I, Srinivas Tennety,

hereby submit this work as part of the requirements for the degree of:

**Master of Science**

in:

**Mechanical Engineering**

It is entitled:

**Simulation of IGVC using Player/Stage**

This work and its defense approved by:

Chair: Dr. Ernest L. Hall

Dr. Richard L. Shell

Dr. Ronald L. Huston
Simulation of IGVC using Player/Stage

A project submitted to the
Division of Research and Advanced Studies
of University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
In the Department of Mechanical, Industrial and Nuclear Engineering
of the College of Engineering

2008

by

Srinivas Tennety
B.Tech in Mechanical Engineering, Sree Nidhi Institute of Science & Technology, 2006

Committee Chair: Dr. Ernest L. Hall
ABSTRACT

We are in an era where robots are being used in each and every phase of life. The robots are fast replacing humans in performing tasks that are classified as dangerous. For successful operation of these robots it is critical to find the correct path be it a war robot, or an unmanned vehicle in an urban setting. Intelligent Ground Vehicle Contest (IGVC) encourages the teams to do research in areas of path planning and obstacle avoidance and development and use of cutting edge technologies which eventually can be used in real world applications such as Military Mobility, Intelligent Transport Systems, Manufacturing etc.

A simulation program has been developed for IGVC using Player/Stage, an open source simulation software developed under GNU General Public License. The robot control program was developed in Java™ 2 Platform Standard Edition 5.0.

The simulation developed can be used as a test bed for testing various path planning and obstacle avoidance algorithms in general and for Intelligent Ground Vehicle Contest (IGVC) in particular. It covers Autonomous challenge and Navigation challenge parts of the contest.
Acknowledgements
Contents:

1. Introduction

2. Literature survey

3. Player/Stage Project

4. Stage Simulation and the Java Client program

5. Path Planning using SVM

6. Conclusion

References

Appendix
List of Figures

Figure 1 Bearcat Cub

Figure 2 Autonomous Challenge at IGVC-08

Figure 3 Obstacle Configurations on the Autonomous Course at IGVC

Figure 4 Navigation Challenge at IGVC

Figure 5 An image file to be loaded into stage

Figure 6 Simulated Stage environment

Figure 7 Navigation challenge at IGVC 08

Figure 8 Robot reaching the way point (3,6)

Figure 9 Avoiding obstacle

Figure 10 Moving around a fence to reach the way point (-4.5, 5)

Figure 11 Moving out of the fence to next way point.

Figure 12 Maximum margin hyperplane(linear)

Figure 13 A simulated robot detecting lines in a stage environment

Figure 14 Initial assigned labels Red -1 and Green +1

Figure 15 Detecting new points

Figure 16 The new points with assigned labels.

Figure 17 A new obstacle detected
Figure 18 The new obstacle classified in to one of the classes

Figure 19 The whole path classified and labeled accordingly.

Figure 20 Autonomous challenge at IGVC 08

Figure 21 An obstacle scanned along the path

Figure 22 SVM path (the blue dotted line)

Figure 23 Robot moving past a complex obstacle arrangement

Figure 24 The lines and obstacles detected and classified along with the SVM path

Figure 25 Detecting obstacles on both sides of the robot.

Figure 26 Obstacles detected on both sides

Figure 27 Lines and obstacles detected and SVM path (blue dots)

Figure 28 Robot moving past the curve with obstacles along the line.

Figure 29 Obstacles detected and classified along the curve.

Figure 30 A sample map with various obstacles arrangements

Figure 31 The whole map detected along with the obstacles
1. Introduction

Gone are the days when Robots were considered to be toys or used as interesting characters in science fiction novels. Today robots are being used in every phase of life. Robot applications span from domestic such as lawn mowing, vacuum cleaning to more superior such as deployment and defusing of bombast the war zones. It is estimated that there are more than 5400 robots being used at Iraq war by USA\(^1\). The successful use of robots would significantly decrease the fatalities of soldiers at war.

There is also ongoing research on developing fully autonomous vehicle for urban setting. According to W.H.O. 1.2 million people every year due to road accidents\(^2\). The autonomous car once fully developed would make driving safer and easier. The development of these kinds of vehicles is a challenge because of the various tasks they have to do such as driving in traffic, performing complex maneuvers such as merging, parking and following all the traffic rules.

