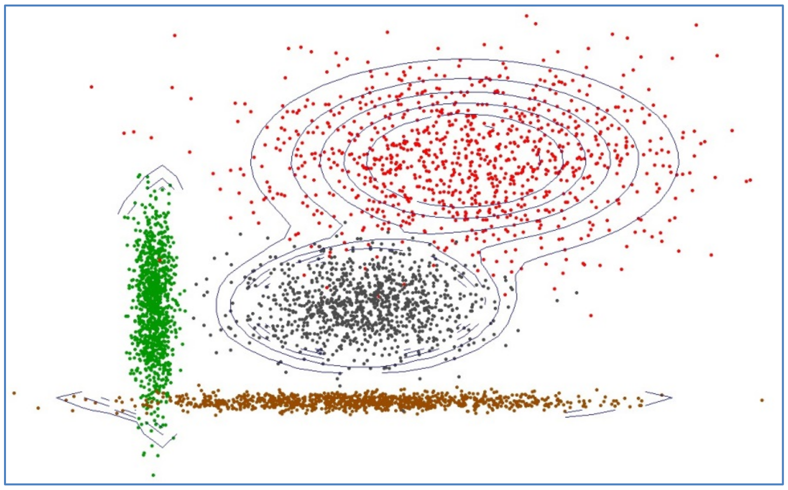


Figure 4. 10 Data collected from the sensors in the right front corner ((a) data marked as "Wall", (b) data marked as "Corner")

As you can see, if the "Wall" data and "Corner" data are only considered as two clumps of data which follow the Gaussian distribution, the over-occupancy would happen. To solve this problem, multiple Gaussian distributions can be built. Take the case in figure as example. At first, assume that there are four clumps in the data. Then, build the Gaussian distributions for these four clumps. Finally, classify the Gaussian distributions. This process is shown in Figure 4.11.



(a)


(b)

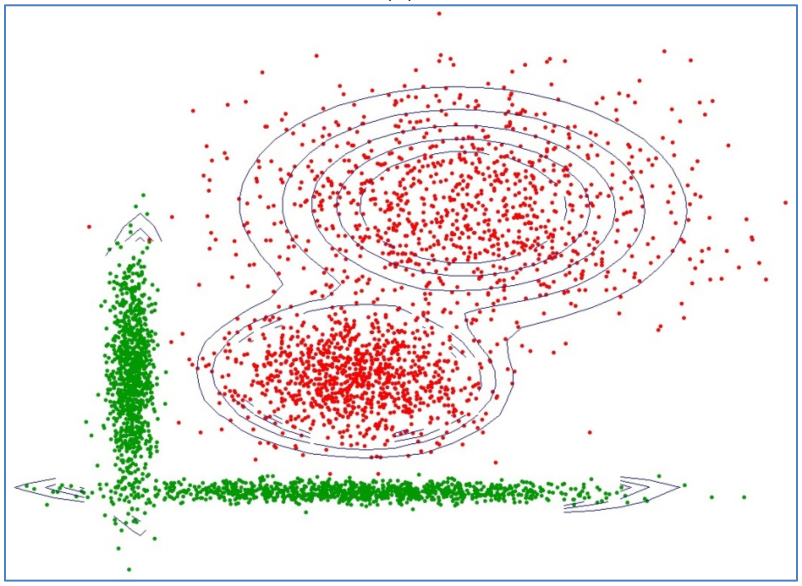

(c)

Figure 4. 11 Method for Handling Over-Occupancy

As it is shown, if the number of the Gaussian densities is accurate, the correct rate would be increased. However, sometimes, it is hard to estimate accurately that how many clumps are in the data. The Gaussian mixture model algorithm is used for solving the problem.

4.4 Feature Identification using Gaussian Mixture Model (GMM)

GMM is very useful for classification. A classic application is text-independent speaker identification [34].

4.4.1 Algorithm Description

A Gaussian mixture density is a weighted sum of M component densities, which take the form

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i * \aleph_i(\vec{x}) \text{ , where}$$

(i) $\vec{x}$ is a D – dimensional vector

(ii) $\aleph_i(\vec{x}), i = 1, \dots, M,$ are the component densities given by the equation

$$\aleph_i(\vec{x}) = \frac{1}{(2\pi)^{d/2}|\sum_i|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu_i})^T {\sum_i}^{-1}(\vec{x} - \vec{\mu_i})\right)$$

with mean vector $\vec{\mu}$ and covariance matrix $\sum$.

$|\sum|$ denotes the deteminant of $\sum$.

(iii) $p_i, i = 1, \dots, M,$ are the mixture weights,

which satisfy the constraint that $\sum_{i=1}^{M} p_i = 1.$

(iv) λ denotes the set of the parameters.

The Gaussian mixture density can be depicted as the diagram in Figure 4.12. The basic idea in GMM is that: if there are three clumps in the data labeled as “A”, whose densities are similar to $\aleph_1, \aleph_2$ and $\aleph_4$, the weight p_1, p_2 and p_4 are high in the GMM for feature “A” and other weights are low. As a result, if $\vec{x}$ is in the data labeled as “A”, it would have a high value of $p(\vec{x}|\lambda)$. The complete Gaussian mixture density is parameterized by the mean vectors $\vec{\mu}$, covariance matrices $\sum$ and mixture weights. However, for the data collected in the experiment, there is no clue to determine these parameters directly.

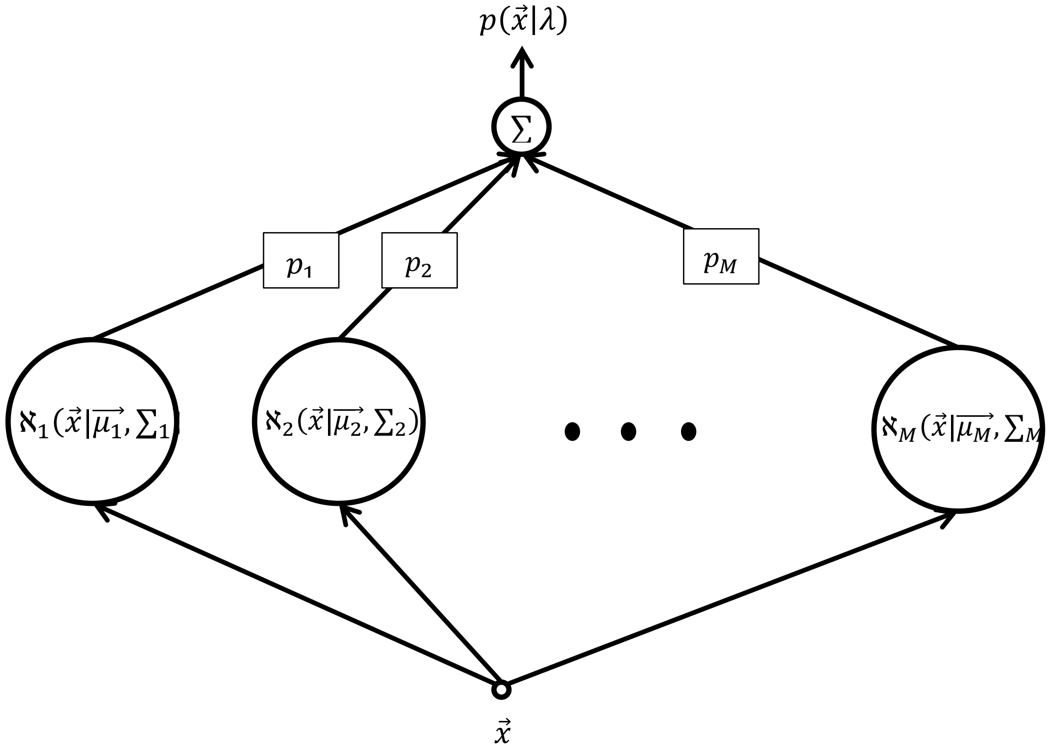

$p(\vec{x}|\lambda)$
$\sum$
p_1
p_2
p_M
$\aleph_1(\vec{x}|\overrightarrow{\mu_1},\sum_1)$
$\aleph_2(\vec{x}|\overrightarrow{\mu_2},\sum_2)$
$\aleph_M(\vec{x}|\overrightarrow{\mu_M},\sum_M)$
$\vec{x}$

Figure 4. 12 An M Component Gaussian Mixture Density.

The expectation-maximization (EM) algorithm [35] is used for estimating the parameter in this article. The idea of EM algorithm is using the training data to refit the parameters of the components iteratively until each component is fitted to the entire data set. The details of the EM algorithm are shown as follows:

E-step:

The training data is $\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \ \ldots, \overrightarrow{x_t}, \ldots, \overrightarrow{x_T}$.

Compute the probability $p(i|\ \overrightarrow{x_t}, \lambda)$, which means the probability that the vector $\overrightarrow{x_t}$ was generated by component i.

