CHAPTER 4
FUZZY SUPPORT VECTOR MACHINE BASED
MULTI-AGENT OPTIMAL PATH PLANNING
APPROACH TO ROBOTICS ENVIRONMENT

4.1 INTRODUCTION

The pervasive presence of uncertainty in sensing and learning makes the choice of a suitable tool of reasoning and decision making that can deal with incomplete information, vital to ensure a robust control system. This problem can be overcome by the proposed navigation method using Fuzzy Support Vector Machine (FSVM). It proposes a fuzzy logic based SVM approach to secure a collision free path avoiding multiple dynamic obstacles for unknown environment. The navigator consists of an FSVM -Based Collision Avoidance. The decisions are taken at each step for the mobile robot to attain the goal position without collision. Fuzzy-SVM rule bases are built, which requires simple evaluation data rather than thousands of input-output training data.

By incorporating multi-agent optimal path planning (Emmanuel et al 2006) and Support Vector Machine learning, special efficiency for robotic navigation can be achieved. One of the important aspects that are still deemed important to consider in mobile robots is obstacle avoidance. A pertinent problem in autonomous navigation is the need to cope with a large amount of uncertainty that is inherent of natural environments. Fuzzy logic with Support Vector Machine is an adequate method to obtain certain and
finite data in an optimal manner. However, this approach proves that it is enough for areas of highest danger coefficients to cause the robot to change direction in multi robotic environment, which consequently reduces the number of fuzzy rules that control the robot motion (Evgeniou et al 2000).

SVM learns the decision surface from two distinct classes of the input points (Shigeo Abe and Takuya 2001) and (Shigeo Abe and Takuya 2002). In many applications, each input point may not be fully assigned to one of these two classes. Here, we apply a fuzzy membership to each input point and reformulate the SVMs such that different input points can make different contributions to the learning of decision surface.

4.2 FSVM SYSTEM STRUCTURE

Figure 4.1 shows the integration of a microcontroller (89C52) with the features of the robot. This architecture uses four sensors out of which the two proximity sensors are used to detect the obstacles, one to detect the color and the last one to detect the pit in the navigation path (Miller 1998). Two stepper motors are used to navigate the robot which will be simulated by the stepper motor driver agent.

![Figure 4.1 Integration of microcontroller for FSVM](image-url)
4.2.1 Support Vector Machines

SVM is a Machine Learning Technique developed on statistical learning theory. For machine learning tasks involving pattern classification, multi sensors information fusion and non-linear system control etc, SVMs have become an increasingly popular tool. SVMs are a set of related supervised learning methods used for classification and regression. Viewing input data as two sets of vectors in an n-dimensional space, an SVM will construct a separating hyper plane in that space, which maximizes the margin between the two data sets. To calculate the margin, two parallel hyper planes are constructed, one on each side of the separating hyper plane, which is "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the neighboring data points of both classes, since in general the larger the margin the better the generalization error of the classifier. The goal is to accurately classify all the data. For mathematical calculations we have,

\[
\text{If } Y_i = +1; \quad wx_i + b \geq 1 \quad (4.1)
\]
\[
\text{If } Y_i = -1; \quad wx_i + b \leq 1 \quad (4.2)
\]
\[
\text{For all } i; \quad y_i (w_i + b) \geq 1 \quad (4.3)
\]

In this equation x is a vector point and \( w \) is weight and is also a vector. So to separate the data \((wx_i + b)\) should always be greater than zero. Among all possible hyper planes, SVM selects the one where the distance of hyper plane is as large as possible. If the training data is good, every test vector is located in radius \( r \) from training vector. Now if the chosen hyper plane is located at the farthest possible from the data this desired hyper plane which maximizes the margin also bisects the lines between closest points on convex hull of the two datasets. The above three equations are represented as given in Figure 4.2.
Figure 4.2 Representation of hyper planes

