International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **4959–4978**

SAFE AND SMOOTH: MOBILE AGENT TRAJECTORY SMOOTHING BY SVM

Chan-Yun Yang[1], Jr-Syu Yang[2] and Feng-Li Lian[3]

[1]Department of Mechanical Engineering
Technology and Science Institute of Northern Taiwan
No. 2, Xue-Yuan Rd., Beitou, Taipei 11202, Taiwan
cyyang.research@gmail.com

[2]Department of Mechanical and Electro-Mechanical Engineering
Tamkang University
No. 151, Yingzhuan Rd., Tamsui Dist., New Taipei City 25137, Taiwan
096034@mail.tku.edu.tw

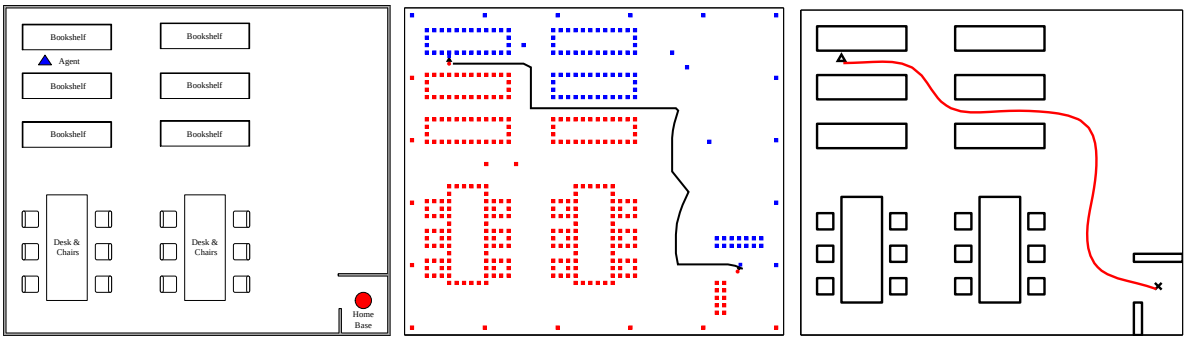
[3]Department of Electrical Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
fengli@ntu.edu.tw

Received April 2011; revised October 2011

Abstract. *The paper presents a model merging Voronoi tessellation with an underlying support vector machine (SVM) in order to develop path planning for guiding an autonomous vehicle safely and smoothly through a space with obstacles. Being a roadmap method for path generation, the Voronoi tessellation is employed as a preprocessor to roughly fit a connection between the initial and goal configurations. Though the Voronoi path is safe for obstacle avoidance, its disjoint linear edges are unsatisfactory when smoothness is requested. Hence, an SVM postprocessor is proposed to make the segmented path smoother. By analogue to the Gaussian potential field, a zero-potential curve in the configuration space is thus obtained by the SVM postprocessor and forms a safe and smooth path. Due to advantages of the SVM with RBF kernel, the post-processed path has the merits of both smoothness from the Gaussian kernel basis, and wide clearance from the large-margin decision boundary. This paper adopts point configurations to represent obstacles in the working space. With additional artificial points for auxiliary constraints, the two-stage path planner is then developed. Detailed property investigations and simulated applications are also included to characterize the path planner. The results of a practical demonstration show a promising future for further applications.*
Keywords: Path planning, Safe, Smooth, Large margin, Support vector machines

1. **Introduction.** Emerging robotic technologies have developed rapidly and expansively in the past two decades due to both software/hardware innovations and application requests. Human beings often look for surrogates to take their place in challenging environments, especially environments that they cannot approach or work in [1]. A typical development for an autonomous ground agent, which is ordinarily abstracted as a mobile robot in pursuit of its own agenda, is driven automatically. The guidance of the agent together with its path planning, including the considerations of obstacle avoidance, route selection, and driving safety, is one of the most important issues. This type of agent to be guided in the configuration space C from initial position I to an objective position O with an important goal can basically be just two-dimensional. Moreover, an agent devised as a service facility in a real environment [1,2] prefers a smooth and safe prescribed path to follow. As the library application illustrated in Figure 1 shows, an agent can be

developed to carry books or even persons for library automation. The agent is generally requested to provide smooth service in all its navigation missions. To fulfill this goal, we introduce an SVM-based trajectory smoother to produce a path that is smooth and has wide clearances through the two-dimensional configuration space.



(a) Obstacle layout (b) Voronoi path (c) Smoothed path

FIGURE 1. Practical application for a library

This study begins with "waypoints" selected from the configuration-free space, C_{free}, by means such as visibility graphs [3], Voronoi diagrams [4], probabilistic roadmaps [5], or rapidly exploring random tree [6], to provide a candidate for a globally planned path. For safety and efficiency, a continuous smooth approximated path, rather than a path made of straight line segments connected by the waypoints, is preferred. To provide a smooth path, various smoothing techniques have been proposed to manipulate the waypoints, such as a combination of segments and circular arcs [7-9], cubic spirals [9], clothoids [10,11], algebraic polynomial splines [12-14], polar splines [12,15], Bezier's curevs or B-splines [11,16-20], trigonometric splines [19], or G3 curves [21,22]. Fraichard and Scheuer [10] have generally divided the smooth paths from these techniques into two categories: one is the curves whose coordinates have a closed-form expression, such as B-splines, algebraic polynomial splines, and polar splines, and the other is parametric curves whose curvature is a function of their arc length, such as cubic spirals and clothoids. Regardless of differences in their form, the admissible smooth paths must generally fulfill the basic necessary conditions of continuity in displacement, velocity, and acceleration of the agent. This is the reason that the algebraic polynomial splines or B-splines are generally maintained to be at least third order to ensure continuity. For example, quintic polynomials [23] or cubic B-splines [24] have been adopted for obtaining smooth, jerk-bounded trajectories of industrial robot manipulators. Although the manipulator is different from a ground vehicle, the law of the order higher than three is still significant. Instead of these categories of smoothing technique, radial basis functions, abbreviated as RBF's, are excellent alternatives due to their ability to produce smooth paths. RBF is defined as a linear combination of radically symmetric basis functions, each centered on a particular point. For approximation, the basis function responds locally in the vicinity of the center in comparison to the other functions, which have a global response. To avoid influence of responded residues, piecewise approximation is generally adopted with the global-responded smoother. However, the smoother also occasionally sacrifices its global smoothness to the piecewise approximation. In contrast, an approximation function composed of the basis functions is relatively intact, and is advantageous to the continuity in the entire planned path. Typical references for the basis-function based path planner are presented as in [25-27].