Robotics team at UC is currently developing two autonomous vehicles. An autonomous jeep is being developed for the future DARPA Urban challenges and Ecocar contests and Bearcat Cub, a small autonomous vehicle for the Intelligent Ground Vehicle Contest (IGVC). It is not easy and possible to create the wide variety of environments for testing the above mentioned robots would encounter in the contest. The simulation developed not only serves the purpose of creating a wide variety of environments but also to test the various obstacle avoidance and path planning algorithms developed for the these robots.
1.1 IGVC

The simulation developed can be used for Autonomous challenge and Navigation challenge of Intelligent Ground Vehicle Contest (IGVC).

1.1.1 Autonomous Challenge

The objective of this part of the contest is to develop an autonomous unmanned vehicle capable of perceiving the course of the environment and avoid obstacles in its path.
The course for this challenge is laid out on grass over an area of approximately 60 to 120 yards long, by 40 to 60 yards wide and be 700 to 800 feet in length. The lane boundaries are either continuous or dashed lines in white or yellow color. The track width is approximately 10 ft wide and turning radius 5 ft. The course has randomly placed obstacles, sandpits and may also have artificial or natural inclines. It is designed to be progressively difficult.

Construction drums, cones, pedestals and barricades can be used as obstacles on the course. They can be in any color orange, white, blue, green, black etc. There will be six feet clearance between obstacles and the lines throughout the course even in the cases where obstacles are in the middle of the lines or adjacent to the line.
The obstacles are placed randomly and arranged in complex ways to make the course difficult to navigate. Switchbacks are arranged with barrels of different colors to test the ability of the vehicle in obstacle detection and avoidance.

1.2.2 Navigation Challenge

The objective of this part of the contest is to develop an autonomous vehicle which would travel from start point to various target positions avoiding any obstacles on its way and return to starting position\(^3\).
Figure 4 Navigation Challenge at IGVC

The GPS waypoints are given in degrees latitude and longitude just before the contest but no XY map is given. The course is set up on an area 40 by 90 meters approximately. The course is separated into two areas by a fence. The vehicle has to reach the other side of the fence through the opening. The obstacles in the course are placed in such a way that the vehicle has to avoid and go past them to reach the destination. These obstacle locations are randomly changed between runs³.
2. Literature survey

2.1 Player/Stage

Player/Stage can be regarded as a de facto standard in open source robotics community\(^4\). Player is a multithreaded TCP socket server that provides network access to sensors and actuators of a simulated or real robot. Its platform and language independent control of these devices allows the user to choose the best tool\(^5\).

The client programs developed using stage were found to work with minor or no modifications on real robots\(^6\). This is clearly an advantage of choosing Player/Stage. The devices used in Stage are simple and are of low fidelity when compared to original devices. This can be an advantage when developing controllers for real robots as it encourages the use of robust control techniques.

In the 2007 IGVC University of Detroit, Mercy achieved significant success using Player/Stage with Matlab client architecture. A simulation was also developed using Player/Stage to accommodate the testing and evaluation of their robot in various complex environments\(^7\).

2.2 SVM

SVMs are maximum margin classifiers that obtain a optimal separating hyperplane between the data sets. Jun\(^8\) proposed the used of SVM for path planning in a known terrain. Saurabh\(^9\) discussed the strategies that can be adopted to use SVM for unknown environments.

The present work discusses the use of Player/Stage in developing a Simulation for IGVC and also validates the strategies proposed by Saurabh for obstacle avoidance and path planning using SVM.
3. Player/Stage

Player/Stage is free software released under GNU General Public License. It was developed by an international team of robot researchers. It is perhaps claimed to be the most widely used robot control interface in the world. It is free to modify, use and distribute.

3.1 Player

Player is a robot device interface. It provides an interface for various sensors and other devices on the robot. The client program developed would communicate to the Player to get all the sensor data and in turn would give commands to the motors.

Player supports a wide variety of hardware like Sick lasers like LMS 200, Garmin GPS receiver Geko 201, SkyeTek RFID readers, Cameras manufactured by companies like Sony, Logitech and Canon. It also has preconfigured robots that are similar to commercial robots like Roomba vaccuming robot from iRobot, Robotic Mobility Platform (RMP), a custom-modified version of the Human Transport (HT) from Segway etc.

Player also supports wide variety of software. The client program can be written in any language that has support for TCP sockets. There are libraries for C, C++, Python are distributed with Player software. Several third party libraries are available for languages like Java, Matlab, Octave, Lisp, Ada etc.

