Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Support Vector Path Planning

Jun Miura
Department of Mechanical Engineering, Osaka University
jun@mech.eng.osaka-u.ac.jp

***Abstract*— This paper describes a unique approach of applying a pattern classification technique to robot path planning. A collision-free path connecting a start and a goal point provides information on the division of the space. In the case of 2D path planning, for example, the path divides the space into two regions. This suggests a *dual* problem of first dividing the whole space into such two regions and then picking up the boundary as a path. We develop a method of solving this dual problem using support vector machine (SVM). SVM generates a non-linear separating surface based on the margin maximization principle. This property is suitable for the purpose of usual path planning problems, that is, generating a safe and smooth path. The details of the path planning methods in 2D and 3D spaces are described with several planning results. Future possibilities of combining the proposed concept with other path planning methodologies are also discussed.**

I. Introduction

Path planning has been one of the important problems in robotics for long years [10]. Path planning is usually regarded as finding a continuous collision-free path, given a start point, a goal point (or a goal region), and obstacles in the space. In earlier works, a path is calculated by searching a graph (e.g., visibility graph [11]) or a grid of free spaces. Construction of free space maps in configuration space and that of search space is, however, sometimes costly especially for path planning in high-dimensional spaces. In recent years, the randomized approaches [7], [1], [5], [4] appear successful in many practical applications which require high-dimensional motion planning. All these works are search-based. Another line of research is potential-based [8], [9], which are more suitable for real-time path planning domains.

Previous works on path planning basically take an approach that a path is constructed from a set of primitive path elements (i.e., a set of unit motions or possible transitions between grid cells). This paper looks at path planning from a different viewpoint. Let us consider a point robot moving among planar obstacles. A collision-free path, connecting a start and a goal point, divides the whole space into two regions: one on the left and the other on the right of the path. This suggests a *dual* problem of first dividing the whole space into such two regions and then picking up the boundary as a path. In more than two dimensional spaces, although a path is not the boundary of two regions, a boundary surface, once obtained, would suggest a possible collision-free path.

Division of the whole space into two regions can be viewed as a two-class classification problem. In the field of pattern recognition, a variety of classification methods have been proposed. Among them, we try to use *support vector machine* (SVM) [14], [3] as a classifier. SVM is one of the powerful classifiers and has been successfully applied to many object recognition tasks such as 3D object recognition [13], face recognition [12], and pattern matching-based tracking [2].

SVM has the following properties that are useful for being used in path planning:

- SVM can generate non-linear separating surfaces, which are suitable for generating smooth paths.
- The idea of margin maximization is suitable for the strategy of seeking safety (keeping away from obstacles) in path planning.
- SVM can calculate optimal separating surfaces with a relatively low cost, in comparison with combinatorial approaches.

This paper describes our support vector mechine-based path planner (called *SVPP*).

II. Support Vector Machine

A. Margin maximization

Support vector machine (SVM) is a binary classification method that finds the optimal separating hyperplane based on the concept of margin maximization [14], [3].

Let $(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)$, $\boldsymbol{x}_i \in R^m$, $t_i \in \{-1, +1\}$ be the training samples separated by a hyperplane $\boldsymbol{w}^T\boldsymbol{x} - h = 0$. If the training data are linearly separable, there exist parameters $\boldsymbol{w}$ and h that satisfy:

$$t_i(\boldsymbol{w}^T\boldsymbol{x}_i - h) \geq 1, \quad (i = 1, \ldots, N). \tag{1}$$

Using such parameters, the two classes are separated by two hyperplanes, $H_1 : \boldsymbol{w}^T\boldsymbol{x} - h = 1$ and $H_2 : \boldsymbol{w}^T\boldsymbol{x} - h = -1$, and no data exist between the hyperplanes. Since the distance between the hyperplanes is $\frac{2}{||\boldsymbol{w}||}$, the optimal parameters are determined by minimizing the objective function:

$$L(\boldsymbol{w}) = ||\boldsymbol{w}||^2/2 \tag{2}$$

under the constraint represented by eq. (1).

