SUPPORT VECTOR MACHINES BASED MOBILE ROBOT PATH PLANNING IN AN UNKNOWN ENVIRONMENT

Srinivas Tennety*, Saurabh Sarkar, Ernest L. Hall**, Manish Kumar****
Department of Computer Science*, Department of Mechanical Engineering**
University of Cincinnati
Cincinnati, Ohio 45220
tennetsp@email.uc.edu

ABSTRACT

In this paper the use of support vector machines (SVM) for path planning has been investigated through a Player/Stage simulation for various case studies. SVMs are maximum margin classifiers that obtain a non-linear class boundary between the data sets. In order to apply SVM to the path planning problem, the entire obstacle course is divided in to two classes of data sets and a separating class boundary is obtained using SVM. This non-linear class boundary line determines the heading of the robot for a collision-free path. Complex obstacles and maps have been created in the simulation environment of Player/Stage. The effectiveness of SVM for path planning on unknown tracks has been studied and the results have been presented. For the classification of newly detected data points in the unknown environment, the k-nearest neighbors algorithm has been studied and implemented.

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 and vacuum cleaning to defense such as deployment and disposal of bombs at war zones. For the successful operation of these robots, path planning is critical. Path planning is also important for autonomous vehicles that are developed to be deployed either on the road or in space. An autonomous car, if fully developed, would make driving safe and easy. Road accidents are still one of the major causes of death according to World Health Organization (W.H.O) and are predicted to increase in the coming years [1]. One of the challenges in developing an autonomous car is dealing with dynamic and uncertain environments which include other moving vehicles, and uncertain road and traffic conditions.

The mobile robot path planning problem in partially known and unknown environments has been studied for a long time. Borenstein and Koren [2] proposed the use of the Virtual Force Field approach which is an integration of Certainty Grids and Potential Fields. This approach accommodated for continuous movement of the robot in contrast with the edge detection method [3] for obstacle avoidance which required the stopping of the robot in front of the obstacle for more accurate measurement. The Virtual Force Field approach was initially developed for sonar sensors but was later extended to laser scanners [4]. For unknown environments, Ersson and Xiaoming [5] proposed an online path planning algorithm using network simplex method which is similar to D* approach proposed by Stentz [6]. The Reinforcement Learning approach was also used for obstacle avoidance in an unknown environment by Macek, Petrovic and Peric [7].

The use of Support Vector Machines (SVM) for path planning has been proposed by Miura [8] for the environments with known obstacles. Sarkar [9] discussed the use of SVM as a path planning algorithm that can help robots navigate through known and unknown environments. In the case of an unknown environment, the robot would make use of information obtained from the sensory devices.

The present work investigates the use of SVM for path planning in an unknown environment using a Player/Stage simulation. The approach of using k-nearest neighbors algorithm for the classification of new data points has been discussed and implemented in this work. The environments simulated are similar to the Intelligent Ground Vehicle Contest (IGVC), 2008. The paper is organized as follows: the introduction of SVM is followed by a brief description of the Player/Stage software and the simulation. Then, the use of SVM for path planning is discussed along with the use of k-nearest neighbors algorithm for classification of new data

points. It is followed by the results of the simulation for various case studies.

SUPPORT VECTOR MACHINES

Support vector machines (SVM) are maximum margin classifiers that obtain an optimal separating hyperplane between the data sets [8, 10]. SVM was primarily developed to solve the data classification problem and was successfully applied by the AT & T laboratory in their Optical Character Recognition (OCR) project. It was later extended to deal with machine learning and prediction problems. Over the past few years SVM has also been successfully applied in the research areas pertaining to multi-sensor information fusion, robot vision, human robot interaction, and mobile robot path planning and navigation [11, 12].