Regarding safety, there are several related papers concerned with obtaining a wide path clearance. In [28], an adaptive Hopfield neural network together with a traditional approach of visibility graphs, simple or generalized Voronoi diagrams, or decomposition methods are used to maximize the clearance of path. Unfortunately, the two-stage study was developed without consideration for smoothness of the planned path. In [29], a multiphase planner provides a parametric model for generating a safe path. An intermediate phase compound of a generalized Voronoi diagram and a consecutive "maximal inscribed discs" procedure is adopted to ensure that the adopted Voronoi paths are wide enough for free passage. Based on the same technique of a Voronoi diagram, Bhattacharya and Gavrilova developed a planner for clearance-based shortest path [30]. This planner is made by filtering out all those edges in the Voronoi diagram which have a clearance less than the minimum clearance required, and applying Dijkstra's algorithm to determine the shortest path in the roadmap. With its margin maximization scheme, the SVM shows excellent performance in obtaining the wide path clearance. Miura [31] proposed the subsequent non-linear separating surface generated from the SVM as an agent path. This was the first discovery of the relationship between path planning and SVM, but due to a lack of suitable preprocessor, that paper shows that direct use of SVM cannot succeed every time.

Intrinsically, a path planning task is the problem of searching for a feasible navigation from I to O. Besides the classical methods [32], techniques of computational machine intelligence, e.g., SVM [33], is introduced to offer particular solutions for ensuring a successful navigation in various scenarios. Wang and Tsai [34] introduced a modified least-mean-square-error classifier, which is similar to our idea, to generate a collision-avoidance path in which wall and obstacle boundaries are prescribed as labeled 2D points in the discriminating feature space. Although their approach is different from ours, the results are still helpful to us.

This paper adopts the Voronoi diagram as a preprocessor to generate a temporal path for further process. In addition to its rapid computational response, an elementary Voronoi path connecting a sequence of waypoints of the Voronoi diagram generally provides wide clearance for agent passing through since those waypoints are always measured equidistantly to the nearest obstacle sites from the definition. Due to these properties, the Voronoi diagram has been frequently adopted as a preprocessor to generate a temporal path in the staged scheme [28-30].

Inheriting intrinsically the excellent smoothness of RBF, a path planner employing the RBF kernel based SVM is able to produce an agent path which majors in its intact smoothness rather than those produced by the approaches listed in the table. On the other hand, with the SVM optimized large margin, the planned path also shares the merit of a wide clearance for the agent passage. Motivated by these merits, this study merges the underlying SVM with a selected Voronoi preprocessor to generate a safe and smooth path for the mobile agent.

2. **Preliminary Primitives.**

2.1. **Mountainous terrain generated by SVM.**

2.1.1. *SVM classifier.* Based on statistical learning theory [35], an SVM based on maximizing the margin [36] has been well developed for the classification and regression learning problem. To employ the SVM in path planning, a related concept is here illustrated first. Assume there is one set of prototypes, $\mathrm{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$, acquired from the 2D space S, $S \in \Re^2$, in which $\mathbf{x}_i$ is a prototype in S, $\mathbf{x}_i = [x_{i1}, x_{i2}]^{\mathrm{T}}$. The term "prototypes", referring to the 2D sample points in the SVM input space, can also be mapped

to the obstacle singletons that are scattered in the obstacle configuration space, and will be explained in Subsection 3.1.1. As a supervised learning scheme, prototypes $\mathbf{x}_i$'s in S have been labeled in advance as y_i, $y_i \in \mathrm{Y} = \{1, -1\}$. The theory of SVM classification, learning with the labeled prototypes, gives then an optimized solution to produce a decision boundary $f(\mathbf{x}) = 0$. Figure 2 shows an example of a linear case, $f(\mathbf{x}) = \mathrm{w}^{\mathrm{T}}\mathbf{x} + b$, where w denotes the vector normal to the decision boundary and b the bias. The decision boundary tends to separate the whole set of prototypes into two categories according to their labels, i.e., it places prototypes with different labels on either side, $f(\mathbf{x}) > 0$ and $f(\mathbf{x}) < 0$, respectively. Though there are methods to generate a variant decision boundary, the boundary generated by the SVM classifier shares the merit of having large margins [37]. The margin, as the defined width of the maximal clearance that the classifier can achieve for separation from the obstacle, can be illustrated as the street clearance along the decision boundary (Figure 2). Intuitively, with a larger margin, there is wider clearance for the agent to pass through.

For this study, a non-linear SVM is more practical for real world applications. Employing the kernel technique [38] for high dimensional similarity manipulation, the linear SVM can be extended to a non-linear SVM. The idea is that a non-linear problem can be solved linearly when mapping the discriminant feature space into a high dimensional reproducing kernel Hilbert space (RKHS) H. The kernel function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j) \tag{1}$$

manipulates the similarity of the feature map, $\varphi(\mathbf{x})$: $\Re^2 \to \Re^H$. Thus instead of the infeasibility of the direct manipulation of the inner product of the high dimensional feature maps, the kernel function implicitly computes the inner product without mapping into such a high-dimensional space, thereby reducing the corresponding computations. With the kernel trick, the quadratic optimization problem of SVM can be abbreviated eventually as follows:

$$\arg\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i\alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{2}$$

$$\text{subject to} \quad 0 \le \alpha_i \le \lambda, \ \forall i, \ \text{and} \tag{3}$$