By Bayes' rule, we have

$$p(i|\,\overrightarrow{x_t},\lambda) = p_i * \frac{\aleph_i(\overrightarrow{x_t})}{\sum_{k=1}^{M} p_k * \aleph_k(\overrightarrow{x_t})} \text{ , where}$$

(i) $\aleph_i(\overrightarrow{x_t})$ is the probability at $\overrightarrow{x_t}$ of the ith Gaussian,

(ii) $p_i$is the weight parameter for the ith Gaussian,

(iii) $\lambda = \{p_i, \overrightarrow{u_i}, \Sigma_i\}, i = 1,2,\dots, M\ are\ the\ parameters\ set\ of\ this\ GMM.$

M-step:

Compute the new mean, covariance, and component weights using the following steps in sequence:

$$\overrightarrow{\overline{\mu_i}} = \sum_{t=1}^{T} \frac{p(i|\,\overrightarrow{x_t},\lambda) * \overrightarrow{x_t}}{\sum_{t=1}^{T} p(i|\,\overrightarrow{x_t},\lambda)} \text{ ,}$$

$$\overline{\sigma}_i^2 = \sum_{t=1}^{T} \frac{p(i|\,\overrightarrow{x_t},\lambda) * \left(\overrightarrow{x_t} - \overrightarrow{\overline{\mu_i}}\right) * (\overrightarrow{x_t} - \overrightarrow{\overline{\mu_i}})^T}{\sum_{t=1}^{T} p(i|\,\overrightarrow{x_t},\lambda)} \text{ ,}$$

$$\overline{p}_i = \frac{1}{T} \sum_{t=1}^{T} p(i|\,\overrightarrow{x_t},\lambda)$$

Two critical factors in training a GMM are selecting the order M of the mixture and initializing the model parameters.

Each feature has a GMM. For data classification, a group of F features $F = \{1, 2, \dots, F\}$ is represented by GMM's $\lambda_1, \lambda_2, \dots, \lambda_F$. The objective is to find the feature model which has the maximum posteriori probability for a testing data. Formally, due to Bayes rule,

$$\hat{F} = \arg \max_{1 \le k \le F} P_r(\lambda_k|X) = \arg \max_{1 \le k \le F} \frac{p(X|\lambda_k) * P_r(\lambda_k)}{p(X)}$$

Assuming that $P_r(\lambda_k) = \frac{1}{F}$, where $k = 1, \dots, F$, and $p(X)$ is the same for all data, the classification rule simplifies to

$$\hat{F} = \arg \max_{1 \le k \le F} P_r(X|\lambda_k)$$

The feature of the data is determined by the maximum value of the probabilities.

4.4.2 Data Analysis by GMM

4.4.2.1 Initialization

As stated in the previous section, the GMM training procedure should be initialized with some starting parameters, such as the mean vectors of the Gaussians, the covariance matrices of the Gaussians and the mixture weights.

For the mean vectors, they are chose randomly from the training data in this project. However, because of the random strategy, some significant features may not be selected, which would prevent the convergence of the training procedure. Therefore, a maximum number of iteration is set, which is 100. If the procedure isn't able to complete in 100 iterations, the initial mean vectors need to be reselected.

Depending on the choice of the covariance matrices, the GMM would have different forms. For example, the model can have one covariance matrix per Gaussian component (nodal covariance), or one covariance matrix shared by all Gaussian components (grand covariance). The covariance matrix can also be diagonal or full. There are no good theoretical methods to guide the selection of the form of the covariance matrix, so it is best to be determined according to the experiment results. When training the GMM, another problem may encounter, which is that the variance elements can become quite small. The small variance may produce a singularity in the model's likelihood function and degrade the classification performance. This situation particularly appears in the GMM with a large number of components. To avoid the singularity, a variance minimum constraint

$$\overline{\sigma}_i^2 = \begin{cases} \sigma_i^2 & \text{if } \sigma_i^2 > \sigma_{\min}^2 \\ \sigma_{\min}^2 & \text{if } \sigma_i^2 \leq \sigma_{\min}^2 \end{cases}$$

is applied to the variance estimates. This method has been shown to provide more robust parameter estimates than the unconstrained version [36], [37]. In the analysis procedure, two minimum constrains had been found between $\sigma_{\min}^2 = 0.1$ and $\sigma_{\min}^2 = 0.001$.

4.4.2.2 Model Order

Determining the number of components in GMM is an important but difficult problem. On the one hand, choosing too few mixture components can not accurately model the training data distribution. On the other hand, choosing too many components would reduce classification performance and increase excessive computational complexity. The appropriate order of GMM also needs to be determined depending on the experiment results. The testing numbers of components are 1, 2, 4, 8, 16, 32 and 64.

4.4.2.3 Determining Covariance Form

The number of components in this experiment was fixed as 20. The sensor data in the right side and in the left front corner were used for testing the performance of the GMMs with the different forms of covariance. The results of the experiment are shown as follows:

Table 4. 20 Performance of the GMMs of the sensor in the right side (part one)

Data: Right Side Covariance Form: Nodal-Full (All the covariance matrices are full matrices but not restricted to be the same.) Variance Minimum Constraint: 0.1 (singularity appears when there is not variance minimum constraint.)			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	99.20%	0.80%	0%
Corner(predict)	7.07%	70.71%	22.22%
LW(predict)	0%	15.24%	84.76%

Then, different variance minimum constraints were tested (see Table 4.20 – 4.22).

Table 4. 21 Performance of the GMMs of the sensor in the right side (part two)

Data: Right Side Covariance Form: Nodal-Full Variance Minimum Constraint: 0.01			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	82.48%	17.52%	0%
Corner(predict)	1.77%	90.47%	7.76%
LW(predict)	0%	30%	70%

Table 4. 22 Performance of the GMMs of the sensor in the right side (part three)

Data: Right Side Covariance Form: Nodal-Full Variance Minimum Constraint: 0.001			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	78.72%	21.28%	0%
Corner(predict)	7.07%	95.20%	3.87%
LW(predict)	0%	34.92%	65.08%

There is something interesting here. The correct rate of classification changes with the variance minimum constraint, which is shown in Figure 4.13.

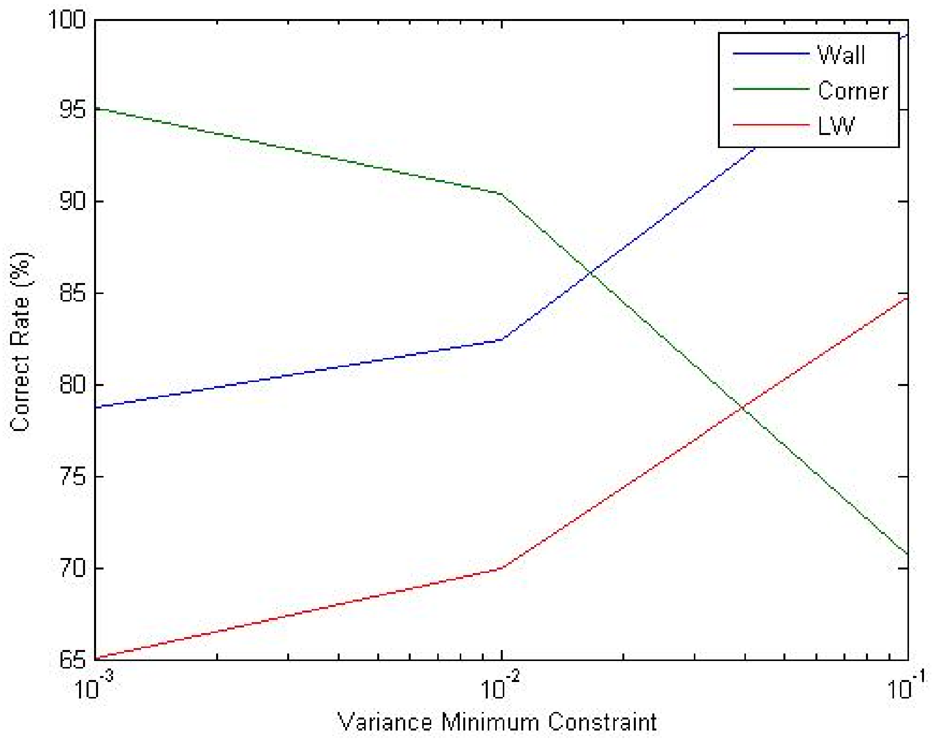


Figure 4. 13 Relationship between Variance Minimum Constraint and Correct Rate

As you can see, a variance limits between $\sigma_{min}^2 = 0.05$ and $\sigma_{min}^2 = 0.01$ would provide the best robustness for the contact features. Finally, it was found that $\sigma_{min}^2 = 0.03$ is an appropriate one (see Table 4.23).