In order to apply the SVM, the input vectors are mapped into a higher dimensional space and then a maximum margin hyperplane is obtained between the data sets. For a training set of instance-label pairs (x_i, t_i) for i = 1,…., n, where $x_i \in R^m$ and $t \in \{1, -1\}^n$, the maximum separating hyperplane can be found from the solution of the optimization problem described in the following expression [11, 13].

$$\arg\min_{w,b,\xi}\left[\frac{1}{2}w^T w+\gamma\sum_{i=1}^{n}\xi_i\right] \tag{1}$$

Under the condition $t_i(w^T\phi(x_i)+h)\geq 1-\xi_i$, where w is the normal vector to the hyperplane, h is the bias, $\xi_i \geq 0$ is the slack variable which shows how much each sample moves to the other side. γ is the weight balancing the margin maximization and error reduction.

The function ϕ maps the input vectors into a higher dimensional space and then the linear separating hyperplane with maximum margin is generated by the SVM.

The $K(x_i, x_j)\equiv\phi(x_i)^T\phi(x_j)$ is the kernel function and the four basic kernel functions generally used are linear, polynomial, radial basis function and sigmoid.

Margin maximization for completely separable data

Let $(x_1, t_1), \ldots (x_n, t_n), x_i \in R^m, t_i \in \{-1, +1\}$ be the training sample [8-10] where x_i is the input vector, t_i is the label of the class to which x_i belongs and $w^T x - h = 0$ be the separating hyperplane. If the training data is linearly separable, then the parameters w and h satisfy

$$t_i(w^T x_i - h)\geq 1, (i=1,\ldots,n) \tag{2}$$

The two hyperplanes that separate the classes in such a way that there are no data points in between them are

$$(w^T x - h)=1 \tag{3}$$

The distance between the hyperplanes is $\frac{2}{\|w\|}$, where $\|.\|$ represents the Euclidean norm. In order to maximize the gap between the hyperplanes, we need to maximize $\frac{2}{\|w\|}$ or minimize the following function:

$$L(w)=\frac{\|w\|^2}{2} \tag{5}$$

under the constraints represented by Eq.(2).

The above problem can also be solved by introducing Lagrange multipliers α_i, i=1…n, one for each of the constraints represented in Eq. (2). The resulting Lagrangian is:

$$L_P \equiv \frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i t_i(x_i.w+h)+\sum_{i=1}^{n}\alpha_i \tag{6}$$

We require that the gradient of L_P with respect to w and h vanish under the constraints shown in Eq. (7) and Eq. (8).

$$w=\sum_{i=1}^{n}\alpha_i t_i x_i \tag{7}$$

$$\sum_{i=1}^{n}\alpha_i t_i = 0, \alpha_i \geq 0 (i=1,\ldots,n) \tag{8}$$

By substituting these conditions in Eq. (6) we get the dual formulation L_D which has to be maximized.

$$L_D(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j t_i t_j x_i^T x_j \tag{9}$$

The training data x_i with positive Lagrange multipliers α_i on any of the two hyperplanes $(w^T x - h)=1$, $(w^T x - h)=-1$ are known as Support vectors. These support vectors determine the parameters of the hyperplanes. Thus a discrimination function from these support vectors is given by

$$y = sign(w^T x - h)= sign\left(\sum_{i\in S}\alpha_i t_i x_i^T x - h\right) \tag{10}$$

where S is the set of all support vectors.

Soft margin for non-separable data

In the cases where the data is not linearly separable, some of the data points crossover the hyperplane to the other side. In those cases the new objective function to be minimized is obtained by the soft margin method.

$$L(w) = \frac{\|w\|^2}{2} + \gamma \sum_{i=1}^{n} \xi_i \qquad (11)$$

The solution to this problem can also be obtained by solving the dual problem similar to the linear case. α_i is non zero in the cases where it is on one of the hyperplanes and for the support vectors inside the hyperplanes where $0 < \alpha_i < \gamma$ holds and for support vectors inside the hyperplanes, $\alpha_i = \gamma$.