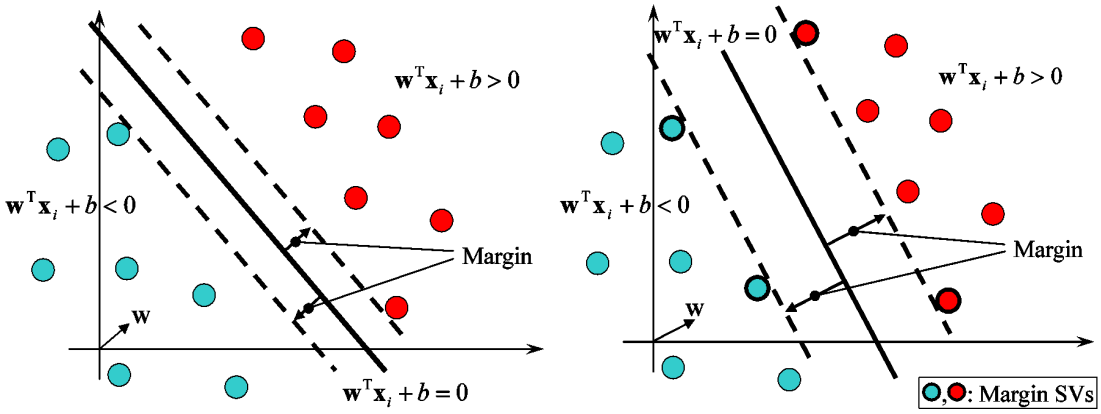
$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{4}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_l]^{\mathrm{T}}$ denotes a set of Lagrange multipliers, and λ denotes a scaling factor to control the trade-off between the maximum margin and the minimum total remaining error of the trained classifier. From the KKT conditions, only those prototypes conforming to the KKT complementarities, having non-zero α_i and referred to as support vectors (SVs), contribute to the solution. The SVs can be categorized more specifically as the margin SVs and non-margin SVs [36]. This fact simplifies the formation of the eventual decision function with the optimized SVs over steps of optimization of (2)-(4):

$$f(\mathbf{x}) = \sum_{\mathrm{SVs}} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \tag{5}$$

where $\mathrm{SVs} = \{i | \alpha_i > 0, \forall i\}$.

2.1.2. *Gaussian RBF kernel.* Several types of kernel function are frequently employed to solve classification problems, for example, the linear, polynomial, and radial basis function (RBF) kernels [41]. Due to a strong relationship to the path planning of the agent, the



(a) Small margin (b) Optimized large margin

FIGURE 2. A large margin can be achieved by the SVM

RBF kernel is particularly re-examined for its use to this purpose. In general, the RBF kernel, also known as the Gaussian kernel, is given as follows:

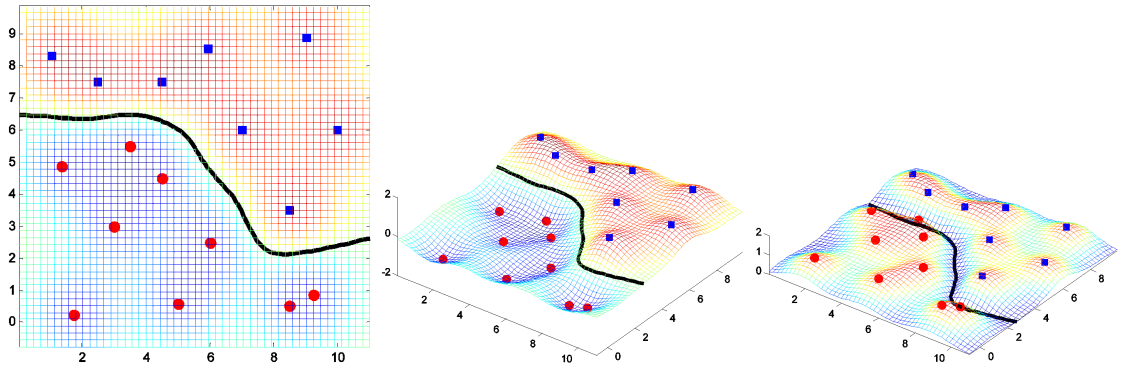
$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \tag{6}$$

where σ denotes the width parameter of the Gaussian function. With the RBF kernel, the decision function (5) can further be re-written as follows:

$$f(\mathbf{x}) = \sum_{\text{SVs}} y_i \alpha_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\sigma^2}\right) + b \tag{7}$$

Except for the bias term b, the decision function $f(\mathbf{x})$ is a function composed of a linear combination of scaled Gaussian functions of the SVs. This turns the three-dimensional profile of $f(\mathbf{x})$ into an analogue of the potential field of a mountainous terrain.

2.2. **Zero-potential curve through mountainous terrain.** With the RBF kernel, the SVM, which is generally used to produce a smooth curve for classification, can also generate a safe and smooth path for the agent. As illustrated in Figure 3(a), a planar non-linear decision curve $f(\mathbf{x}) = 0$ is generated by the RBF kernel SVM. The planar separation can become 3D as the viewing is changed from an inclined angle for a clear observation (Figure 3(b)). As shown, the separation line is in fact a zero-potential (solid black) curve passing through the mountainous terrain like potential field. Negative labeled prototypes on the left-hand side of the curve exhibit negative potential, whereas positive labeled prototypes on right-hand side exhibit positive potential. Along the separation curve, equal attitudes of the positive and negative potentials would be canceled and form the zero-potential curve. For path planning, the zero-potential curve can also be used as an optimized path for an agent passing through the field if the scattered prototypes are considered as obstacles in configuration space. This idea can be understood more easily if the negative potential is reversed as the same upward orientation of the positive potential (Figure 3(c)), where the path is fitted with a road between the high areas. This path deserves particular attention because it is generated by the large margin SVM. This large margin, analogous to a clearance zone tolerance for the agent's movement, is crucial for achieving a safe path.