A solution to this problem is given by solving the following *dual* problem that is to maximize:

$$L_D(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i\alpha_j t_i t_j \boldsymbol{x}_i^T\boldsymbol{x}_j \tag{3}$$

under the constraints:

$$\sum_{i=1}^{N} \alpha_i t_i \quad = \quad 0, \quad \alpha_i \geq 0 \ (i = 1, \ldots, N). \tag{4}$$

The training data $\boldsymbol{x}_i$ with non-zero α_i are on either one of the hyperplanes $\boldsymbol{w}^T\boldsymbol{x} - h = 1$ or $\boldsymbol{w}^T\boldsymbol{x} - h = -1$; such data are called *support vectors* because they are the only data that determine the parameters. From these data, a discrimination function is given by

$$y = \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x} - h) = \mathrm{sign}(\sum_{i\in\mathcal{S}} \alpha_i t_i \boldsymbol{x}_i^T\boldsymbol{x} - h\,), \tag{5}$$

where $\mathcal{S}$ indicates the set of indices for support vectors.

B. Soft margin

In linearly separable cases, all samples are outside of the region formed by two hyperplanes, H_1 and H_2. If sample data are not linearly separable, we allow some samples to move into the opposite side over one of the hyperplanes. Instead of eq. (2), we use the following objective function:

$$L(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 + \gamma\sum_{i=1}^{N}\xi_i, \tag{6}$$

where $\xi_i \geq 0$ is called a slack variable indicating how much each sample moves into the opposite side, and γ is a weight balancing the margin maximization and the error reduction. The solution to this minimization problem can also be obtained by solving its dual problem, similarly to the linearly separable case.

Non-zero α_i corresponds to the following two cases: for support vectors on either of two hyperplanes, H_1 and H_2, the inequality $0 < \alpha_i < \gamma$ holds; for support vectors inside the hyperplanes, $\alpha_i = \gamma$.

C. Non-linear SVM using kernel tricks

SVMs can be used to determine non-linear separating surfaces using kernel tricks [14], [3]. By using some appropriate kernel function, which calculates inner products in a high-dimensional space, we can change the problem of finding a non-linear separating surface in the *original* space into that of finding a separating hyperplane in the *high-dimensional* space, thus reducing the calculation cost and making it possible to obtain an optimal non-linear separating surface in the original space.

Let π be the mapping from the original to the high-dimensional space. Such a mapping exist for some class of kernel function K, that is:

$$\pi(\boldsymbol{x}_1)^T\pi(\boldsymbol{x}_2) = K(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{7}$$

For the discrimination in the high-dimensional space, the original objective function (eq. (3)) is modified as

$$L_D(\alpha) \;=\; \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{8}$$

and the discrimination function (eq. (5)) is modified as

$$y \;=\; \mathrm{sign}(\sum_{i\in\mathcal{S}}\alpha_i t_i K(\boldsymbol{x}_i, \boldsymbol{x}) - h\,). \tag{9}$$

Since all necessary calculations in the high-dimensional space is given by the calculation of the kernel function in the original space, we can determine non-linear separating surfaces without explicitly representing the high-dimensional space. Fig. 1 shows an example of non-linear classification.

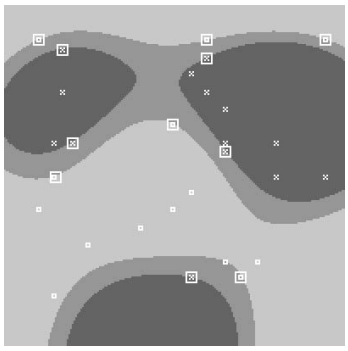

Fig. 1. An example of obtaining non-linear separating surfaces. Small box marks and $\times$ marks indicate positive and negative samples, respectively. Bright, medium, and dark regions indicate positive, intermediate, and negative regions, respectively. Samples with larger boxes are support vectors.

III. SVM-Based Path Planning: 2D case

A. Pre- and post-processing for applying SVM to path planning

We first investigate what processes should be performed for applying SVM to path planning problems. Let V be the output of the following expression (this is actually the argument of the *sign* function in eq. (9)):

$$V = \sum_{i\in\mathcal{S}}\alpha_i t_i K(\boldsymbol{x}_i, \boldsymbol{x}) - h. \tag{10}$$

Based on the value of V, the whole space is divided into the following three regions (see Fig. 1):

- *Positive* region where $V \geq 1$.
- *Negative* region where $V \leq -1$.
- *Intermediate* region where $-1 < V < 1$.

If we can generate these regions so that all obstacles are in positive or negative regions, a collision-free path can be found inside intermediate regions. For this purpose, we divide obstacles into two classes, and positive samples for support vector learning are generated from one class and negative samples from the other. We also set *virtual* obstacles around a start and a goal point so that they lie in intermediate regions.