Non-linear SVM using kernel functions

Using appropriate kernel functions, non-linear separating hyperplane can be obtained in higher dimensions. This hyperplane in original space forms a non-linear separating surface according to Burges [8-10].

Let π map the data from original space to higher dimension. This mapping is provided by certain kernel functions K, that is:

$$\pi(x_1)^T \pi(x_2) = K(x_1, x_2) \qquad (12)$$

Thus the objective function in higher dimension becomes:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j K(x_i, x_j) \qquad (13)$$

The discrimination function which helps to classify new points is

$$y = sign\left(\sum_{i \in S} \alpha_i t_i K(x_i, x) - h \right) \qquad (14)$$

PLAYER/STAGE SIMULATION

Player/Stage is one of the popular software platforms in the open source robotics community. It is a free software released under GNU General Public License. It was developed by an international team of robot researchers.

Player

Player is a multi-threaded TCP/IP socket server that provides network access to sensors and actuators of a simulated or real robot [14]. It provides an interface for various sensors and other devices on the robot. The control program developed would communicate to the Player to receive the sensor data and send the command signals to the devices. Player supports a wide variety of hardware and software. It is platform and language independent. The client programs for Player need not be written in any preset structure. They can be multi threaded, read-think-act or interactive type.

Stage

Stage is the 2-D simulation software for Player. It creates a virtual environment which can be populated by robots, obstacles and other objects. Furthermore, it can accept an image file as shown in Fig. 1 on the left and generate a simulated robot environment as shown in the right with the solid line representing path boundaries or obstacles. The client programs developed for Stage are claimed to be effective on the real robots with minor modifications [15].

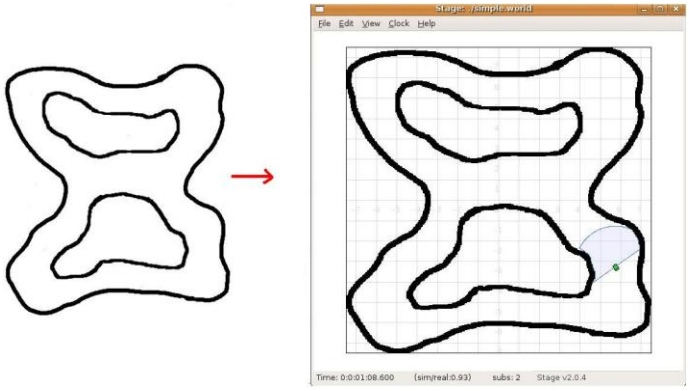

FIGURE 1. THE SIMULATED STAGE ENVIRONMENT FROM AN IMAGE FILE

Software

The robot control program for the simulation has been developed using Java™ 2 SE 5.0. The LIBSVM software has been used for implementing the SVM in the simulation. LIBSVM is a library for support vector machines whose goal is to help users implement and use SVM [16].

SVM BASED PATH PLANNING

In this section, the SVM based path planning approach is discussed in detail and simulation results are presented. The task the robot has to accomplish is to stay within the solid line boundaries and avoid any obstacles placed on its path. The simulated robot has no prior information about the environment. It learns about the environment online and uses SVM to find a collision-free path.

In order to apply the SVM for path planning, the whole environment that the robot detects is divided into two classes of data sets and the SVM is used to determine the maximum margin hyperplane between the data sets belonging to the two classes. This hyperplane represents a collision-free path, assuming that there is one. The conventions followed for the simulation results are shown in Fig. 2.

The two classes with labels -1 and +1 are represented in red (+) and green (x) points respectively and the SVM path is represented by the blue (*) points.