(a) Planar view (b) 3D view (c) Reversed negative potential

FIGURE 3. Zero-potential curve produced by SVM

2.3. **Voronoi preprocessor.** At this point, a safe and smooth path has been achieved with the given labeled prototypes. But a method used as a preprocessor to label the prototypes is required since the labels of those prototypes are initially unknown in the study. The preprocessor is like a thumbnail sketch outlining a rough path between I and O, called P_{Vor}. The preprocessor prepares only the rough path and the temporary labels for the prototypes. This study first employs a Voronoi tessellation [4] to construct a graph, and then extracts the rough path from the tessellation. With the simplified version of point-configured obstacles (the unlabeled point prototypes $\mathbf{x}_i$'s, where $\mathbf{x}_i \in \mathrm{X}$, in this case), the Voronoi tessellation [4] can be given as follows:

$$\Gamma = \{\mathbf{z} | \mathbf{z} \in C_{\text{free}}, \exists \mathbf{x}_i, \mathbf{x}_j \in C_{\text{obst}}, \mathbf{x}_i \neq \mathbf{x}_j, s(\mathbf{x}_i, \mathbf{z}) = s(\mathbf{x}_j, \mathbf{z})\} \tag{8}$$

where

$$s(\mathbf{x}, \mathbf{z}) = \min_{\mathbf{x} \in C_{\text{obst}}} \text{distance}(\mathbf{x}, \mathbf{z}) \tag{9}$$

produces a set of convex polygons Γ such that each polygon contains exactly one source prototype, and every point of a produced polygon is closer to its source prototype than to any others. For convenience, we refer to the vertices of the polygon edges as waypoints since they could be visited in the path tracking. The Voronoi path P_{Vor} is hence defined as an arbitrarily picked path following the edges of Γ from a waypoint near position I to a waypoint near position O. The Voronoi tessellation can still be applied with the presence of blocking obstacles. Nothing changes except that the minimal distance s, from $\mathbf{z}$ to those obstacles $\mathbf{x}$ in (9), is measured from the boundary if the block-obstacle is presented.

3. **Modeling the Planner.** The planner, given a procedure with known I, O, and obstacles, is devised to search for a safe and smooth path in the C_{free}. The steps of the planner, as those given in Figure 4, will be individually described in the followings.

3.1. **Map formatting.** As stated, we used the prototypes as inputs of the system. Hence, formatting the obstacles in the working space into a class of prototypes should be considered first. In addition, additional prototypes are needed to restrict the development behavior and generate a qualified path. At least three types of prototypes are included in this study to represent the configuration space obstacles, C_{obst}: the real obstacles, pseudo-obstacles for outer frame, and endpoint footcuffs, as described below.

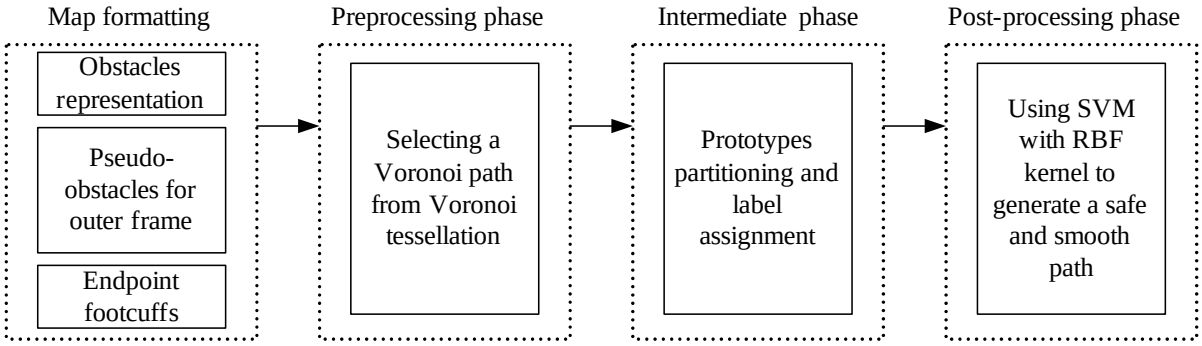


FIGURE 4. Flowchart of the model

3.1.1. *Obstacles representation.* The SVM basically requires the individual prototypes in the input space. Obstacles in the working space hence must be decomposed as those scattered points, here referred to as obstacle singletons, for converting the working space into the planner input space to meet the requirement. Algorithm 1 made the conversion, and abstracted the scattered sparse obstacle singletons as the prototypes.

Algorithm 1 (Obstacle singleton conversion):
Input: Map of working space
Output: A class of individual singletons for planner input space

1. In the map of the working space, extract and decompose individually the obstacles to form C_{obst}, including the singleton-obstacles and block-obstacles.
2. Further decompose the block-obstacles into sets of definite obstacle singletons.
3. Collect all the obstacle singletons to form a set of natural prototypes for input.

Referring to (7), the final decision function f can be given by a superposition of scaled Gaussian functions in addition to the obstacle singletons. The decision function is, in fact, the final path for the agent. For a small individual obstacle, the singleton can be extracted simply by means of an impulse function located at the center of the obstacle. For larger obstacles, it can be expressed as a cluster of close singletons (Figure 5). In this study, we use a group of singleton grids which are placed together with a fixed gap p to represent a large obstacle. Hence, the block-obstacle can be decomposed into many obstacle singletons, and furthermore also abstracted as prototypes. The prototype generated from either obstacle singletons or block-obstacles is called natural prototype to distinguish them from the other type of auxiliary artificial prototypes.