Distance of closest point on hyperplane to origin can be found by maximizing the $x$ as $x$ is on the hyper plane. Similarly for the other side points we have a similar scenario. Thus solving and subtracting the two distances we get the summed distance from the separating hyper plane to nearest points.

$$\text{Maximum Margin} = M = \frac{2}{||w||}. \quad (4.4)$$

Now maximizing the margin is same as minimum. Now we have a quadratic optimization problem and we need to solve for $w$ and $b$. To solve this, we need to optimize the quadratic function with linear constraints. The solution involves constructing a dual problem and where a Langlier’s multiplier $\alpha_i$ is associated. We need to find $w$ and $b$ such that $\Phi (w) = \frac{1}{2} ||w'|| ||w||$ is minimized:

for all $\{(x_i, y_i)\}: \ y_i (w \cdot x_i + b) \geq 1 \quad (4.5)$

Now solving: we get that

$$w = \Sigma \alpha_i \cdot x_i; \ b = y_k - w \cdot x_k \ for \ any \ x_k \ such \ that \ \alpha_k \neq 0 \quad (4.6)$$

Now the classifying function will have the following form:

$$f(x) = \Sigma \alpha_i y_i x_i \cdot x + b \quad (4.7)$$
4.2.2 SVM Representation

In this we present the QP formulation for SVM classification. This is a simple representation only.

**SVM classification**

\[
\min_{f, \xi} \|f\|^2_K + C \sum_{i=1}^{l} \xi_i
\]  

\[
y_i f(x_i) \geq 1 - \xi_i, \text{ for all } i \quad \xi_i \geq 0
\]  

**SVM classification, Dual formulation:**

\[
\min_{\alpha_i} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad 0 \leq \alpha_i \leq C, \text{ for all } i;
\]  

\[
\sum_{i=1}^{l} \alpha_i y_i = 0
\]

Variables \(\xi_i\) are called slack variables and they measure the error made at point \((x_i, y_i)\). Training SVM becomes quite challenging when the number of training points is large (Figure 4.3). A number of methods for fast SVM training have been proposed.
4.2.3  Soft Margin Classifier

In real world problem it is not likely to get an exactly separate line dividing the data within the space. And we might have a curved decision boundary. We might have a hyper plane which might exactly separate the data but this may not be desirable if the data has noise in it. It is better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers. This is handled in a different way; here we hear the term slack variables being introduced. Now we have,

\[ y_i (w'x + b) \geq 1 - S_k. \]  \hspace{1cm} (4.11)

This allows a point to be a small distance $S_k$ on the wrong side of the hyper plane without violating the constraint. Now we might end up having huge slack variables which allow any line to separate the data, thus in such scenarios we have the Lagrangian variable introduced which penalizes the large slacks.

\[ \text{Min } L = \frac{1}{2} w'w - \sum a_i ( y_k (w'x_k + b) + s_k -1) + \alpha \sum s_k \]  \hspace{1cm} (4.12)

where reducing $\alpha$ allows more data to lie on the wrong side of hyper plane and would be treated as outliers which give smoother decision boundary.

4.2.3.1  Kernel Trick

Kernel: If data is linear, a separating hyper plane may be used to divide the data. However it is often the case that the data is far from linear and the datasets are inseparable. To allow this, kernels are used to non-linearly map the input data to a high-dimensional space. The new mapping is then linearly separable. A very simple illustration of this is shown below in Figure 4.4
This mapping is defined by the Kernel:

\[ K(x, y) = \Phi(x) \cdot \Phi(y) \]  \hspace{1cm} (4.13)

**Feature Space:** Transforming the data into feature space makes it possible to define a similarity measure on the basis of the dot product. If the feature space is chosen suitably, pattern recognition can be easy as given in equation (4.14) and Figure 4.5.

\[ \langle x_1 \cdot x_2 \rangle \leftarrow K(x_1, x_2) = \langle \Phi(x_1) \cdot \Phi(x_2) \rangle \]  \hspace{1cm} (4.14)
Now getting back to the kernel trick, we see that when \( w, b \) is obtained the problem is solved for a simple linear scenario in which data is separated by a hyper plane. The Kernel trick allows SVM’s to form nonlinear boundaries. Steps involved in kernel trick are given below.