In general, however, even if both a start and a goal point lie inside intermediate regions, it may not be the case where the two points are in the same intermediate region, because intermediate regions may be divided into several parts as shown in Fig. 1. Such a case happens mainly because enough constraints for dividing the space as expected are not provided due to the scarceness of samples in several places between the start and the goal point. To cope with this, we add several data as *guide samples*.

Once an intermediate region is obtained which contains both a start and a goal point, the next step is to generate an actual path. This is basically done by following the separating surface

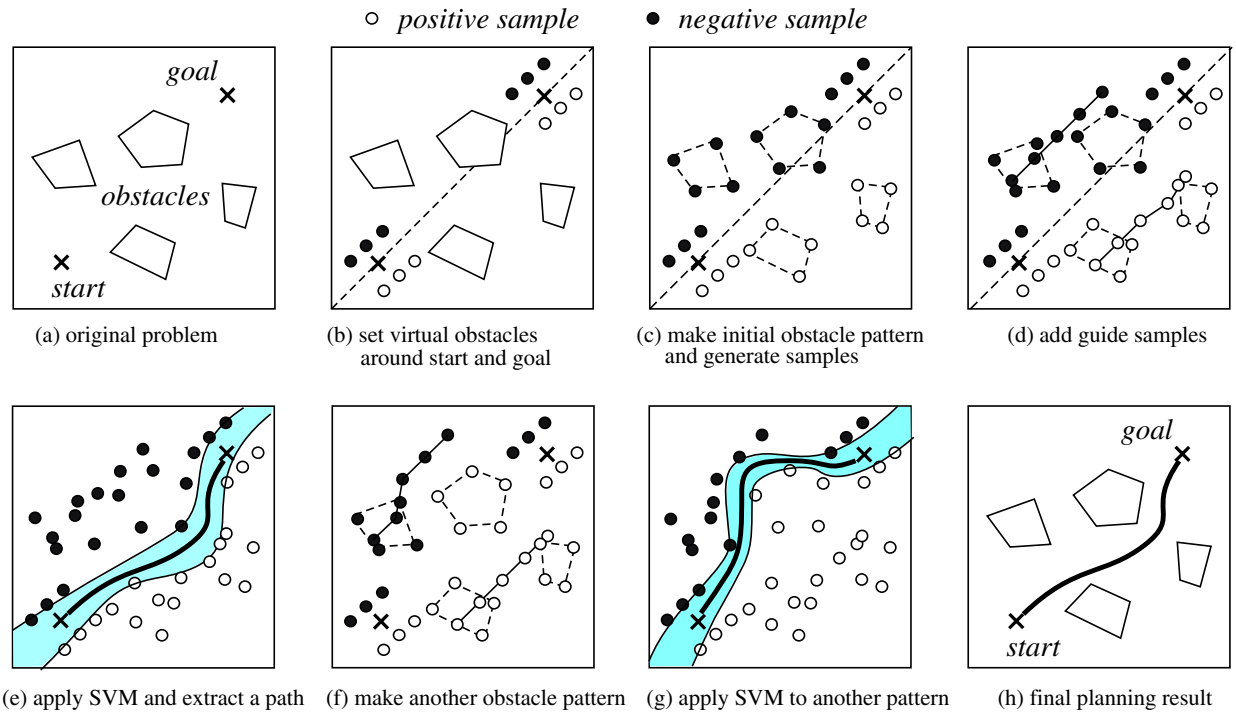

Fig. 2. Steps for SVM-based path planning.

on which the output of SVM is zero. This surface following step can also be used for verifying if a start and a goal point can be connected with each other.

B. 2D SVPP algorithm

1) Outline of the algorithm: All obstacles should be labeled as positive or negative before applying SVM. One way is to test all possible patterns of labels and choose the best one which provides the shortest path. This is, however, evidently intractable when the number of obstacles is large. We therefore take a randomized approach in which a limited number of obstacle (labeling) patterns are randomly selected and tested. For each selected pattern, we generate positive and negative samples and feed them to SVM to calculate a separating surface.

The outline of the algorithm is as follows:

a) Set virtual obstacles around the start and the goal point.
b) Make the initial obstacle pattern and generate positive and negative samples accordingly.
c) Set guide samples to *wrap* the possible traversed region.
d) Apply SVM to the generated samples and extract a feasible path by analyzing the learned model.
e) Try other patterns until the termination condition is satisfied.