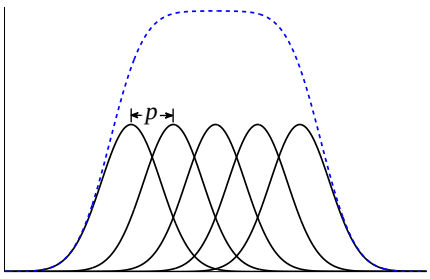


FIGURE 5. Multiple singletons form a large obstacle

3.1.2. *Pseudo-obstacles for outer frame (POOF).* A set of artificial singletons is added in the planner input space to extend the creation of Γ. As defined, Γ is a network created by means of the growth of adjacent convex polygons containing exactly one prototype inside. In general, only the prototypes within the scattered area have a chance to complete the given convex polygon, and those close to the border of the area can not. Without an outer reference, the open polygon can not be completed, and an incomplete edge may break a candidate Voronoi path if the path prefers the incomplete edge for passing through. To improve this situation, the set of auxiliary artificial prototypes around the border of the planner input space is generated as the references to complete all the convex polygons of the natural prototypes (Algorithm 2). The auxiliary artificial prototypes are called pseudo-obstacles for the outer frame, abbreviated as POOF's. Although a magnified silhouetted contour extended in the shape of the distribution of the scattered natural prototypes is ideal for locating the POOF's, for convenience we simply use a rectangle to situate the POOF's in this study. The POOF's are situated equidistant to the border natural prototypes, and they inherit all the same properties of natural prototypes during the path planning.

Algorithm 2 (POOF's generation):
Input: Natural prototypes, I, O, and δ
Output: A set of POOF's

1. Find the two-dimensional coordinate maximum $[\mathbf{x}_{\max 1}, \mathbf{x}_{\max 2}]^{\mathrm{T}}$ and minimum $[\mathbf{x}_{\min 1}, \mathbf{x}_{\min 2}]^{\mathrm{T}}$ of I, O, and all the natural prototypes.
2. Use vertices $[\mathbf{x}_{\max 1} + \delta, \mathbf{x}_{\max 2} + \delta]^{\mathrm{T}}$, $[\mathbf{x}_{\max 1} + \delta, \mathbf{x}_{\min 2} - \delta]^{\mathrm{T}}$, $[\mathbf{x}_{\min 1} - \delta, \mathbf{x}_{\max 2} + \delta]^{\mathrm{T}}$, and $[\mathbf{x}_{\min 1} - \delta, \mathbf{x}_{\min 2} - \delta]^{\mathrm{T}}$ to form a rectangle.
3. Generate a set of auxiliary artificial prototypes, POOF's, on the rectangle with a fixed gap or equal division.

3.1.3. *Endpoint footcuffs.* The endpoint footcuffs are used to restrict the space that is feasible for a clear passage near endpoints I and O (Figure 6). Generally, the decision boundary is optimized mainly with a wide margin, but it is difficult to limit the SVM-optimized path to pass through exactly a certain point in the planner input space in advance. The difficulty is severe since a planned path with either one missing endpoint, either I or O, is not allowed for the agent. The footcuffs are hence employed to force the planned path to pass through the endpoints. Supported by the SVM, the generated smooth path is formed by a maximizing margin that is equidistant to the margin SVs, i.e., those SVs riding exactly on one of the margin (Figure 2). Using this idea, one pair of auxiliary artificial prototypes with opposite labels are placed symmetrically each separated from the endpoints by the same small equal distance $d/2$ (Figure 6). Because of the small distance between the paired prototypes, they will behave as the margin SVs during the optimization like footcuffs to restrict the generated path. The orientation of the footcuffs can be controlled by the angle θ, $-\pi/2 \leq \theta \leq \pi/2$, which is measured from the endpoint to the first connected waypoint. In this study, the default value for θ is set to $\pi/2$. It can be seen that the footcuffs on either side of the endpoints, as those shown in Figure 6, should be given different labels for producing equivalent adversary forces to urge the SVM decision boundary to pass through the endpoint precisely. The distance d in this case should be small enough for precise alignment of the SVM decision boundary and the endpoint. Algorithm 3 shows the procedure for adding the footcuffs.

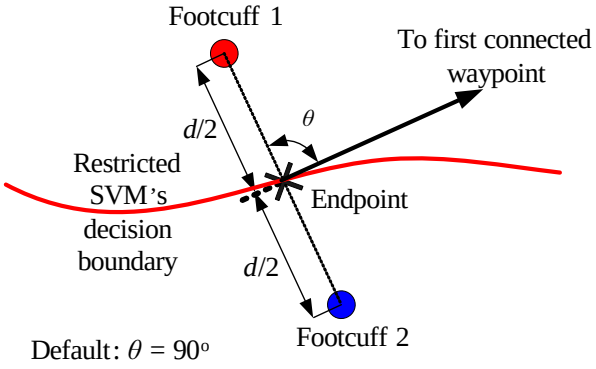


FIGURE 6. Adding footcuffs to the endpoint

Algorithm 3 (Endpoint footcuffs placement):
Input: Endpoints I, O, and P_{Vor}
Output: Paired footcuffs for each endpoint
Parameter: d and θ

1. Place sequentially paired footcuffs each spaced symmetrically from endpoints I and O by equal distance $d/2$ and orientation θ.
2. Assign the labels of all prototypes to be 1 and -1 according to the separation of P_{Vor}.

3.2. **Voronoi path.** Once a Voronoi tessellation is built using both the obstacle singletons and the POOF's, a candidate P_{Vor} can in principle be selected freely from the tessellation. There are several possible algorithms for this. To generate a short path, this paper adopts a strategy to consecutively select the edges only when their forward direction is the closest direction from I to O. This selection criterion guarantees that the piecewise linear path is as short as possible. Further considering the large clustered obstacle blocks, those clusters consisting of several singletons are spaced apart by fixed gap p, as shown in Figure 5. Then the agent taking the candidate path passing through the gaps narrower than p should be rejected to avoid a collision with the corresponding large obstacle block. To avoid a large obstacle, the final P_{Vor} may make a detour around the outskirts of the block-obstacles, but takes a truly correct route to O. The algorithm for a valid P_{Vor} is abbreviated as Algorithm 4.