Table 4. 23 Performance of the GMMs of the sensor in the right side (part four)

Data: Right Side Covariance Form: Nodal-Full Variance Minimum Constraint: 0.03			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	92.26%	7.74%	0%
Corner(predict)	4.55%	81.20%	14.25%
LW(predict)	0%	20%	80%

Next, the diagonal matrix was examined (see Table 4.24).

Table 4. 24 Performance of the GMMs of the sensor in the right side (part five)

Data: Right Side Covariance Form: Nodal-Diagonal (All the covariance matrices are diagonal matrices but not restricted to be the same.) Variance Minimum Constraint: 0.03			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	92.26%	7.74%	0%
Corner(predict)	4.04%	81.06%	14.90%
LW(predict)	0%	21.02%	78.98%

The performance of the GMMs with diagonal covariance matrices was almost the same as the one with full covariance matrices. Next, if all Gaussian components are shared one covariance matrix, the results were:

Table 4. 25 Performance of the GMMs of the sensor in the right side (part six)

Data: Right Side Covariance Form: Grand-Full (All the covariance matrices are full matrices and restricted to be the same) Variance Minimum Constraint: None			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	64.29%	35.71%	0%
Corner(predict)	1.01%	95.71%	3.28%
LW(predict)	0%	57.14%	42.86%

Table 4. 26 Performance of the GMMs of the sensor in the right side (part seven)

Data: Right Side Covariance Form: Grand-Diagonal (All the covariance matrices are diagonal matrices and restricted to be the same) Variance Minimum Constraint: None			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	69.94%	30.06%	0%
Corner(predict)	1.26%	93.18%	5.56%
LW(predict)	0%	60%	40%

The performances of classification in these cases were poor.

Then, the data collected from the sensor in the left front corner were trained and tested in the same manner as described above. It was found that $\sigma_{min}^2 = 0.1$ was the most appropriate variance minimum constraint. The correct rate label was:

Table 4. 27 Performance of the GMMs of the sensor in the left front corner (part one)

Data: Left Front Corner Covariance Form: Nodal-Full Variance Minimum Constraint: 0.1		
	Wall(true)	Corner(true)
Wall(predict)	87.0%	13.0%
Corner(predict)	8.33%	91.67%

The performances of the GMMs using other covariance forms were shown as follows:

Table 4. 28 Performance of the GMMs of the sensor in the left front corner (part two)

Data: Left Front Corner Covariance Form: Nodal-Diagonal Variance Minimum Constraint: 0.1		
	Wall(true)	Corner(true)
Wall(predict)	87.0%	13.0%
Corner(predict)	8.33%	91.67%

Table 4. 29 Performance of the GMMs of the sensor in the left front corner (part three)

Data: Left Front Corner Covariance Form: Grand-Full Variance Minimum Constraint: None		
	Wall(true)	Corner(true)
Wall(predict)	29.95%	70.05%
Corner(predict)	27.78%	72.22%

Table 4. 30 Performance of the GMMs of the sensor in the left front corner (part three)

Data: Left Front Corner Covariance Form: Grand-Diagonal Variance Minimum Constraint: None		
	Wall(true)	Corner(true)
Wall(predict)	44.55%	55.45%
Corner(predict)	9.63%	90.37%

As expected, the performances given for two kinds of data are similar. Thus, the best form of the covariance matrix is diagonal, nodal covariance. Different forms have their own suitable variance minimum constraint.

4.4.2.4 Determining Model Order

Choosing the diagonal and nodal covariance matrices, the GMMs with 1, 2, 4, 8, 16, 32 and 64 component densities were trained using the data of sensor in the right side. Table 4.31 shows the complete classification results.

Table 4. 31 Performances of GMMs with different components

Model Order		Feature		
		Wall	Corner	LW
1	$\sigma_{min}^2 = 0.02$	87.30%	89.91%	70.04%
	$\sigma_{min}^2 = 0.03$	87.39%	89.91%	70.29%
	$\sigma_{min}^2 = 0.04$	87.24%	89.66%	70.17%
2	$\sigma_{min}^2 = 0.02$	89.58%	88.38%	69.52%
	$\sigma_{min}^2 = 0.03$	94.43%	83.84%	70.71%
	$\sigma_{min}^2 = 0.04$	94.77%	80.24%	71.54%
4	$\sigma_{min}^2 = 0.02$	83.78%	91.67%	69.52%
	$\sigma_{min}^2 = 0.03$	95.98%	88.38%	70.48%
	$\sigma_{min}^2 = 0.04$	92.11%	92.17%	70.48%
8	$\sigma_{min}^2 = 0.02$	85.86%	87.37%	79.05%
	$\sigma_{min}^2 = 0.03$	88.54%	81.57%	82.86%
	$\sigma_{min}^2 = 0.04$	89.58%	91.67%	70.48%
16	$\sigma_{min}^2 = 0.02$	84.23%	88.64%	75.24%
	$\sigma_{min}^2 = 0.03$	90.77%	83.84%	80.00%
	$\sigma_{min}^2 = 0.04$	94.35%	78.03%	82.86%
32	$\sigma_{min}^2 = 0.02$	87.65%	86.62%	71.43%
	$\sigma_{min}^2 = 0.03$	89.29%	85.10%	75.24%
	$\sigma_{min}^2 = 0.04$	98.96%	77.27%	80.00%
64	$\sigma_{min}^2 = 0.02$	87.20%	86.87%	71.43%
	$\sigma_{min}^2 = 0.03$	94.94%	83.59%	74.29%
	$\sigma_{min}^2 = 0.04$	98.96%	76.77%	77.14%

There are several observations to be made from these results. First, the relationship between the variance minimum constraint and the correct rate is similar to the situation described above, and the best performance is given when the variance minimum constraint is around 0.03. Second, Figure 4.14 shows the average, lowest and highest percent correct classification versus the number of Gaussian components.

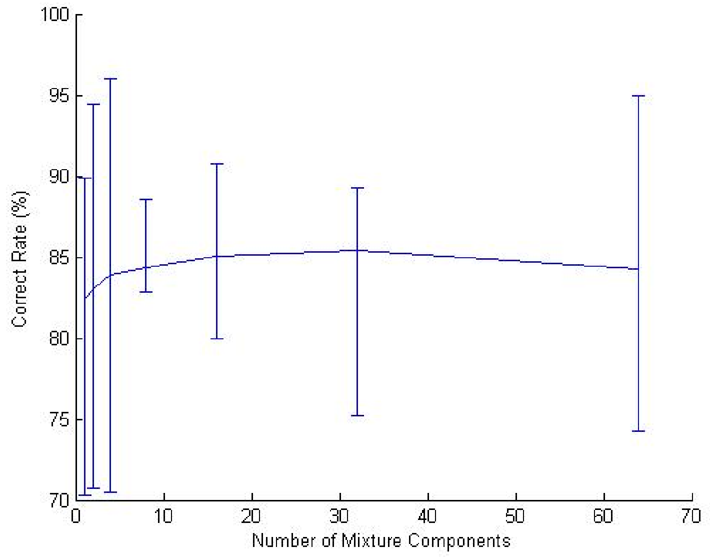


Figure 4. 14 Relationship between the Number of Mixture Components and the Correct Rate

The increase in classification performance from 1 to 8 mixture components, and leveling off above 16 components, indicates that there is a lower limit in the number of mixture components necessary to adequately model the contact features. At the same time, the variation range is small when there are 8 to 32 components. The small variation range indicates that there isn’t a feature whose classification performance is very pool and averaged by a feature having high performance. Therefore, Models must contain at least 8 components and no more than 32 components to maintain good classification performance.