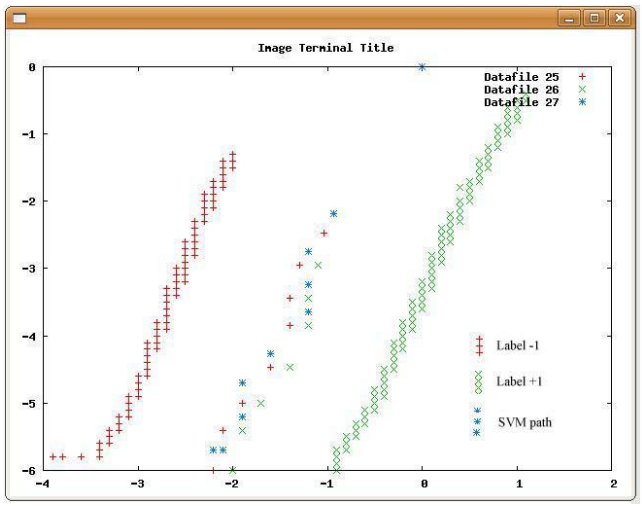

FIGURE 2. IMAGE SHOWING THE CLASSES WITH LABELS AND SVM PATH

Classification of data points

The first few points are classified into two classes 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 as shown in Fig. 3.

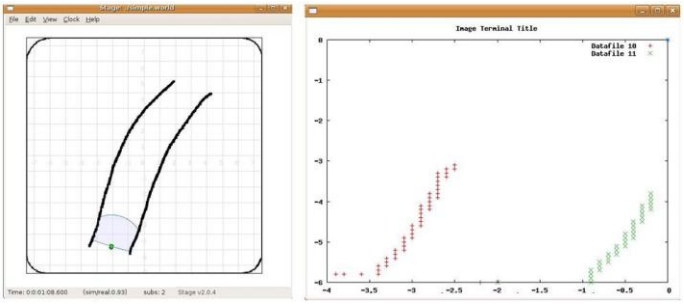

FIGURE 3. THE INITIAL LABELS FOR THE DATA POINTS ASSIGNED MANUALLY

After the initial phase of assigning labels for the few points, further classification is done using the k-nearest neighbors algorithm.

Using k-nearest neighbors

The k-nearest neighbors algorithm is a machine learning algorithm in which a data point or object is classified depending on its 'k' nearest neighbors. The data point is assigned a label that is most common among its 'k' nearest neighbors. In the current approach of using SVM, 'k' is chosen to be 1. So, the data point to be classified is assigned the label of its nearest neighbor. The labels assigned using 1-nearest neighbor strategy are shown in Fig. 4.

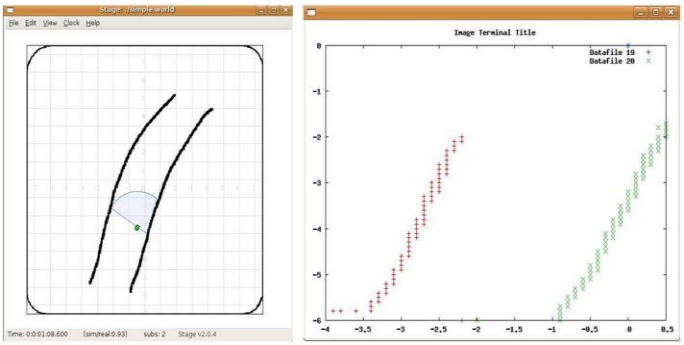

FIGURE 4. THE NEW DATA POINTS CLASSIFIED USING 1-NEAREST NEIGHBOR STRATEGY

The obstacles that the robot might encounter in its path are also classified in to one of the classes depending on the proximity with other points as shown in Fig. 5 and Fig. 6.