3.3. **Prototype partitioning and labeling.** Although P_{Vor} is not the smooth path we want, we use it temporarily to apportion the prototypes into two classes, to which the classification of SVM can be applied. In other words, all three kinds of prototypes should be labeled prior to the SVM training, and the labels are given according to the partitions that P_{Vor} has divided (Algorithm 3). Since the SVM is intrinsically a supervised learning classifier, it generates a separating boundary by learning from the adversary labeled prototypes. Tracing back the idea, the unlabeled prototypes can conversely be labeled using a known separating boundary even though the boundary is naïve. The prototype labels are hence given easily by the partitions formed by P_{Vor}. To finish the labeling, artificial POOF's should be considered further. Since P_{Vor} ends up at both I and O, the partition cannot cover the POOF's (Figure 7), and the separating boundary should therefore be extended from I and O to the space border. In this study, the default directions along the edges connected with the endpoints are adopted for the extension.

Algorithm 4 (Voronoi path selection):
Input: Natural prototypes, POOF's, I, and O
Output: P_{Vor}

1. Generate a Voronoi tessellation Γ with the natural prototypes and POOF's
2. Initial a queue P_{Vor} with first edge E_{I1}, the edge from I to its most closest waypoint $\mathbf{z}_1$.
3. $i = 1$, $s_{\min} = \infty$.
4. Evaluate all the waypoints $\mathbf{z}_j$ having candidate edges connected to $\mathbf{z}_i$, and record all $\mathbf{z}_j$, $j = 1$ to l, in a list L.
5. For $j = 1$ to l
   a. Evaluate all the possible passing gaps across $E_{iL(j)}$, and record all the gaps' length p_k, $k = 1$ to m, in a list M.
   b. For $k = 1$ to m
      If $p_{M(k)} \leq p$ then return to step 4.
   c. If Euclidian distance $s_{L(j)O} < s_{\min}$ then $j\min = L(j)$ and $s_{\min} = s_{L(j)O}$.
6. Save the edge $E_{ij\min}$ to P_{Vor}.
7. $i = j\min$.
8. Repeat steps 3-6 until the $\mathbf{z}_i$ is the closest waypoint to O.
9. Save the edge E_{iO} to P_{Vor}, and return P_{Vor}.

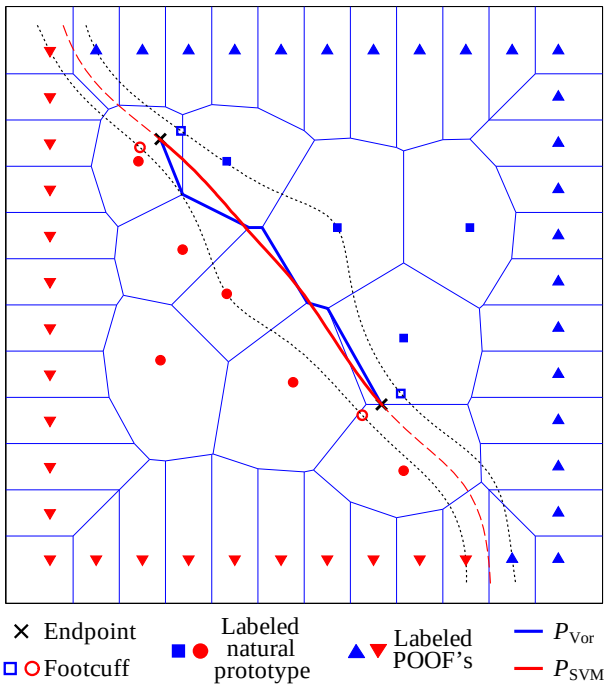


FIGURE 7. Three kinds of labeled prototypes corresponding to the generated smooth path

Figure 7 demonstrates an example P_{Vor} with two distinct colors to label the subsequent prototypes. As shown in the figure, not only the natural prototypes but also the artificial footcuffs and POOF's should be labeled for further processing. The figure also uses dashed lines to envelop the space which is clearance for the SVM postprocessor.

3.4. **SVM path smoother.** The SVM decision function f in (7) is in turn used straightforwardly to generate the smooth path, P_{SVM}, with the labeled prototypes. As expected, the smoothing has taken effect as shown by the red solid line in Figure 7. As described in the previous section, the RBF kernel is the key to ensure the smoothing effect. It should be noted that the P_{SVM} is obtained by the SVM optimizer. The smooth and wide clearance of P_{SVM} is generated with no involvement from the Voronoi preprocessor. Indeed, the Voronoi preprocessor provides only a temporal reference for labeling the prototypes. But any kind of preprocessor capable of providing the rough reference can be a substitute for the Voronoi preprocessor, and can generate the same equivalent result.

The example case of Figure 7 shows that the planned path proceeds equidistant from the dashed lines is as much as possible in the corresponding enveloped space, and passes through the endpoints nearly perfectly. This meets the goals of this study – a safe and smooth road for the agent.

4. **Model Characterization.** To assess the performance of the proposed model, a series of experiments have been drawn. Scenarios of the obstacle layout, as simple imitations of the real-world environment of the agent, were prepared not only for validation, but also for feedback and recursive improvement of the model. For quantitative comparison of the smoothness, absolute tangential variations of the overall planned P_{SVM} are averaged as a metric, called V_S, to assess the corresponding smoothness of the path. To calculate V_S, the eventual P_{SVM} is first discretized into a sequence of connected uniformly-spaced linear segments. The V_S then sequentially measures the slopes of the discretized segments, which amount to an aggregate value (Algorithm 5). For an unprejudiced comparison, V_S's of various paths in comparison must be calculated with the same number of segment-divisions. Since a path can not always be exactly divided, after discretization there is sometimes a remainder close to the endpoints. This remainder is generally ignored due to its insignificant influence in the V_S calculation.

Algorithm 5 (V_S calculation):

Input: P_{SVM}, segment counts l
Output: V_S

1. Lay down P_{SVM} in the orientation for which the reference line connecting I and O in Figure 8 is aligned horizontally.
2. Discretize horizontally oriented P_{SVM} as a sequence of uniformly-spaced linear segments connected by vertices $\mathbf{p}_1, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_l$.
3. For $i = 1$ to $l - 1$
   a. Measure $dx_{i,1}$ and $dx_{i,2}$ of the consecutive vertices $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, as shown in Figure 8, and calculate $tv_i = |dx_{i,2}/dx_{i,1}|$.
4. Obtain V_S by $V_S = 1/(l-1)\sum_{i=1}^{l-1} tv_i$.