At last, implementing the same methods to processing the sensor data in each part, the complete performance results are shown as follows:

Table 4. 32 Performance of the GMMs of the sensor in the front side

Data: Front Side Covariance Form: Nodal-Diagonal Variance Minimum Constraint: 0.1 Order of Model:16		
	Wall(true)	Corner(true)
Wall(predict)	86.11%	13.89%
Corner(predict)	16.67%	83.33%

Table 4. 33 Performance of the GMMs of the sensor in the left front corner

Data: Left Front Corner Covariance Form: Nodal-Diagonal Variance Minimum Constraint: 0.1 Order of Model:16		
	Wall(true)	Corner(true)
Wall(predict)	87.32%	12.68%
Corner(predict)	8.33%	91.67%

Table 4. 34 Performance of the GMMs of the sensor in the right front corner

Data: Right Front Corner Covariance Form: Nodal-Diagonal Variance Minimum Constraint: 0.1 Order of Model: 16		
	Wall(true)	Corner(true)
Wall(predict)	90.24%	9.76%
Corner(predict)	11.28%	88.72%

Table 4. 35 Performance of the GMMs of the sensor in the left side

Data: Left Side Covariance Form: Nodal-Full Variance Minimum Constraint: 0.03 Order of Model: 20			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	90.77%	9.23%	0%
Corner(predict)	4.04%	83.26%	12.7%
LW(predict)	0%	20.43%	79.57%

Table 4. 36 Performance of the GMMs of the sensor in the right side

Data: Right Side Covariance Form: Nodal-Full Variance Minimum Constraint: 0.03 Order of Model: 20			
	Wall(true)	Corner(true)	LW(true)
Wall(predict)	92.26%	7.74%	0%
Corner(predict)	4.55%	81.20%	14.25%
LW(predict)	0%	20%	80%

Table 4. 37 Performance of the GMMs of the sensor in the left rear corner

Data: Left Rear Corner Covariance Form: Nodal-Diagonal Variance Minimum Constraint: 0.1 Order of Model:20		
	Wall(true)	Corner(true)
Wall(predict)	90.24%	9.76%
Corner(predict)	43.59%	56.41%

Table 4. 38 Performance of the GMMs of the sensor in the right rear corner

Data: Right Rear Corner Covariance Form: Nodal-Diagonal Variance Minimum Constraint: 0.1 Order of Model:20		
	Wall(true)	Corner(true)
Wall(predict)	89.08%	10.92%
Corner(predict)	47.27%	52.73%

4.4.2.5 Error Analysis

From the final contact feature models, it is found that most errors appeared when there was only one sensor contacting with object. For example, using 1 to represent that the sensor has contact signal and 0 to represent that the sensor has no contact signal, the classification results of the "LW" data of sensor in the right side are:

Table 4. 39 Classification results of the “LW” data of sensors in the right side

Contact Situation	Corner(predict)	LW(predict)
0000000001	156	26
0000000011	106	27
0000000111	33	20
0000001111	14	18
0000011111	6	16
0000111111	3	18

As you can see, when the number of the contacting sensors was limited, most of the “LW” data had been recognized as “Corner” data. This situation also appears when classifying the data in the rear corner, since there is only one sensor has contacting signal in this case. Actually, the problem is also hard for humans.

Another reason for the error is the limited data. There are only 22 “LW” data collected from the sensors in the left rear corner (see Table 4.18) and 12 “LW” data collected from the sensors in the right rear corner (see Table 4.15). It is hard to build an appropriate model according to the insufficient data.

4.5 Hole Detection

Among the features “Wall”, “Corner” and “LW”, the car can also detect the hole according to the tactile feedback. If there is a hole, the width of the hole has to be detected for deciding if the car can moved into it. The basic idea is to compute the width according to the geometric shape of the car. The specific steps are:

1. Find out all the fragments of the contacted data.
2. The hole is the space between two fragments.
3. The width of the hole is the distance between the last sensor in the previous fragment and the first sensor in the current fragment (see Figure 4.15).

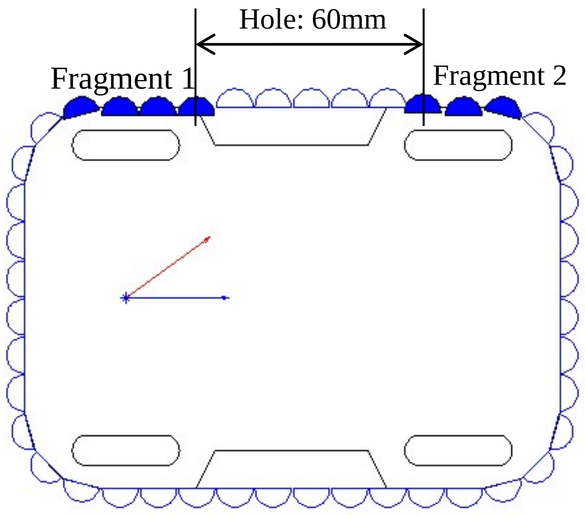


Figure 4. 15 Hole Detection Method

4.6 Conclusion

We can use the data to learn such features as “Wall”, “Corner”, “LW” and “Hole”. It is found that, in some cases, the feature was difficult to identification. To increase the accuracy of the feature classification, a checking process can be designed, since it is not necessary to make right classification immediately after first contact. The simple method is counting the number of times a feature appeared continuously or adding the probabilities obtained from the GMMs in each step. A threshold needs to been set for making decision. More precisely, a Bayesian network can be designed, which can deal with the uncertainty in the classification process. Currently, in the application which described in next Chapter, the first method is used.

Chapter 5

Application

We want to use the data to module the robot’s behavior. However, from the data processing results, we learn that is difficult. Thus, we plan to design autonomous navigation algorithm from the start with simple cases in a control environment.

5.1 Application Description

A simple navigation problem was designed for examining the effect of the feature classification model which had been obtained in the previous chapter. The scene of the problem is shown in the Figure 5.1. A wall blocks the car’s path to the destination. There are a hole, a door, a narrow path and bumps in the wall.

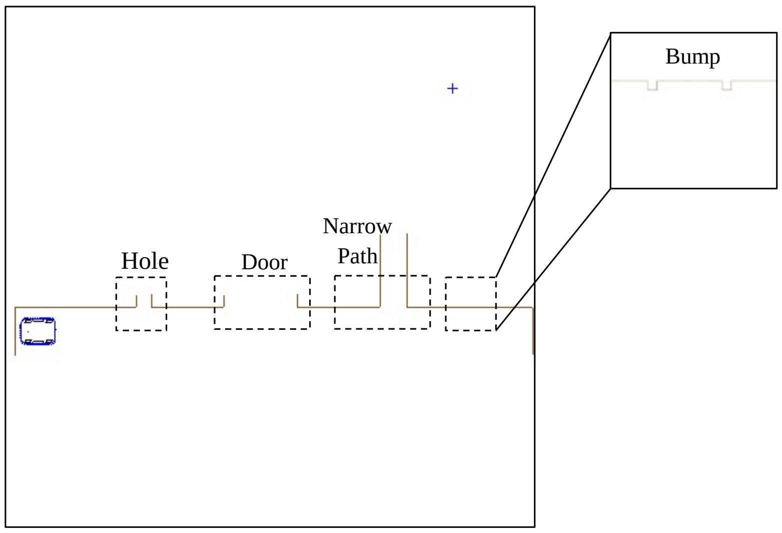


Figure 5. 1 The scene of the navigation problem

The objectives of the problem are:

- Drive the car automatically to approach the destination.
- Check the whole wall.
- Find the path to go through the wall.

5.2 Control Strategy

To accomplish these objectives, some control strategies based on the contact status are built, such as “go straight”, “head to destination”, “check contact”, “follow wall”, “move through wall” and “U turn”. The details of these strategies are introduced as follows:

1. Go Straight

   Prerequisites: none

   Stop conditions: The car is blocked and cannot move straight anymore, or the contact status is “Wall”.

   Rule: Move straight.

2. Head to Destination

   Prerequisites: There is a destination.

   Stop conditions: The car contacts with an object, the car approaches the destination, or the car is blocked and cannot follow the control rule.

   Rule: First, rotate the car till it faces the destination. Second, move straight.

3. Check Contact

   Prerequisites: There is a contact.

   Stop conditions: The contact status can be determined as “Wall” or “Corner”.

Rule: First, move straight. Second, when the car cannot move straight anymore, rotate a slight angle and then move straight again. Third, repeat step one and step two till the stop conditions appear.

4. Follow Wall

   Prerequisites: the contact status is "Wall".

   Stop conditions: the contact status is not "Wall" or the car is blocked and cannot follow the control rule.

   Rule: keep the sensors in a side contacting with the wall. If the maximum value of the data is greater than 4 (the upper bound of the data is 5) or the mean value of the contacting data is greater than 3, turn a slight angle to reduce these value. If the maximum value of the data is less than 0.5 of the mean value of the contacting data is less than 2, turn a slight angle to increase these value. In other situations, move straight.

5. Move through Wall

   Prerequisites: the contact status is "Hole" and the width of the hole is greater than 100.

   Stop conditions: there is a contact or the car is blocked and cannot follow the control rule.

   Rule: First, move the car to face the hole. Second, move in to the hole.

6. U Turn

   Prerequisites: the front contact status is "Wall".

   Stop conditions: the car rotates 180 degree or the car is blocked and cannot follow the control rule.

   Rule: If the sensor in left side has contact, turn right 180 degree. If the sensor in right side has contact, turn left 180 degree.

5.3 Experiment

5.3.1 Experiment Panel

The scene of the problem, the status of the car, the control strategies and the sensor data can be shown in the experiment panel (see Figure 5.3). Comparing with the previous GUIs, there are two new parts. Part 1 shows all the control strategies and the one being used currently. Part 2 shows the statuses of the car part by part and the integrated result.