The client programs for player need not be written in any preset structure. They can be multi threaded, read-think-act or interactive type. They are platform independent. They can be run from any machine that can communicate to the robot. Player can support any number of clients.
3.2 Stage

Stage is a 2-D simulation software for Player. It creates a virtual environment over a image file. The environment can be populated by robots, obstacles and many more things. The client program developed for Stage was found to be effective on real robots with no or minor modifications.

It is fairly simple to create an environment using Stage. The stage environment is defined in a “.world” file. The robot, devices and the obstacles are all defined in the world file. The desired background of the environment can be set using an image file (.png).

Figure 5 An image file to be loaded in to stage
Figure 6 Simulated Stage environment
4. Simulation and the Client program

The simulation developed for IGVC was in two parts: Autonomous challenge and Navigation challenge. Player/Stage was used to develop both the parts.

Features of the Simulation:

- Operating system: Ubuntu 7.04.
- Robot device server: Player 2.0.5.
- Simulator: Stage 2.0.4 for robots moving in 2D bitmapped environment.
- Robot Control program: Java™ SE 5.0 and JavaClient2.
- SVM for Path planning in the autonomous challenge.

4.1 Setting up the Stage:\(^1\)

4.1.1 World file

Everything on the stage environment has to be defined in this file.

- Size, scale and center of the GUI window
- Size of the world
- Resolution
- Image file to be uploaded and its size
- The robot model to be included
- Laser
- Map
4.1.2 Map model: The map model is defined in a file “map.inc”. In the map model we set the color of lines and also choose to enable some options like

- Visible boundary
- Movable environment
- Visible grid lines
- GUI outline

4.1.3 Robot model:

We basically set the properties of the robot like

- Size
- Weight
- visible nose to determine the direction of movement
- size of the transducers
- steering model

4.1.4 Laser model:

The properties of the laser are set in this file

- Minimum range
- maximum range
- scan angle
4.1.5 Configuration file

The configuration file is the place where the drivers, stage devices are loaded for the map, robot’s position2d interface and laser.

4.2 Java Client Program

The Client program for controlling the robot was developed in Java™ 2 SE 5.0. The JavaClient2 libraries and the standard libraries of JDK 5.0 were used. Java Swing was used to display the mapped coordinates and the SVM path.

4.2.1 Autonomous Challenge:

The Java implementation of LIBSVM¹² software was used for the path planning. The sequences of steps for the client program are:

1. Import all the classes required from JavaClient2, LIBSVM and the standard JDK 5.0 libraries.
2. Define all the variables and constants including the ones for PlayerClient object, Laser interface and Position2DInterface.
3. Construct the “PlayerClient” object and connect to the robot.
4. Request devices.
5. Read the data from the devices.
6. Store the coordinates of obstacles and lines and send them to SVM for training.
7. Predict the labels for the dummy points.
8. Determine the point at which the label changes.
9. Steer the robot with the direction obtained from SVM.

10. Display a map of the coordinates of the obstacles and the robot’s path.

The steps 5-10 are repeated till the destination is reached. The path planning is discussed in the chapter Path Planning by SVM. The actual code is given in the appendix A.

### 4.2.2 Navigation Challenge:

The Java Program implementation of Navigation Challenge follows the steps:

1. Import all the classes required from JavaClient2 and the standard JDK 5.0 libraries.

2. Define all the variables and constants including the ones for PlayerClient object, Laser interface and Position2Dinterface.

3. Obtain the waypoints and store them in a list.

4. Construct the “PlayerClient” object and connect to the robot.

5. Request devices.

6. Read the data from the devices.

7. Determine the heading required to reach the destination waypoint and steer in that direction.

8. Detect the obstacles in the path.

9. Avoid obstacles and go back to step 6.

10. After a way point is reached remove it from the list and set new destination and go back to step 6.

11. Quit after all the waypoints are reached.

The actual implementation code of way point challenge is given in the appendix B.

The simulation program was tested on maps that closely depict the actual contest. The client program was found to be stable in many complex situations.
Figure 7 Navigation challenge at IGVC 08
Figure 8 Robot reaching the way point (3, 6)
Figure 9 Avoiding obstacle
Figure 10 Moving around a fence to reach the way point (-4.5, 5)
Figure 11 Moving out of the fence to next way point.
5. Path Planning using SVM

Support vector machines (SVM) are extensively used in data classification and regression problems. They belong to a family of generalized linear classifiers. They are also known as maximum margin classifiers because they minimize the empirical classification error and simultaneously maximize the geometric margin\(^\text{13}\).