(a) The algorithm is expressed using only the inner products of data sets. This is also called as dual problem.

(b) Original data are passed through non linear maps to form new data with respect to new dimensions by adding a pair wise product of some of the original data dimension to each data vector.

(c) Rather than an inner product on these new, larger vectors, and stored in tables and later doing a table lookup, we can represent a dot product of the data after doing non linear mapping on them. This function is the kernel function. More on kernel functions is given below.

**Kernel Trick: Dual Problem**

First we convert the problem with optimization to the dual form in which we try to eliminate \( w \), and a Lagrangian now is only a function of \( \alpha_i \). To solve the problem we should maximize the \( L_D \) with respect to \( \alpha_i \). The dual form simplifies the optimization and we see that the major achievement is the dot product obtained.

**Kernel Functions**

The idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner product does not need to be evaluated in the
feature space. We want the function to perform mapping of the attributes of
the input space to the feature space. The kernel function plays a critical role in
SVM and its performance. It is based upon reproducing Kernel Hilbert Spaces.

\[ K(x, x') = \langle \phi(x), \phi(x') \rangle \]  \hspace{1cm} (4.15)

If \( K \) is a symmetric positive definite function, which satisfies
Mercer’s Conditions,

\[ K(x, x') = \sum_m a_m \phi_m(x)(\phi_m)(x'), \quad a_m \geq 0, \]  \hspace{1cm} (4.16)

\[ \iint K(x, x') g(x)g(x')dxdx' > 0, \quad g \in L_2 \]  \hspace{1cm} (4.17)

Then the kernel represents a legitimate inner product in feature
space. The training set is not linearly separable in an input space. The training
set is linearly separable in the feature space. This is called the “Kernel trick”.

4.3 MULTI CLASS SUPPORT VECTOR MACHINE

For the conventional SVM an \( n \) class problem is converted into \( n \)
two-class problem and for the \( i^{th} \) two-class problem, class \( i \) is separated from
remaining class with decision function that classifies class \( i \) and remaining
classes be

\[ D_i(x) = w_i^T x + b_i \]  \hspace{1cm} (4.18)

SVM, if for the input vector \( x \)

\( D_i(x) > 0 \) is classified for \( i \), \( x \) is classified in to class \( i \).

Let the decision function for class \( i \) against class, with the
maximum margin, be
\[ D_{ij}(x) = w_{ij}^T x + b_{ij}, \quad (4.19) \]

where \( D_{ij}(x) = D_{ji}(x) \) for the input vector \( x \), we calculated \( n \)

\[ D_i(x) = \sum_{i=1}^{n} \text{sign}(D_{ij}(x)) \quad (4.20) \]

and classify \( x \) into the class

\[ \arg \max_{i=1,...,n} D_i(x) \quad (4.21) \]

4.4 FUZZY SUPPORT VECTOR MACHINE (FSVM)

SVM learns the decision surface from two distinct classes of the input points. In many applications, each input point may not be fully assigned to one of these two classes. In this thesis, we apply a fuzzy membership to each input point and reformulate the SVMs such that different input points can make different contribution to the learning of decision surface.

Class \( i \) is defined as a one dimensional membership functions \( m_{ii}(x) \) on the direction orthogonal to the optimal separating hyper planes \( D_j(x) = 0 \) as given below