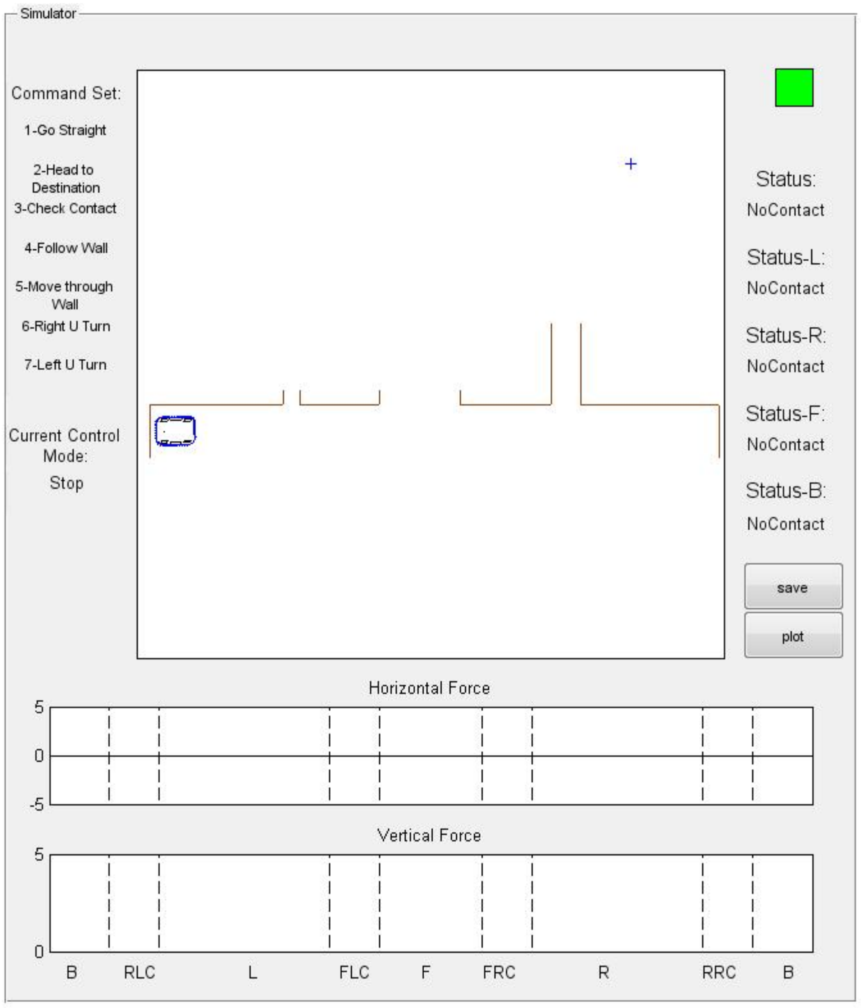


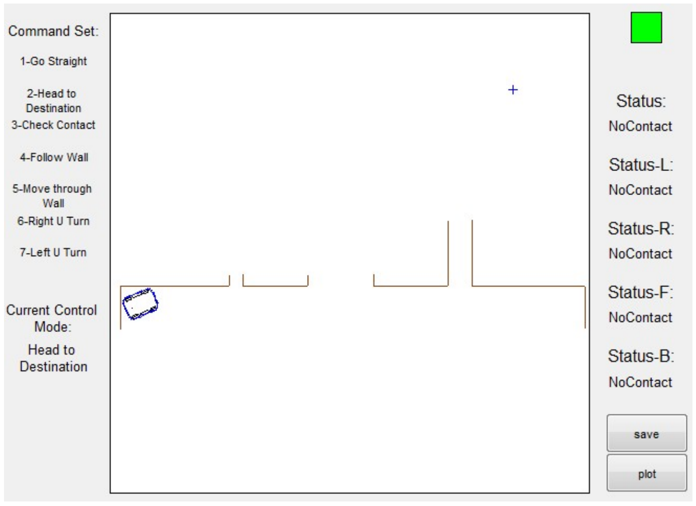
Figure 5. 2 Experiment Panel GUI

5.3.2 Experiment Result

Using the control strategies defined above, the car can be drove to approach the destination following the process below.

Stage one

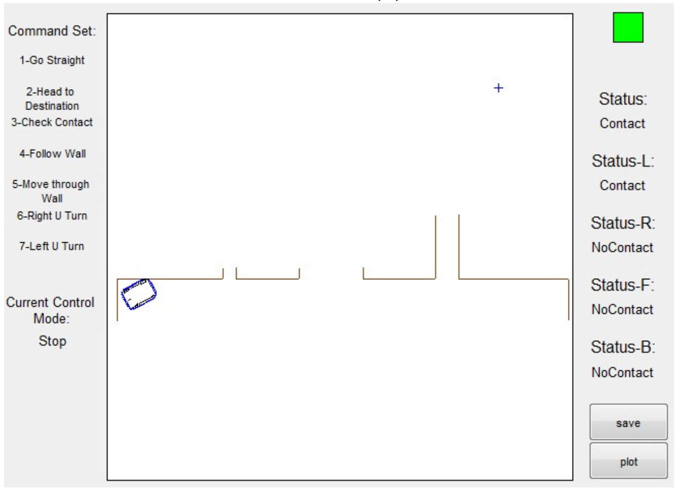
When there is no contact with the car, the car would head to the destination and not stop until contact appears. (see Figure 5.4)



Current Control Mode:
Head to Destination

Status:
NoContact
Status-L:
NoContact
Status-R:
NoContact
Status-F:
NoContact
Status-B:
NoContact

(a)



Current Control Mode:
Stop

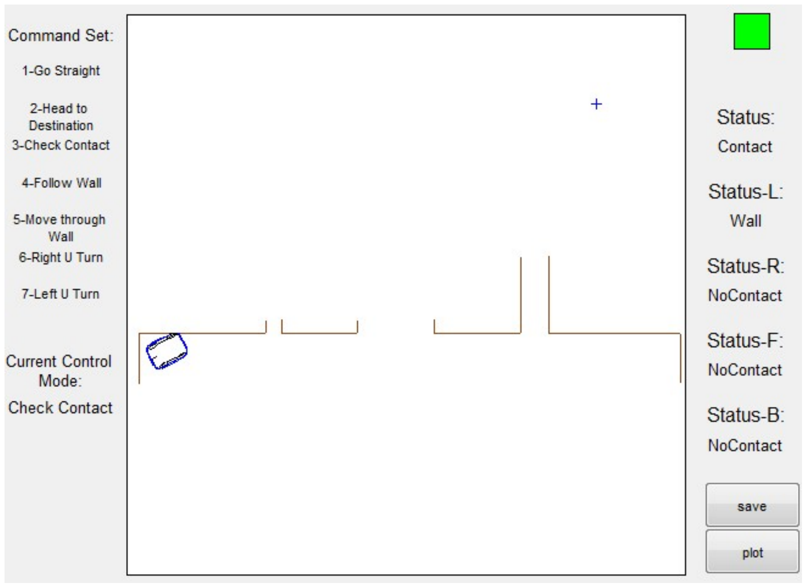
Status:
Contact
Status-L:
Contact
Status-R:
NoContact
Status-F:
NoContact
Status-B:
NoContact

(b)

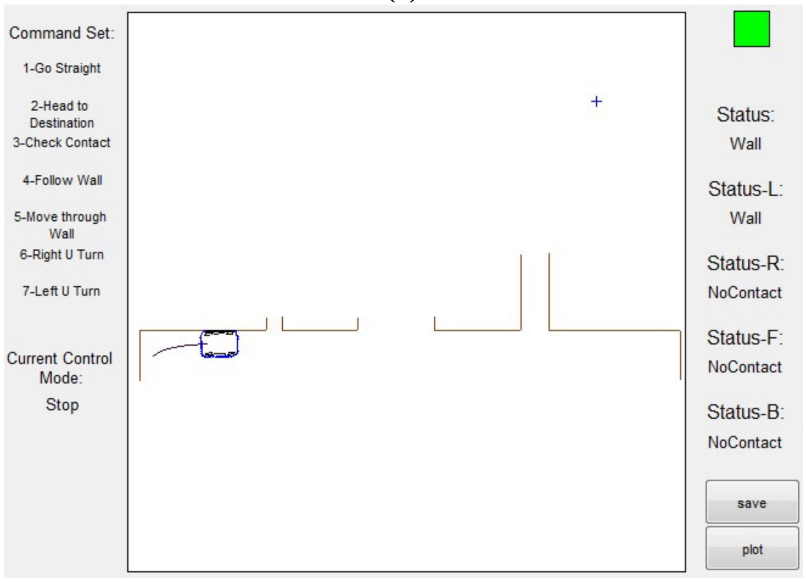
Figure 5. 3 Process of Heading to Destination ((a) in progress (b)stop)

Stage two

When there is a contact with the car, the feature of the object need to be detected. The car is in the "Check Contact" Mode. It would not stop until the specific feature is determined.(see Figure 5.5)