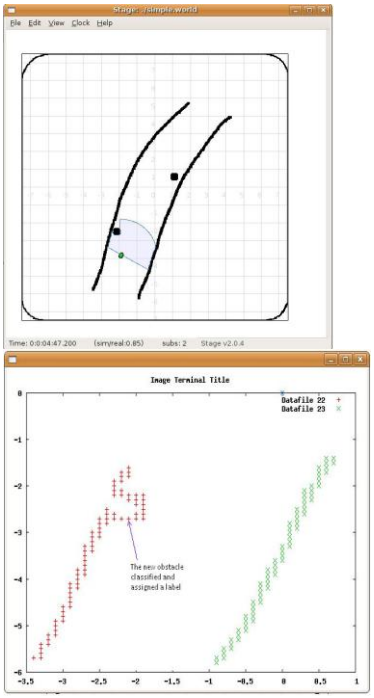

FIGURE 5. THE OBSTACLE CLASSIFIED IN TO THE CLASS OF ITS NEAREST NEIGHBOR

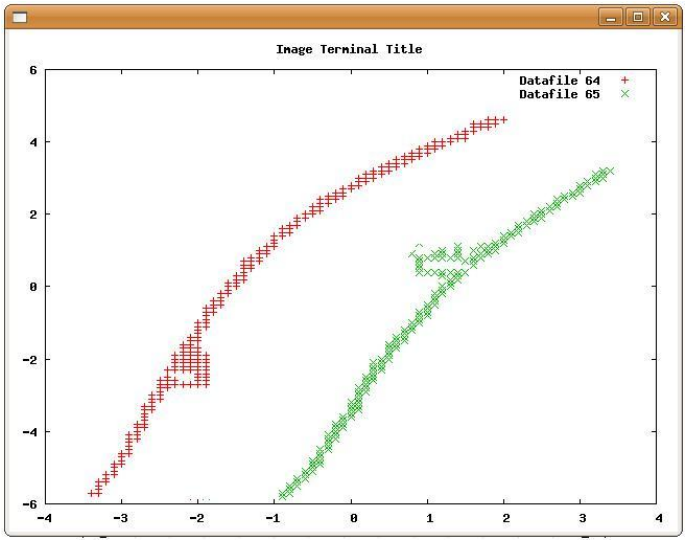

FIGURE 6. THE WHOLE PATH CLASSIFIED IN TO TWO DATA CLASSES

Predicting the collision-free path

In order to use the LIBSVM [16] tool for implementing SVM to obtain the maximum margin hyperplane, the data points were transformed into the format compatible with the tool. The data was scaled on a ratio of 10:1. The kernel function used was Radial Basis Function (RBF) [9].

$$\text{RBF Kernel: } K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2\right), \lambda > 0 \quad (15)$$

Here λ is a kernel parameter. In order to keep the SVM path to always pass through the current position of the robot, two imaginary points from both the classes are added to the data set before predicting the path.

Simulation results for various obstacle configurations

The simulation was tested with various obstacle arrangements and maps. The results have been presented here. In all these case studies, the robot has no prior knowledge of the environment and it learns everything online while sensing the environment. An appropriate range was chosen for the simulated sensor in the Player/Stage environment. If the solid line boundary falls within the sensing range, data points are added and processed for classification. A simple path with a few obstacles is shown in Fig. 7.

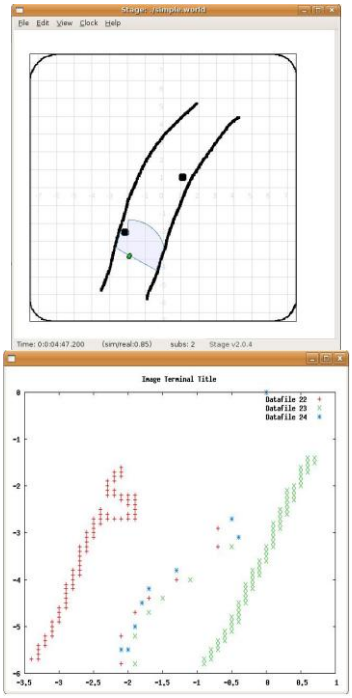

FIGURE 7. THE SVM PATH FOR A SIMPLE OBSTACLE PATTERN

Figure 8 shows a complex map employed to test the effectiveness of the approach. Figure 9 shows how the lines and the obstacles were classified and also the path obtained using SVM.