1. For \( i = j \)
   \[ m_{ii}(x) = 1 \text{ for } D_i(x) > 1, \quad (4.22) \]
   \( D_i(x) \) otherwise

2. For \( i \neq j \)
   \[ m_{ij}(x) = 1 \text{ for } D_j(x) <-1, \quad (4.23) \]
   \( -D_i(x) \) otherwise
We define the class $i$ membership function of $x$ using the minimum operator for

$$m_i(x) = \min_{j=1,\ldots,n} m_{ij}(x) \quad (4.24)$$

Now the datum $x$ is classified into the class $\arg\max_{i=1,\ldots,n} m_i(x)$ if $x$ satisfies

$$D_k(x) = \begin{cases} > 0 & \text{for } k = i, \\ \leq 0 & \text{for } k \neq i, k = 1,\ldots,n \end{cases} \quad (4.25)$$

and $m_k(x)$ is given by

1. $k \in i_1,\ldots,i_l$
   $$m_k(x) = \min_{j=i_1,\ldots,i_l, j \neq k} -D_j(x). \quad (4.26)$$

2. $k \neq j$ ($j = i_1,\ldots,i_l$)
   $$m_k(x) = \min_{j=i_1,\ldots,i_l} -D_j(x). \quad (4.27)$$

Thus the maximum degree of membership is achieved among $m_k(x)$, $k = i_1,\ldots,i_l$ (Figure 4.6).

Figure 4.6 Contour lines of the class $i$ membership function
4.5 IMPLEMENTATION DETAILS

Architecture: The robotic system architecture used in this thesis consists of two layers. The Hardware Layer is a collection of modules communicating with the robot's hardware devices such as infrared sensors and motors. One module can be shared by two or more agents, which reduces redundancy in coding. The development tools used is C language and Keil software, which is a part of Microsoft turbo C and 89C52 IC. (Same Architecture as in Chapter 2 Figure 2.8)

4.5.1 FSVM based Multi-Agent System

The agent basically interacts with the other components of the system by manipulating information on the interface agent. The information on the interface agent may represent facts, assumptions, and deductions made by the system during the course of solving the problem. An agent is a partial problem solver which may employ a different problem solving strategy and contribute to the solution by viewing the information on the interface agent. The system has four independent agents: Fuzzy Collision Detector, Pit Detection Agent, Color Sensor and Drive Controller. Note that the arrows in Figure 4.7 represent the flow of information. The diagram shows that all the four agents are allowed to read / write information on the interface agent. Each of the four agents basically executes their tasks independently using the information on the interface agent and posts any result back to the interface agent.

4.5.2 Fuzzy SVM based Collision Avoidance

The agent called the Fuzzy Collision Detector (Carole Fayad and Phil Webb 2002), is a fuzzy SVM-based Collision Avoidance Controller. The fuzzy logic controller has one input fuzzy set for the sensor value and three
output fuzzy sets for linear-distance, velocity and turn-angle. Each set is defined by one or more membership functions that map numeric values onto linguistic terms; each input point may not be fully assigned to one of these two classes. In this thesis, we apply a fuzzy membership to each input point and reformulate the SVMs such that different input points can make different contributions to the learning of decision surface.

The fuzzy-based agent is fed with sensor values as an input, acquired from a set of infrared proximity detectors. The values are fuzzified with designated linguistic terms (near, medium, and far). Among three output fuzzy sets, the turn-angle fuzzy set has been uniquely defined. The angle lies between -30° and 30° which act as a default. The total angle of 60° is divided into six amplitudes represented by six member functions, and each of which is associated with the following linguistic terms: positive-left (PL), negative-left (NL), positive-center (PC), negative-center (NC), positive-right (PR), and negative-right (NR)

Pit Detection Sensor: An apparatus and method for uniquely detecting pits on a smooth surface by irradiating an area of the surface; separately sensing radiation scattered from the surface in the near-specular region indicative of a pit and in the far-specular region indicative of a flaw and producing signals representative thereof; normalizing the near-specular signal with respect to the far-specular signal to indicate a pit.

1. Color detection: Identifying the presence or absence of a specific color;
2. Color measurement: Identifying a color based on its red, green and blue components;
3. Color control: Using the color sensor as part of a closed-loop feedback system to produce and maintain a required color.
4. Drive Controller Agent: The agent primarily holds responsibility for the robot's actuator via the device driver that controls motors through the stepper control module. The agent is made of modules responsible for the motor initialization and termination, the communication between layers, and the maneuvering of the robot.