(a)

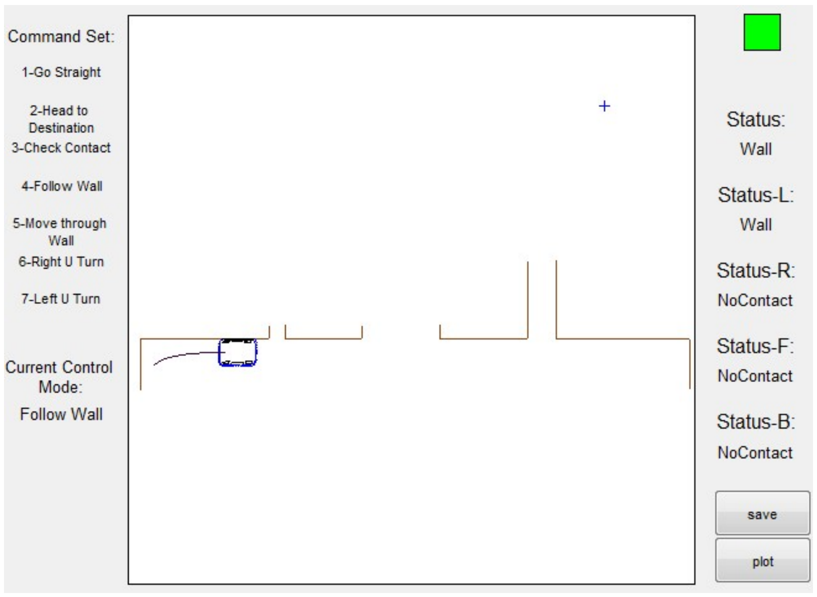


(b)

Figure 5. 4 Process of Checking Contact ((a) in progress (b)stop)

Stage three

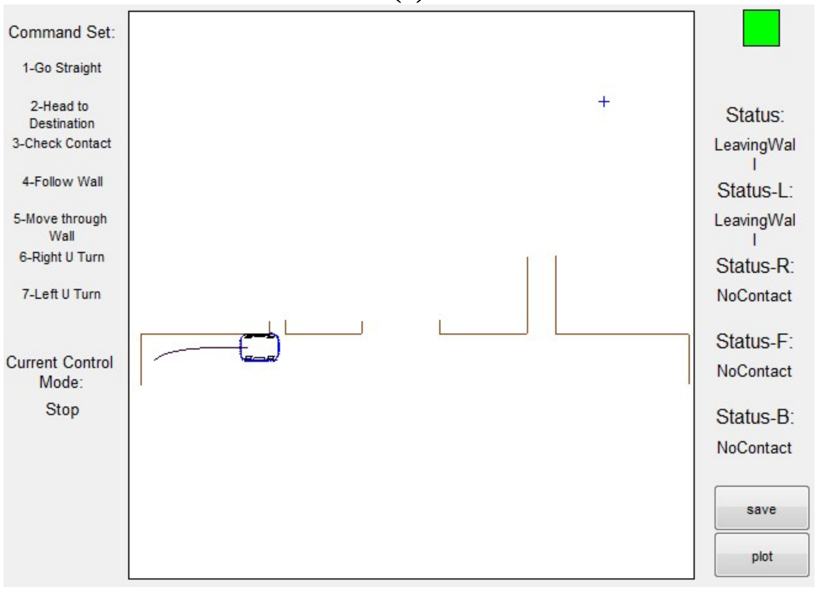
When the status is checked as “Wall”, the car would follow the wall. When the status changes to “LeavingWall”, the car stops. (see Figure 5.6)



Current Control Mode: Follow Wall

Status: Wall
Status-L: Wall
Status-R: NoContact
Status-F: NoContact
Status-B: NoContact

(a)



Current Control Mode: Stop

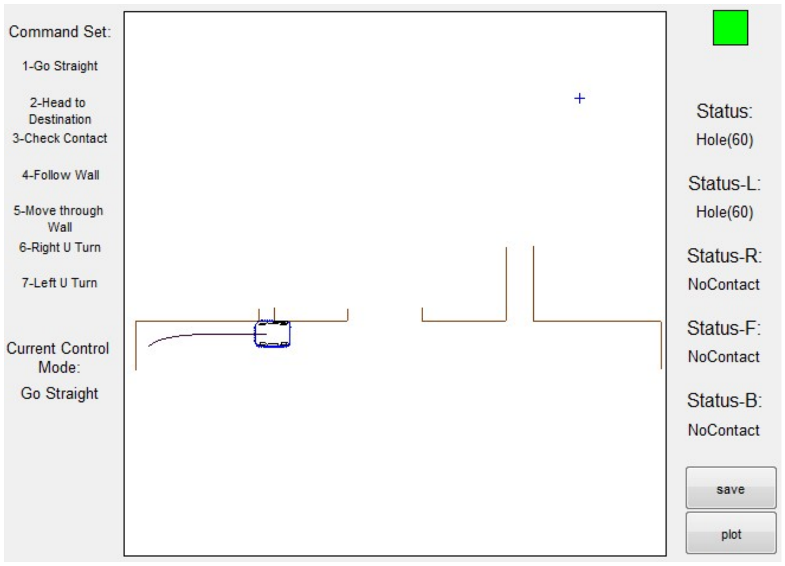
Status: LeavingWall
Status-L: LeavingWall
Status-R: NoContact
Status-F: NoContact
Status-B: NoContact

(b)

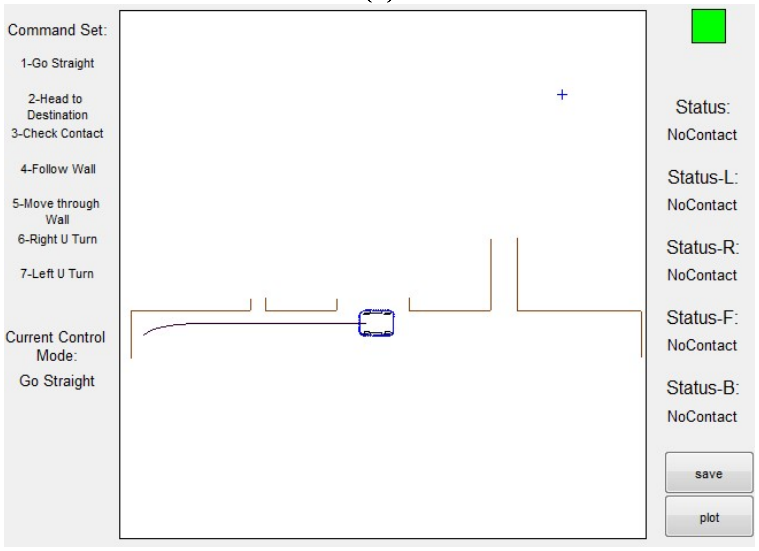
Figure 5. 5 Process of Following Wall ((a) in progress (b)stop)

Stage Four

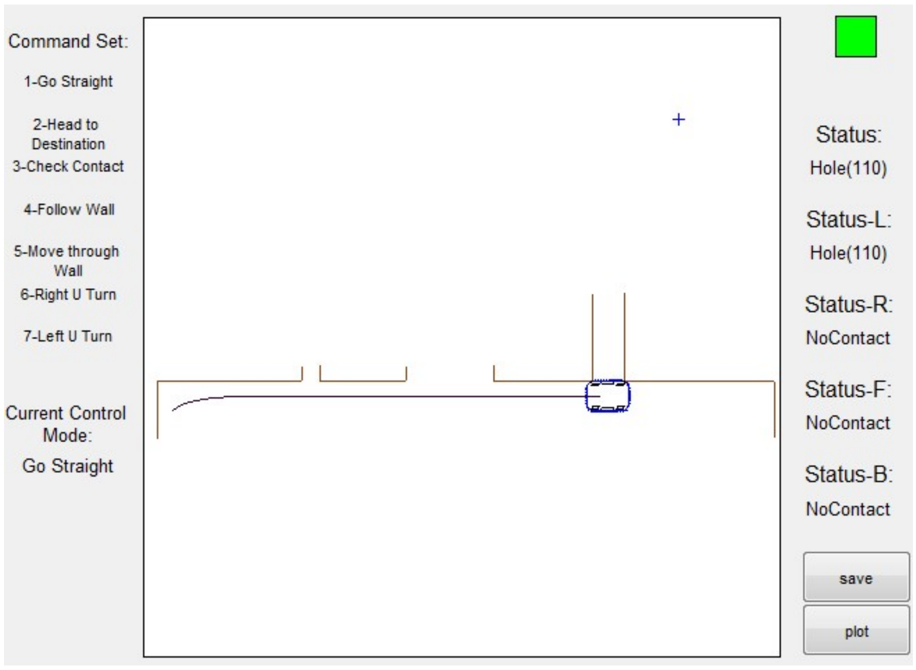
Since one of the objectives of the problem is to detect the whole wall, the car would move straight instead of heading to destination when the status is "Leaving Wall" or "NoContact". After a series of "Go Straight" and "Follow Wall" controls, the car can cross the hole, door and bumps, and finish the process of detecting the wall. (see Figure 5.7)