We will explain each step in detail using Fig. 2.

2) Setting virtual obstacles around start and goal point: We set virtual obstacles around the start and the goal point so that the intermediate region includes these points (see Fig. 2(b)). More concretely, we calculate the line connecting the points (called the *nominal line*) and put N_v positive (on the right side) and N_v negative ones (on the left) at a certain distance (d_v) from and in parallel to the nominal line.

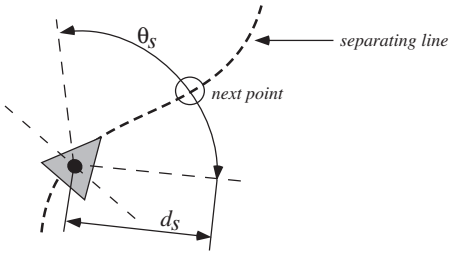

Fig. 3. Search for the next point on the separating surface.

3) Initial obstacle pattern and sample generation: The initial obstacle pattern is determined as follows. If the centroid of an obstacle is on the right (left) side of the nominal line, the obstacle is labeled as positive (negative). From each obstacle, we generate samples at vertices and at midpoints of edges. Fig. 2(c) shows the initial pattern and the generated samples.

4) Setting guide samples: Guide samples are arranged in parallel with the nominal line, with a predetermined distance (d_g) from the line and with a certain spacing (d_p) between samples. Samples connected by solid lines in Fig. 2(d)(f) are guide samples. If an obstacle moves into the opposite side, the guide samples on that side near the obstacle are shifted according to how much the obstacle moves in (see negative (black) guide samples in Fig. 2(f), for example).

5) Applying SVM and analyzing the learning result: We then apply an SVM learning algorithm to the generated sam-

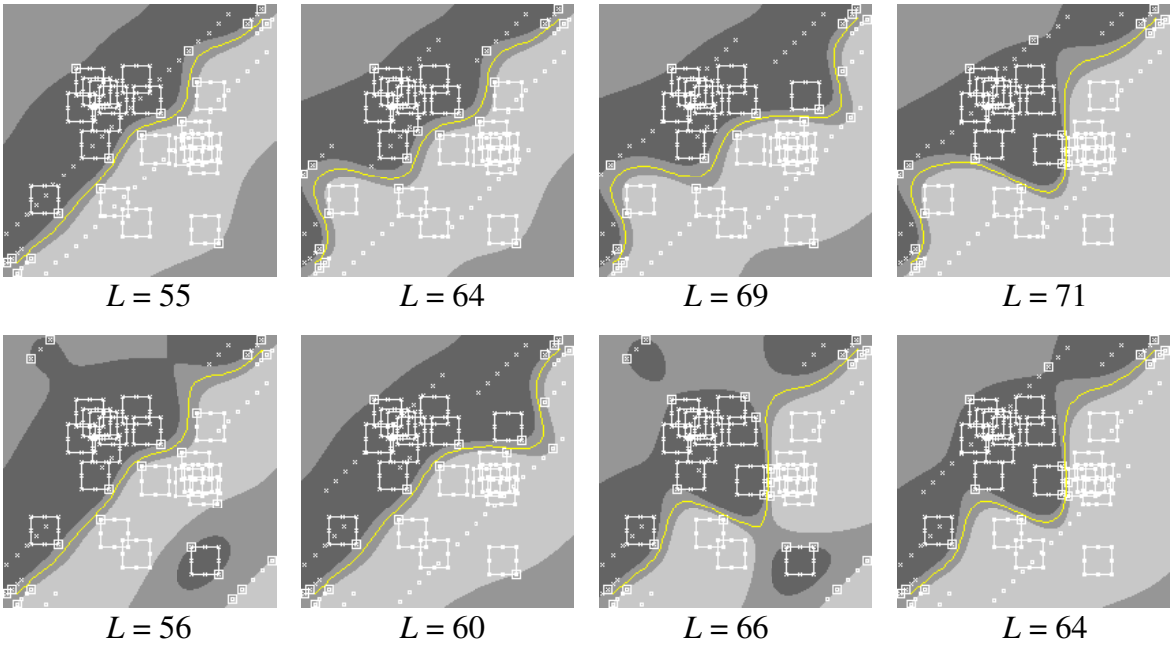

Fig. 4. 2D path planning results.