5. Interface Agent: The interface agent operates as a central repository for all shared information and a communication medium for all agents.

4.5.3 FSVM based Multi-agent Optimal Path Planning

Problems of multi-agent robot systems control have got significance (Rajibul Huq et al 2008). Each multi-agent robot system has some transport subsystem, which consists of several mobile robots. We have a method based on graph optimization algorithms to control such mobile robot group. Novelty of the developed multi-agent path planning algorithm is as follows:

- Mobile robots are considered as dynamic obstacles.
- Graph representation of common environment mode is used for path planning.
- Each edge of the graph has two weights, distance and motion time (speed).
- Weights of edges can be modified during path planning.
- The quickest path is planned (time optimization).
- Expert rules for speed and path correction are synthesized to provide collision avoidance.
These algorithms provide global optimality. Multi-agent path planning algorithm also provides robots collision avoidance. As we have two weights for graph such as distance and time required to travel, we can use AO* algorithm to find shortest path. We need to update the weight of graph during particular interval. Ant colony optimization also can be used to control a group of robot.

4.6 EXPERIMENTATION RESULTS

The navigation technique described was implemented in C and interfaced with the microcontroller using Keil. For the test we specified the start point, the target point and velocity of the point robot. It was repeated for different navigation tests in real time. Figure 4.7 shows the initial and goal position of robots in multi robotic environment. Figure 4.8 shows an example of robot navigation in cluttered environments.

![Figure 4.7 Initial and goal position of robots](image)
Figure 4.8 Robot reaches its goal position
Table 4.1  Comparative performance of fuzzy logic and AVH, ACO and type -2, FSVM

<table>
<thead>
<tr>
<th>Memory(Kb)</th>
<th>Fuzzy and AVH</th>
<th>ACO and Type-2 fuzzy</th>
<th>FSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time in sec for Unknown environment</td>
<td>Time in sec for Unknown environment</td>
<td>Time in sec for Unknown environment</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>4.6</td>
<td>4.2</td>
</tr>
<tr>
<td>20</td>
<td>4.3</td>
<td>4</td>
<td>3.7</td>
</tr>
<tr>
<td>30</td>
<td>3.7</td>
<td>3.7</td>
<td>3.2</td>
</tr>
<tr>
<td>40</td>
<td>3.2</td>
<td>3.2</td>
<td>3.1</td>
</tr>
<tr>
<td>50</td>
<td>2.8</td>
<td>2.6</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>2.6</td>
<td>2.3</td>
<td>1.9</td>
</tr>
<tr>
<td>70</td>
<td>2.5</td>
<td>1.9</td>
<td>1.8</td>
</tr>
<tr>
<td>80</td>
<td>2.5</td>
<td>1.9</td>
<td>1.5</td>
</tr>
<tr>
<td>90</td>
<td>2.4</td>
<td>1.7</td>
<td>1.3</td>
</tr>
<tr>
<td>100</td>
<td>2.2</td>
<td>1.7</td>
<td>1.1</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
<td>1.6</td>
<td>1.1</td>
</tr>
<tr>
<td>120</td>
<td>1.9</td>
<td>1.6</td>
<td>1.1</td>
</tr>
<tr>
<td>130</td>
<td>1.9</td>
<td>1.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 4.1 shows the robot response time for different techniques, in this study we observed that FSVM response time is much lower than other techniques. Fuzzy support Vector Machine is best for robot navigation in unknown and unpredictable multi robotic environment.
4.7 CONCLUSION

The multi-agent and multi-agent optimal path planning approach system was enhanced with Incremental FSVM with Kernel will give better result in group robots. This system deals with the real-time navigation of a mobile robot in a totally unknown environment. Fuzzy SVM is the best tool to incorporate human procedural knowledge in to robot. Type 2 fuzzy gives a better approach to solve uncertainty. Based on fuzzy logic and Support Vector Machine, a collision free technique has been developed.