(a)



(b)



Current Control Mode:
Go Straight

Status:
Hole(110)

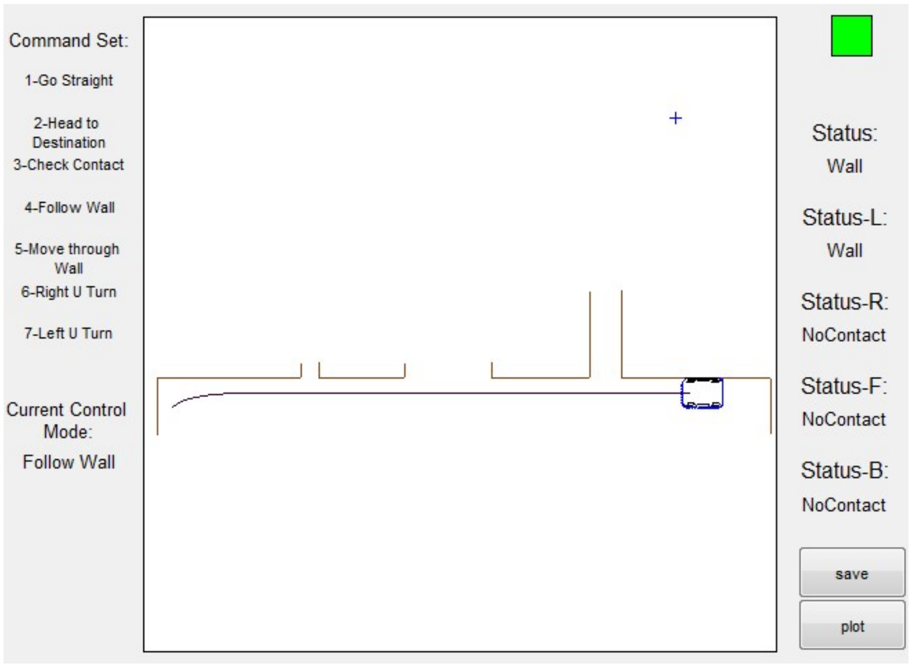
Status-L:
Hole(110)

Status-R:
NoContact

Status-F:
NoContact

Status-B:
NoContact

(c)



Current Control Mode:
Follow Wall

Status:
Wall

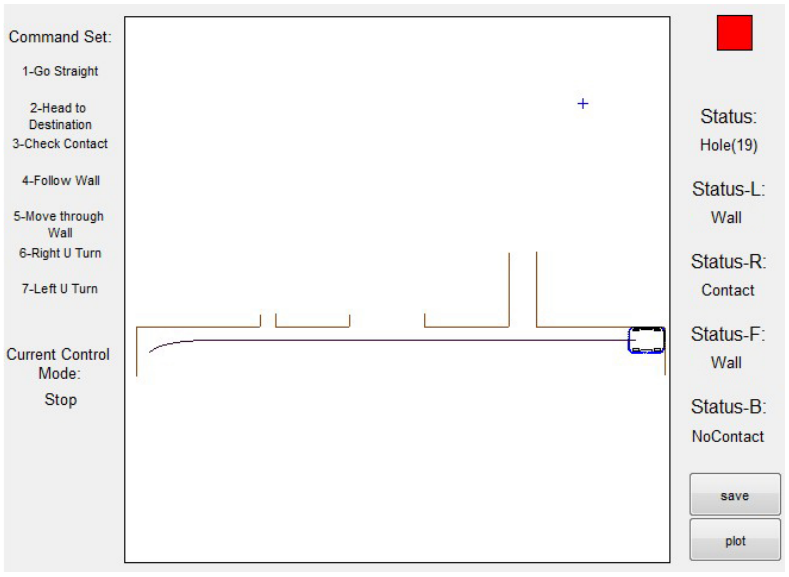
Status-L:
Wall

Status-R:
NoContact

Status-F:
NoContact

Status-B:
NoContact

(d)


Command Set:

1-Go Straight

2-Head to Destination

3-Check Contact

4-Follow Wall

5-Move through Wall

6-Right U Turn

7-Left U Turn

Current Control Mode:

Stop

Status:

Hole(19)

Status-L:

Wall

Status-R:

Contact

Status-F:

Wall

Status-B:

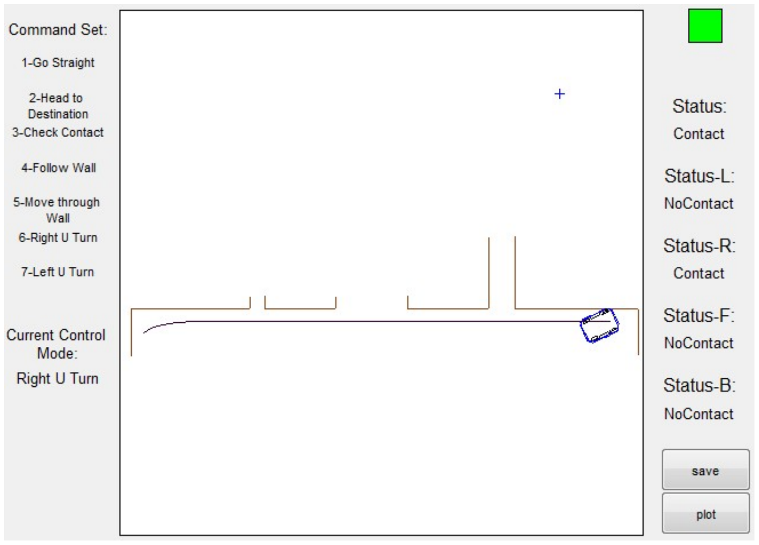
NoContact

(e)

Figure 5. 6 Process of Checking the Whole Wall ((a) detecting hole, (b) moving through the door, (c) detecting narrow path, (d) passing the bump, (e) stop)

Stage Five

Reviewing the records of the contact, it is found that there is a hole whose width is 110. Thus, the car need to turn around, move back until finding the hole, and check if the hole can be move into. (see Figure 5.8)


Command Set:

1-Go Straight

2-Head to Destination

3-Check Contact

4-Follow Wall

5-Move through Wall

6-Right U Turn

7-Left U Turn

Current Control Mode:

Right U Turn

Status:

Contact

Status-L:

NoContact

Status-R:

Contact

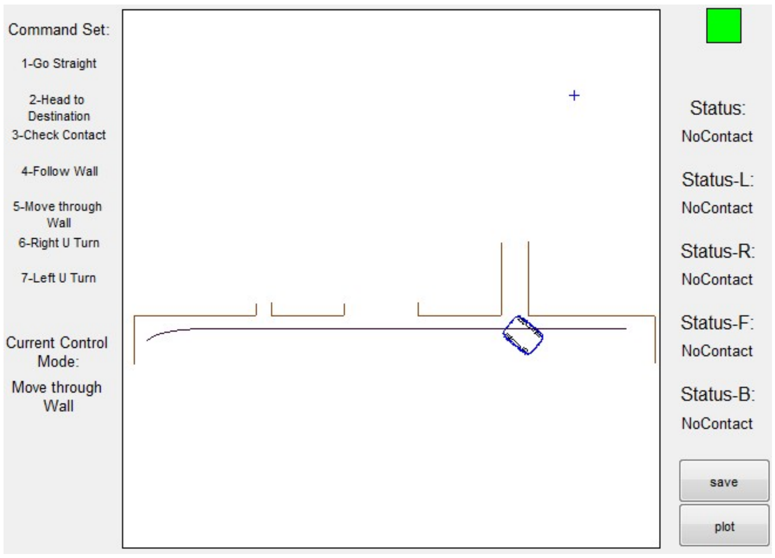
Status-F:

NoContact

Status-B:

NoContact

(a)



(b)

Figure 5. 7 Process of Detecting the Narrow Path ((a) in progress (b)stop)

Stage Six

After adjusting the head of the car, it would move through the narrow path and then head to the destination. (see Figure 5.9)