The V_S aggregates elementary tv_i's which reflect the significance of instantaneous direction changes through the segmental slopes, and this reveals the roughness of the path. In short, the smoother the path, the smaller the V_S.

4.1. **Comparison with cubic spline.** As presented in Section 1, splines are usually used to derive a smooth path with some available waypoints. A comparison to the spline methods is hence made with arbitrarily chosen scenarios. For example, the paths of P_{NCS} and P_{CBS}, which are generated by the methods of natural cubic spline [40], and cubic B-spline with open uniform knot vector [12], respectively, are depicted with known P_{Vor} in

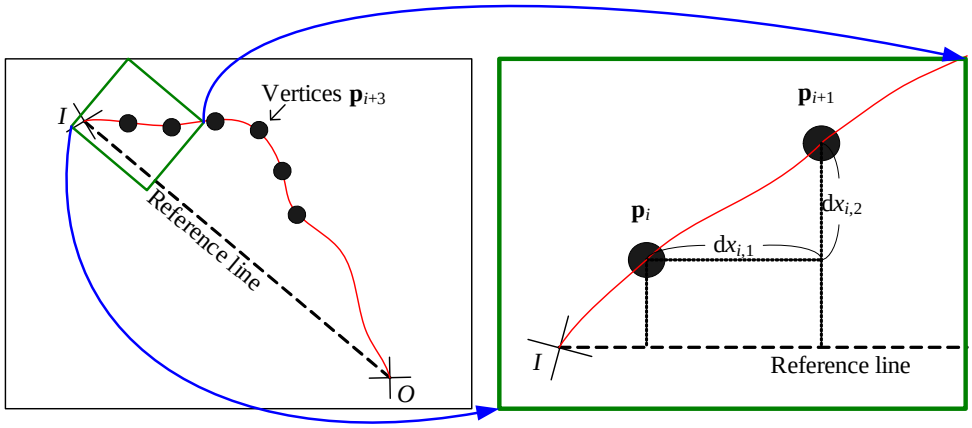


FIGURE 8. V_S calculation

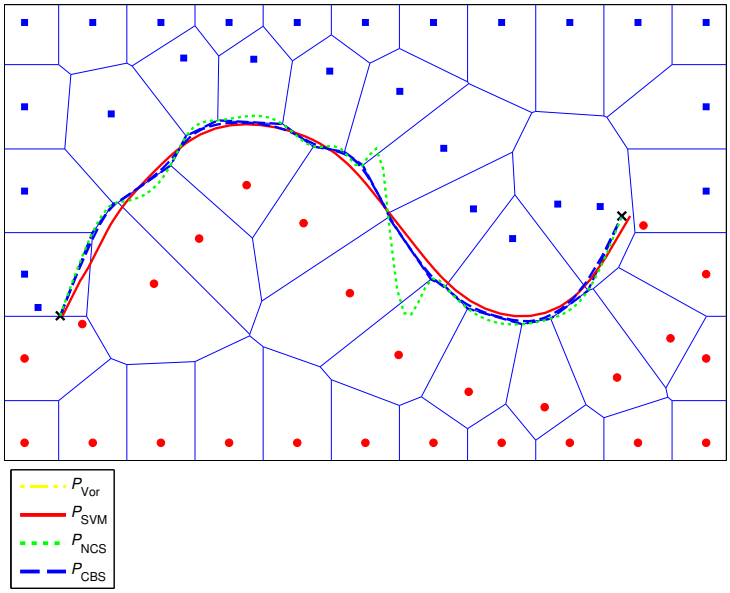


FIGURE 9. Comparison with splines

the scenario of Figure 9 for comparison. As known, the smoothing splines adopt piecewise curves to interpolate the waypoints. All the piecewise curves should be generated locally following the steps of P_{Vor}, even if the P_{Vor} is segmentally rugged. The splines' dependency on the P_{Vor} can be observed from Figure 9. On the contrary, the P_{SVM} generated with $\lambda = 10$ and $\sigma = 2$ (Figure 9) can be more globally smooth because it is wholly optimized once with the whole set of prototypes. In fact, the global intact smoothness is a general property of P_{SVM} despite the difference in parameter settings. The intact smoothness makes the agent-tracking easier. Moreover, there is one another benefit of adopting the P_{SVM}. Since the generation of P_{SVM} relies on the available labeled prototypes, the precise waypoints from P_{Vor} for obstacle avoidance are apparently no longer the key for the safe path generation. Instead of the heavy dependency on P_{Vor} for generating P_{NCS} or P_{CBS}, the P_{SVM} itself generates the path with a wide clearance. Nevertheless, it should be noted that the improvement is gained at the cost of increased computing time (Table 1). The computational tradeoff is mainly from the SVM optimizer and should be reduced to an allowable scale when applied to real applications. In addition, Table 1 also shows the advantages of shorter path length and lower V_S of the P_{SVM}.

TABLE 1. Comparisons with splines

Path	V_S	Length	Smoothing Time
P_{SVM}	0.83	63.52	6.38×10^{-1}
P_{NCS}	1.31	78.57	3.79×10^{-3}
P_{CBS}	0.91	67.19	7.33×10^{-3}

4.2. **Effect of parameters λ and σ.** Referring to the SVM model (2)-(4), there are two parameters for tuning P_{SVM}. Parameter selection for the SVM is critical when it is employed for solving classification problems, while for path planning, the selection is still important, but not critical. To explain this, observations from changing the parameters with certain scenarios are depicted in Figure 10. From the viewpoint of SVM, the penalty factor λ is used for penalizing the misclassified prototypes in the trade-off optimization. The trade-off was originally devised to balance the expected margin and the summed error from the misclassified prototypes. In this study, all the prototypes, both natural and auxiliary artificial, have been adequately labeled and clearly partitioned prior to the optimization. No contaminated prototype is generated, i.e., there is no misclassification for trade-off in the subsequent SVM learning. The selection of λ is hence not critical; indeed, a sufficiently large λ suffices for the learning. Here, a relatively degenerated λ is generally adopted to maintain the balance between the flexibility of the curve and the influence of the obstacles, e.g., the exponential grids in a range $[1, 1 \times 10^3]$.