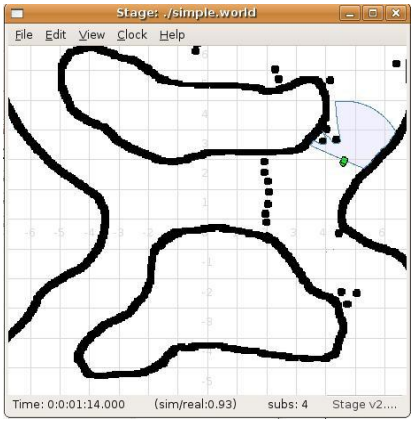

FIGURE 8. A COMPLEX MAP

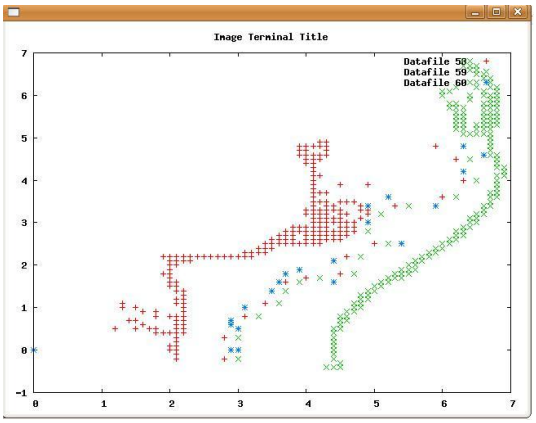

FIGURE 9. ROBOT NAVIGATING USING SVM

In Fig. 10, the robot detects obstacles on both sides and successfully classifies them in to one of the classes. The updated collision-free path obtained by SVM can be seen in Fig. 11.

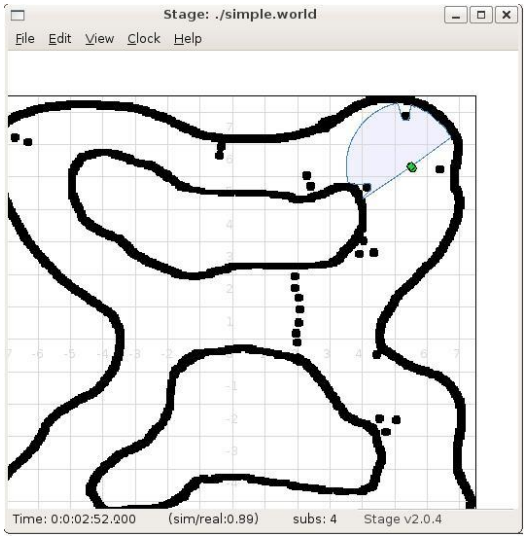

FIGURE 10. DETECTING OBSTACLES ON BOTH SIDES OF THE PATH

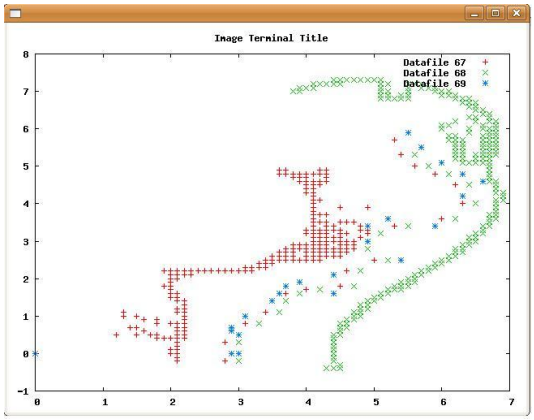

FIGURE 11. THE COLLSION FREE SVM PATH

The map was designed in such a way that the robot would encounter lanes with varying width and different obstacle patterns and the effectiveness of SVM in accommodating to those changes has been investigated. The updated SVM path when the lane width decreases is shown in Fig. 12.