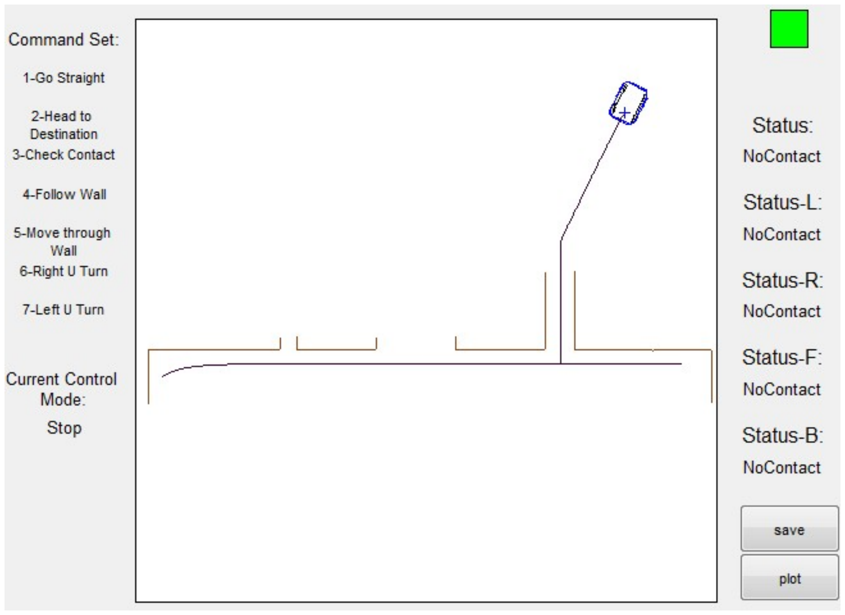


Figure 5. 8 Arriving at the Destination

This experiment proves that the simple navigation problem can be solved by some basic control strategies. The right status estimation is significant and helpful, since the control strategies need to be triggered and stop according to the status. It can also be proved that the feature classification model built in previous chapter is reliable because of the successful navigation result.

Chapter 6

Conclusion and Future Work

We have shown that the simulated robot can be driven to approach the destination in the simulator based on the tactile feedback, and that control behaviors can be developed for tactile navigation. The simulation is a good start and an effective tool to study tactile-based navigation. It can provide evidences for the theoretical feasibility of the robot system. However, there are numerous actual situations that the simulator cannot simulate. Thus, the algorithms will be tested in an actual robot in the near future.

Exploring around an object to identify it is another future extension of this work. This will be useful in number of scenarios. For example, in the Chapter 5, the robot moved through the narrow path. But, before the robot passed the path completely, it was hard to determine if the path is passable. Thus, how to detect the hole and how to prove that the hole is a dead path is some interesting works can be attempted to do.

The ultimate goal of the tactile based navigation is to handle the cluttered environment. This work provides some foundation to develop fully autonomous navigation by understanding the contact information.

Bibliography

1. Okuda, K., et al., *Obstacle arrangement detection using multichannel ultrasonic sonar for indoor mobile robots.* Artificial Life and Robotics, 2010. **15**(2): p. 229-233.
2. Abiyev, R., D. Ibrahim, and B. Erin, *Navigation of mobile robots in the presence of obstacles.* Advances in Engineering Software, 2010. **41**(10-11): p. 1179-1186.
3. Hähnel, D., D. Schulz, and W. Burgard, *Mobile robot mapping in populated environments.* Advanced Robotics, 2003. **17**(7): p. 579-597.
4. Jing Chen, Y.G., Xiaogang Ruan, and Hongjun Song, *A new dynamic self-organizing method for mobile robot environment mapping.* Journal of Intelligent Learning Systems and Applications, Nov. 2011: p. 249.
5. Morioka, H.Y., S.; Hasegawa, O., *Vision-based mobile robot's SLAM and navigation in crowded environments.* Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, 2011: p. 3998-4005.
6. Song, S.-Y.H.J.-B., *Monocular Vision-Based SLAM in Indoor Environment Using Corner, Lamp, and Door Features From Upward-Looking Camera.* Industrial Electronics, IEEE Transactions on, 2011. **58**: p. 4804-4812.
7. Fahimi, F., *Autonomous Robots - Modeling, Path Planning and Control.* Springer US, 2009.
8. Cowlagi, R.V.T., P., *Hierarchical Motion Planning With Dynamical Feasibility Guarantees for Mobile Robotic Vehicles.* Robotics, IEEE Transactions on, April 2012. **28**: p. 379-395.
9. Thrun, S., et al., *Stanley: The robot that won the DARPA Grand Challenge.* Journal of Field Robotics, 2006. **23**(9): p. 661-692.
10. Rover, C., *http://www.nasa.gov/images/content/750516main_pia16937-43_946-710.jpg.*
11. Yamada, Y., et al., *A tactile sensor system for universal joint sections of manipulators.* Robotics and Automation, IEEE Transactions on, 1993. **9**(4): p. 512-517.
12. JOURDAN, B.D., *Cool Cave Adventures.* http://onaquasirelatednote.wordpress.com/tag/caves/, 2013.

13. Someya, T., et al., *Conformable, flexible, large-area networks of pressure and thermal sensors with organic transistor active matrixes.* Proc Natl Acad Sci U S A, 2005. **102**(35): p. 12321-5.
14. Jonathan Engel, J.C.a.C.L., *Development of polyimide flexible tactile sensor skin.* JOURNAL OFMICROMECHANICS ANDMICROENGINEERING, 24 February 2003.
15. Heo, J.-S., J.-H. Chung, and J.-J. Lee, *Tactile sensor arrays using fiber Bragg grating sensors.* Sensors and Actuators A: Physical, 2006. **126**(2): p. 312-327.
16. Massaro, A., et al., *Real time optical pressure sensing for tactile detection using gold nanocomposite material.* Microelectronic Engineering, 2011. **88**(8): p. 2767-2770.
17. Hyung-Kew Lee, J.C., Sun-Il Chang, and Euisik Yoon, *Normal and Shear ForceMeasurement Using a Flexible Polymer Tactile SensorWith EmbeddedMultiple Capacitors.* JOURNAL OFMICROELECTROMECHANICAL SYSTEMS, AUGUST 2008. **17**(No.4).
18. Peng Peng, R.R., *Flexible Microtactile Sensor for Normal and Shear Elasticity Measurements.* IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, DECEMBER 2012.
19. Ando, S.S., H., *Ultrasonic emission tactile sensing.* Control Systems, IEEE, Feb 1995. **15**: p. 61-69.
20. Torres-Jara, E., *A soft touch: compliant tactile sensors for sensitive manipulation.* MIT CSAIL TR, March 1, 2006.
21. Jamone, L., et al. *James: A Humanoid Robot Acting over an Unstructured World*. in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. 2006.
22. Ravinder S. Dahiya, M., IEEE, Giorgio Metta,Maurizio Valle, Member, IEEE, and Giulio Sandini, *Tactile sensing—from humans to humanoids.* FEBRUARY 2010.
23. McMath, W.S., et al. *High sampling resolution tactile sensor for object recognition*. in *Instrumentation and Measurement Technology Conference, 1993. IMTC/93. Conference Record., IEEE*. 1993.
24. Rimon, Y.G.a.E., *C-Space Characterization of Contact Preserving Paths with Application to Tactile-Sensor Based Mobile Robot Navigation.* IEEE International Conference on Robotics and Automation, May 19-23, 2008.
25. David Jung, A.Z., *Whisker Based Mobile Robot Navigation.* Intelligent Robots and Systems '96, 1996.
26. Markus Fritzsche, N.E., and Erik Schulenburg, *Tactile Sensing: A Key Technology for Safe Physical Human Robot Interaction.* HRI'11, March, 2011.
27. Boll, S., A. Asif, and W. Heuten, *Feel your route: a tactile display for car navigation.* Pervasive Computing, IEEE, 2011. **10**(3): p. 35-42.
28. Sinyukov, D., T. Padir, and E. Torres-Jara. *Analysis of a contact problem for tactile sensors and a computationally effective simulation for grasping*. in

Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on. 2012.
29. Torres-Jara, E., *Sensitive Manipulation.* Ph.D Thesis, January 2007.
30. stepanov, V.J.L.a.A., *Path planning strategies for a point object moving amidst unknown obstacles of arbitrary shape.* Algorithmica, 1987. **403 - 430**.
31. Vidyassagar, A.S.a.M., *A new path planning algorithm for a point object moving amidst unknown obstacles in a plane.* IEEE Conference ON Roboteics and Automation, 1990.
32. Bishop, C.M., *Pattern Recognition and Machine Learning.* Springer, 2006.
33. Trevor Hastie, R.T., Jerome Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction.* Springer, 2010.
34. Reynolds, D.A. and R.C. Rose, *Robust text-independent speaker identification using Gaussian mixture speaker models.* Speech and Audio Processing, IEEE Transactions on, 1995. **3**(1): p. 72-83.
35. Russell S.J., N.P., *Artificial Intelligence: A Modern Approach.* Prentice Hall, 3rd edition, 2010.
36. Hathaway, R., *A constrained formulation of maximum-likelyhood estimation for normal mixture distribution.* Ann.Stat., 1985. **13**: p. 795-800.
37. McLachian, G., *Mixture Models.* New York: Marcel Dekker, 1988.