The data to be classified is divided into two input vectors and mapped into higher dimension space. The SVM constructs a hyperplane between the data sets which would maximize the margin between them. To achieve this two separate parallel hyperplanes are constructed and placed so that they are as close as possible to the data sets. The best separating hyperplane is the one which maximizes the margin between these two hyperplanes\(^\text{14}\).
Figure 12 Maximum margin hyperplane (linear)
5.1 Path Planning

LIBSVM software was used for the support vector classification. The coordinates of the lines and obstacles with their corresponding labels were used as data sets for SVM classification. The label -1 was assigned to the coordinates that are on the left side of the robot and the label +1 for the coordinates on the right.

The following steps were followed for using LIBSVM\textsuperscript{12}.

1. The data (coordinates with labels) was transformed into the format that is compatible with LIBSVM software.

2. The data scaled to a ratio 10:1.

3. RBF kernel was found to be effective and was used.

4. A model was generated using the training set.

5. The model generated was used to classify the test data.

6. The labels obtained for the test data determine the direction in which the robot has to steer in order to avoid obstacles.

5.1.1 Classification of data

The first few points are classified into two classes manually depending on whether they are on the left or right side of the robot. The labels are then assigned for the two classes as -1 for the points on the left and +1 for the points on the right.
Figure 13 A simulated robot detecting lines in a stage environment
After the initial phase of assigning labels for the few points, further classification is done depending on the distance between the new point and the previous points. The new point is assigned the same label as that of the class of the nearest point\textsuperscript{9}. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image_title}
\caption{Initial assigned labels Red -1 and Green +1}
\end{figure}
Figure 15 Detecting new points
Figure 16 The new points with assigned labels.
Similarly when a new obstacle such as a barrel is encountered it is assigned the label of the nearest point on the line.

Figure 17 A new obstacle detected
Figure 18 The new obstacle classified in to one of the classes
5.1.2 Training of SVM

As the robot moves along the path it detects new lines and obstacles. The coordinates of these new points are scaled to \(1/10^{th}\) of their value and stored in a data file along with their labels.

The training of SVM is done with the set parameters and it generates a model file. The model file is used to predict the labels of the test data.
5.1.3 Prediction:

To model file obtained from SVM is used to predict the labels for the test data. The test data contains a set of 11 dummy points aligned along a semi circle with centre as the position of the robot and radius 0.5m. The labels for these dummy points are predicted using the model file. The point at which the label changes from one class to the other is set as temporary goal for the robot. This process of training and prediction and continued till the robot reaches its final destination.

5.1.4 Testing

The Client program written for autonomous challenge was tested for many complex obstacle arrangements and maps that are identical to the IGVC 2008 course. It was found that SVM is effective in many situations.

Figure 20 Autonomous challenge at IGVC 08
Figure 21 An obstacle scanned along the path
Figure 22 SVM path (the blue dotted line)
Figure 23 Robot moving past a complex obstacle arrangement.
Figure 24 The lines and obstacles detected and classified along with the SVM path
Figure 25 Detecting obstacles on both sides of the robot.
Figure 26 Obstacles detected on both sides.
Figure 27 Lines and obstacles detected and SVM path (blue dots)
Figure 28 Robot moving past the curve with obstacles along the line.
Figure 29 Obstacles detected and classified along the curve.
Figure 30 A sample map with various obstacles arrangements.
Figure 31 The whole map detected along with the obstacles
6. Conclusions:

Player/Stage was found to be an effective tool for developing a simulation for IGVC. The support for various hardware devices, robots and algorithms makes it an ideal choice for developing simulations. The programs developed for simulations are claimed to work on actual robots with no significant changes. So, it can be used as a test bed to test various path planning and obstacle avoidance strategies and the successful client programs can be used in the original robot.

The SVM was found to be effective for path planning in many situations. Since it classifies both the lines and obstacles in to classes, it would eliminate the use of separate algorithms for path planning and obstacle avoidance if used effectively. Choosing the right size of training data and the parameters plays a critical role. In the real time situations the computation cost would play an important role.

The simulation developed for navigation challenge can be further improved by including VFH* algorithm for obstacle avoidance and search algorithms like A* and D* Light for better path planning.
References:

1. “Iraqi War Robots Recalled Following Alarming Behavior”
   http://news.bbc.co.uk/1/hi/world/americas/4199935.stm


10. Wiki, “Player Project”, URL: http://playerstage.sourceforge.net/wiki/Main_Page


14. Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, “A Practical Guide to Support Vector Classification”, Department of Computer Science National Taiwan University, Taipei, Taiwan
APPENDIX

APPENDIX A

To be appended.