Large squares indicate obstacles, small box marks indicate positive samples, and × marks indicate negative samples. Marks surrounded by larger squares indicate support vectors. The whole region is divided into three types of regions: the lightest-colored regions indicate "positive", the darkest-colored indicate "negative", and the others indicate "intermediate" that include separating surfaces. The yellow continuous line indicates the planned path. The total path length differs according to the obstacle pattern, and the shortest one is selected as the final planned path.

ples. By this learning, we get a set of support vectors and their weights; they constitute a discrimination function to determine to which region (positive, negative, and intermediate) each point belong.

The separating surface, which is the separating line in 2D, represents candidates of collision-free paths. The surface is not, however, explicitly represented by the model (i.e., support vectors and weights). We therefore determine the separating line by repeatedly searching for the next point where value V in eq. (10) is zero, on the line with a fixed step (see Fig. 3). The step length d_s and the search angle range θ_s are adjusted according to the space size.

By this search, we expect to get a path from the start to the goal point (see Fig. 2(e)). For some obstacle patterns, however, the two points may belong to different "intermediate" regions with each other. So we terminate the above search when the number of steps exceeds a threshold. We also terminate the search when the safe next point, where $|V| < 1.0$, is not found in the one-step search shown in Fig. 3; this case occurs when, for example, the distance to the obstacles on the both side of the path is very small.

6) Trying other patterns: We try to generate paths for several other patterns. We randomly select one obstacle and flip it (from positive to negative or vice versa) to make another pattern. If this pattern is new, we generate samples accordingly and apply the above steps to generate another path. We repeat the path generation process until the number of patterns tried exceeds a predetermined number (N_p) *and* at least one feasible path is found (See Fig. 2(h)).

C. Experimental Results

This subsection describes experimental results. Obstacles of the same size are randomly placed within a square workspace. A set of obstacles that overlap with each other is considered to be a single obstacle. We currently use a publicly-available SVM software, SVM^{light} [6]. The kernel function used is a Gaussian kernel.

Fig. 4 shows a set of planned paths for a path planning problem. In this problem, the size of the square workspace is 2.0^2, the start and the goal point are at $(0.1, 0.1)$ and $(1.9, 1.9)$, respectively; obstacle size is 0.2^2 and the number of obstacles is initially twenty but merged into seven, by considering the overlaps of obstacles. The figure shows the planning results for eight obstacle patterns. Although different patterns generate different paths, a smooth path is generated in all cases. The selected shortest path is the one of the left- and upper-most pattern.

In this simulation, we use the following parameters; the size of Gaussian kernel is 10; the weight for the soft margin is 1000.0; the number N_v of samples for each virtual obstacle is 3; the distance d_v of the samples to the nominal line is 0.05. Concerning guide samples, the distance d_g to the nominal line is 0.2, the spacing d_p between samples is 0.1. The minimum number N_p of the trial on various obstacle patterns is 20. The search angle range θ_s is 120 $[deg]$ and the step length d_s is 0.05 (see Fig. 3).

We examined the calculation time for 10 planning problems with 20 obstacles. The averaged time for applying SVM for one obstacle pattern is about $114.4\,[ms]$ with standard deviation σ=$82.7\,[ms]$, but that for the patterns for which paths were found is about $80.1\,[ms]$ with σ=$31.6\,[ms]$; discrimination is easier for such patterns and thus takes less time and less uncertainty for applying SVM. The averaged time for path generation for such patterns is about $61\,[ms]$ with σ=$9.8\,[ms]$. The total planning time depends on the minimum number of trials, N_p; for $N_p = 20$, the averaged total planning time is about $10.0\,[s]$ with $\sigma = 2.69\,[s]$.

IV. SVM-Based Path Planning: 3D case

A. Determining nominal plane

In the 2D case, the line connecting a start and a goal point (nominal line) divides the whole space into the regions. This property is used for setting virtual obstacles, determining an initial obstacle pattern, and setting guide samples. In the 3D case, however, the line does not have such a property. So we add one more dimension in the direction having the largest diversity of obstacle positions. This direction is determined by applying PCA (principal component analysis) to the set of obstacle vertices; i.e., determined as the direction of the eigenvector with the largest eigenvalue.

We consider the plane which includes the nominal line and whose normal is the eigenvector mentioned-above. Using this plane, we divide the whole 3D space into two regions. We call this plane the *nominal plane*.

B. 3D SVPP algorithm

1) Setting virtual obstacles around start and goal point: We set two planes on both sides of the nominal plane around the start point or the goal point, and generate virtual obstacles (and thus samples) on these planes. Samples on one side are positive and those on the other are negative.