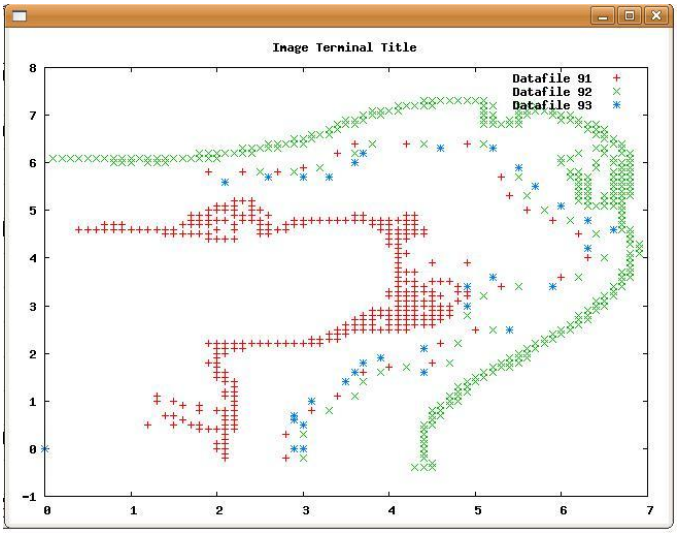

FIGURE 12. THE ROBOT GOING FURTHER SUCCESSFULLY

The Fig. 13 shows the robot taking a turn along the curve and avoiding the obstacles detected in its path. The SVM path can be seen as blue dots in Fig 14. In Fig. 14, we can also see the whole obstacle course divided in to two classes and the collision-free path obtained using SVM.

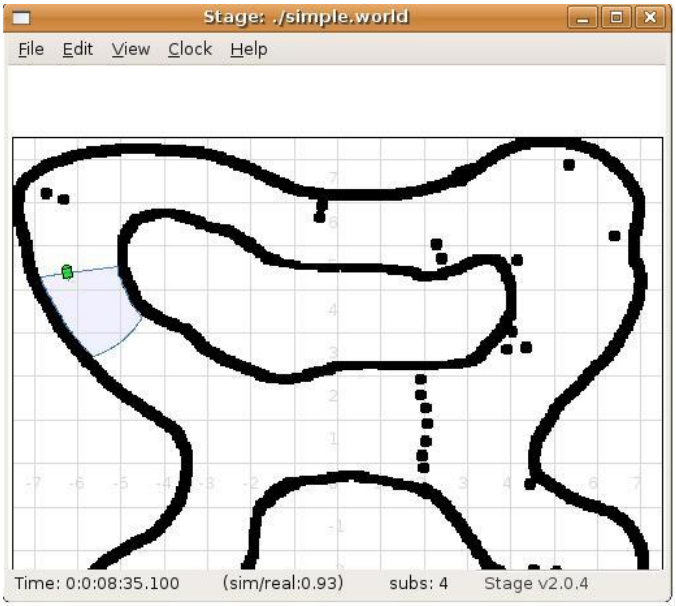

FIGURE 13. ROBOT MOVING PAST THE CURVE AND AVOIDING OBSTACLES ALONG THE LINE

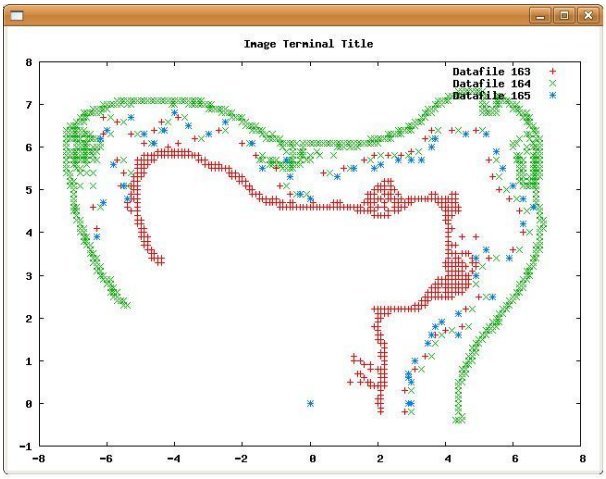

FIGURE 14. OBSTACLES DETECTED, CLASSIFIED AND SVM PATH DETERMINED

CONCLUSIONS

In this work the Support Vector Machines theory has been used in conjunction with k-nearest neighbors algorithm for mobile robot path planning in an unknown environment. The simulation results using the Player/Stage software have been presented for various complex maps and obstacle arrangements. It has been found that the proposed approach is effective in most situations. The path obtained from SVM is smooth, free of obstacles and facilitates continuous movement of the robot.