2) Initial obstacle pattern and sample generation: The obstacles are divided into positive and negative classes based on the position of centroid with respect to the nominal plane, and the obstacles in the positive (negative) class are labeled as positive (negative). Samples are generated at not only vertices of obstacles but also their edges and faces.

3) Setting guide samples: We set a pair of planes parallel to and on both sides of the nominal plane, at the same distance. Samples are generated on these planes to cover an enough area of the planes. If an obstacle moves into the opposite side, the guide samples on that side near the obstacle are shifted along the direction of the normal of the nominal plane, according to how much the obstacle moves in.

4) Applying SVM and analyzing the learning result: We apply the same SVM learning algorithm to all generated samples and obtain a set of support vectors and their weights. As in the 2D case, we search for a path using these data. The search strategy is, however, more complex in the 3D case because we have to determine not only the separating surface but also the moving direction on that surface.

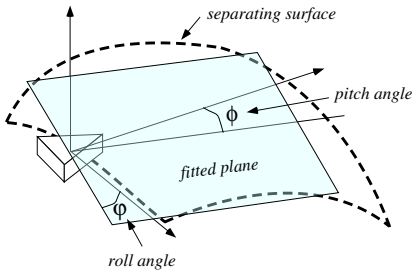

Fig. 5. Local plane approximation of the separating surface.

To search for a path, we first fit a plane to the separating surface within a nearby region of the robot. This fitted plane is determined by searching the space of the pitch and the roll angles for the best plane which minimizes the sum of the distances between the plane and the surface (see Fig. 5). Once this plane is determined, we project the nominal line onto the plane and select a point on the projected line.

C. Experimental results

Fig. 6 shows a result of 3D path planning. The size of the cubic workspace is 2.0^3, the start and the goal point are at $(0.1, 0.1, 0.1)$ and $(1.9, 1.9, 1.9)$, respectively; the number of obstacles, whose size is 0.26^3, is initially twenty and merged into twelve. We examined the calculation time for 4 planning problems with 10 obstacles; the minimum number of trials N_p is set to 10. The averaged time for applying SVM for one obstacle pattern is about $175.8\,[ms]$ with σ=$65.0\,[ms]$, but that for the patterns for which paths were found (i.e., easier patterns) is about $147.6\,[ms]$ with σ=$39.7\,[ms]$. The averaged time for path generation for such patterns is about $3.3\,[s]$ with σ=$0.28\,[s]$. The averaged total planning time is about $1\,[min]\,20\,[sec]$ with $\sigma = 10.0\,[sec]$; most of planning time is spent for the path generation step.

V. Combination of SVPP with Other Methodologies

The current SVPP algorithms are not efficient enough to be used alone. This section therefore seeks several possibilities of combining the SVPP concept with other path planning methodologies.

A. Generating a Smooth Path

Search-based path planning methods usually generate a piecewise linear path. In controlling a robot, a smooth path is preferable for safety and stability. SVM, which can generate non-linear separating surface, can thus contribute to generating a smooth path. Given a piecewise linear path, classification of obstacles into positive or negative is easily performed. For an feasible classification pattern, it takes a relatively small cost to calculate a separating surface and generate a smooth path, as previously shown.

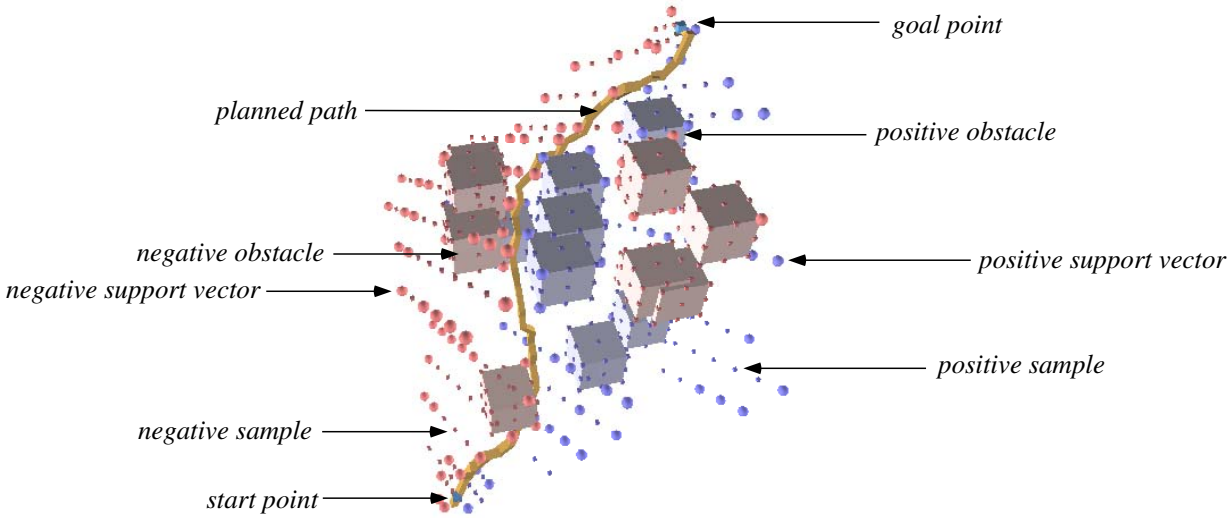

Fig. 6. A planning result in 3D.

Blue means positive and red means negative. Large cubes are obstacles, (very) small cubes are samples, and small spheres are support vectors. The yellow line indicates the planned path.

B. Using the Discrimination Function as a Potential

Path generation from a start point to a goal point is currently done by a search-based method and is costly. Since the method basically follows a separating surface (or a line in 2D) and the distance to the surface is evaluated by the discrimination function (more specifically, the absolute value of V in eq. (10)), $|V|$ (or a monotonic function of $|V|$) can be used as a potential function; a lower potential means more distant from obstacles. We could design a potential function combining the potential $|V|$ and those defined by the position of the destination.

C. Using Support Vectors as Graph Nodes

Support vectors used for determining a separating surface also carry important information for planning a safe path. Even if a generated set of support vectors is not sufficient for generating a feasible path due to an inappropriate obstacle classification, some of the vectors may carry useful information for generating a part of a feasible path. Therefore, support vectors generated by applying SVM to several classifications can be, for example, candidates for the nodes in a visibility graph-based path planner; such a preprocessing using SVM may be able to reduce the total planning cost.

VI. Conclusions

This paper has described a unique approach of applying SVM to robot path planning. SVM has a nice property that it can generate a continuous non-linear separating surface between labeled data samples. We have described the pre- and post-processing for using this property in robot path planning in 2D and 3D. Applications to higher-dimensional path planning and reduction of planning cost are future works. We have also discussed future possibilities of combining SVPP concept with other path planning methodologies.

References

[1] N.M. Amato and Y. Wu. A Randomized Roadmap Method for Path and Manipulation Planning. In *Proceedings of 1996 IEEE Int. Conf. on Robotics and Automation*, pp. 113–120, 1996.

[2] S. Avidan. Support Vector Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 8, pp. 1064–1072, 2004.

[3] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 121–167, 1998.

[4] S. Caprin and G. Pillonetto. Robot Motion Planning Using Adaptive Random Walks. In *Proceedings of 2003 IEEE Int. Conf. on Robotics and Automation*, pp. 3809–3814, 2003.

[5] D. Hsu, R. Kindel, J.C. Latombe, and S. Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. *Int. J. of Robotics Research*, Vol. 21, No. 3, pp. 233–255, 2002.

[6] T. Joachims. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. The MIT Press, 1999.

[7] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probablistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 4, pp. 566–580, 1996.

[8] O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. of Robotics Research*, Vol. 5, No. 1, pp. 90–98, 1986.

[9] J.-O. Kim and P.K. Khosla. Real-time Obstacle Avoidance using Harmonic Potential Functions. *IEEE Trans. on Robotics and Automation*, Vol. 8, No. 3, pp. 338–349, 1992.

[10] J.-C. Latombe. Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts. *Int. J. of Robotics Research, Special Issue on Robotics at the Millennium – Part I*, Vol. 18, No. 11, pp. 1119–1128, 1999.

[11] T. Lozano-Perez and M.A. Wisley. An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles. *Comm. of ACM*, Vol. 22, No. 10, pp. 560–570, 1979.

[12] T. Okabe and Y. Sato. Support Vector Machines for Object Recognition under Varying Illumination Conditions. In *Proceedings of 2004 Asian Conf. on Computer Vision*, pp. 724–729, 2004.

[13] M. Pontil and A. Verri. Support Vector Machines for 3D Objec Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 6, pp. 637–646, 1998.

[14] V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.