In situations that demand both lane compliance and obstacle avoidance, the use of SVM would simplify the problem. It would eliminate the scope of any conflicts that may arise due to the usage of two different algorithms for the tasks. The computational complexity for classification of new data points was found to increase with the increase in the size of the data set containing the previously classified points. An appropriate limit can be set to data size to make the computations faster.

ACKNOWLEDGMENTS

The Player/Stage software for creating the simulation and the JAVA™ source for the robot control program have been obtained from Sourceforge (URL:http://playerstage.sourceforge.net, http://java-player.sourceforge.net) LIBSVM software used to implement SVM has been obtained from Chih-Chung Chang and Chih-Jen Lin (URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm).

REFERENCES

[1] W.H.O., 2004. The global burden of disease. Part 2, pp. 8-26.

[2] J. Borenstein and Y. Koren, 1989. “Real-time obstacle avoidance for fast mobile robots”. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), pp. 1179-1187.

[3] J. Borenstein and Y. Koren, 1988. “Obstacle avoidance with ultrasonic sensors”. *IEEE Journal of Robotics and Automation*, Vol. RA-4, No. 2, pp. 213-218.

[4] S. Sarkar, 2007. ”Path planning and obstacle avoidance in mobile robots”. *Master’s Thesis*, University of Cincinnati.

[5] T. Ersson and H. Xiaoming, 2001. “Path planning and navigation of mobile robots in unknown environments” Intelligent Robots and Systems, Proceedings, IEEE/RSJ International Conference, Volume: 2, pp. 858-864.

[6] A. Stentz, 1994. “Optimal and efficient path planning for partially-known environments”. Proceedings of IEEE International Conference on Robotics and Automation, vol. 4, pp. 3310-3317.

[7] K. Macek, I. Petrovic and N. Peric, 2002. “A reinforcement learning approach to obstacle avoidance of mobile robots”. 7th International Workshop on Advanced Motion Control, pp. 462- 466.

[8] J. Miura, 2006. “Support Vector Path Planning”. International Conference on Intelligent Robots and Systems, IEEE/RSJ, Beijing, China.

[9] S. Sarkar, E. Hall, M. Kumar, 2008. “Mobile robot path planning using support vector machines”. ASME Dynamic Systems and Control Conference, October 20-22, Ann Arbor, Michigan, USA.

[10] C. J. C. Burges, 1998. “A tutorial on support vector machines for pattern recognition”. Data Mining and Knowledge Discovery 2(2), 121-167.

[11] J. Lei, Q. Song, J. Ma, L. Qiu and Y. Ge, 2005. “Application of SVM in intelligent robot information acquisition and processing: A survey”. International Conference on Information Acquisition, IEEE, June 27 – July 3, Hong Kong and Macau, China.

[12] Vapnik V N., 1998. Statistical learning theory. New York: John Wiley & Sons.

[13] Vapnik V N., 1995. The nature of statistical learning theory. 1995. Springer.

[14] B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhatme and M. J Mataric. 2001. “Most valuable player: A robot device server for distributed control”. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), October 29 - November 3, pp. 1226-1231, Wailea, Hawaii.

[15] B. Gerkey, R. T. Vaughan and A. Howard. 2003. “The Player/Stage project: Tools for multi-robot and distributed sensor systems”. In Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003), June, pages 317-323, Coimbra, Portugal.

[16] C. Chang and C. Lin, 2001. “LIBSVM: a library for support vector machines”. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.