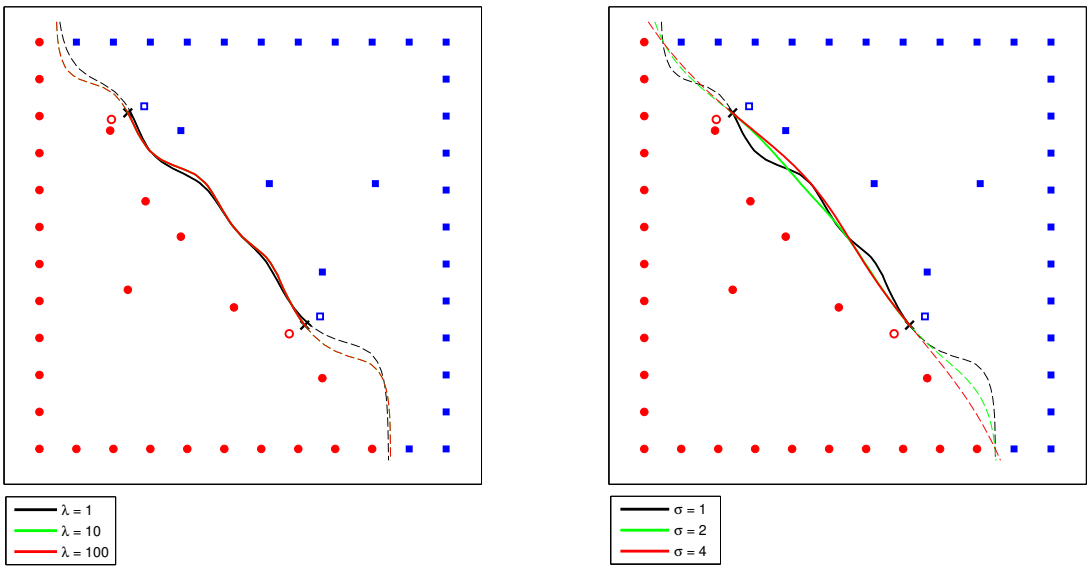
The selection of σ, which is the width parameter of the RBF kernel, is more difficult since it significantly affects the appearance of the induced path. In general, σ mainly determines the terrain unevenness of the planner input space, as shown in Figure 3. Whereas a loose value of σ may not reflect the terrain unevenness, an exaggerated value would smear the terrain surface. In short, SVM is used to manipulate the wide-margin path through the terrain. As shown in Figure 10(b), the variation of the planned paths differs with the settings of σ. A large value of σ generally produces a smoother path with narrower variation than does a small value. But however smooth the path will become, the accompanying smeared surface may cause the path to drift far from the neutral position. The corresponding measures of V_S and path length for variables λ and σ are shown in Table 2. To avoid over-smoothing and keep the flexibility in principle, a small σ is generally adopted. By experiments, σ is selected roughly as 1 and refined in the range $[0.5, 4]$ in this study.

TABLE 2. Influence of parameters λ and σ

λ	σ	V_S	Path Length	Smoothing Time
1	1	1.72×10^{-1}	7.84	1.50×10^{-1}
10	1	2.19×10^{-1}	8.03	1.21×10^{-1}
100	1	2.19×10^{-1}	8.03	1.19×10^{-1}
100	1	2.19×10^{-1}	8.03	1.15×10^{-1}
100	2	0.67×10^{-1}	7.82	2.81×10^{-1}
100	4	1.11×10^{-1}	7.86	2.54×10^{-1}

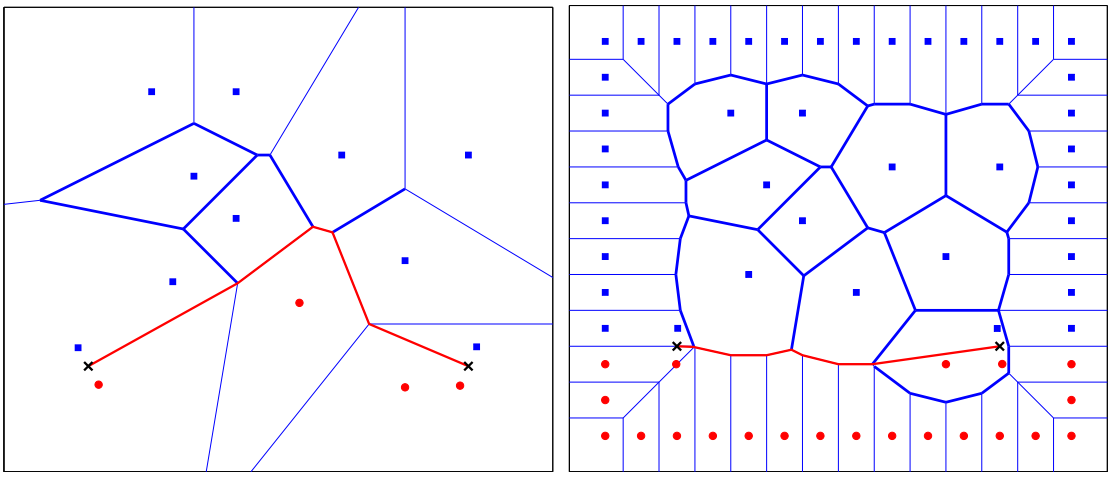
4.3. **Serving without POOF.**

4.3.1. *Effect to Voronoi path.* As presented above, the candidate P_{Vor} that prefers any outer edge of an incomplete convex polygon may be broken if the POOF's were not sufficient. This is the first reason for adding POOF's prior to the generation of the P_{Vor}.



(a) $\sigma = 1$ (b) $\lambda = 100$

FIGURE 10. Effect of parameters λ and σ. Both green and red lines are fully overlapped in (a).



(a) without POOF's (b) with POOF's

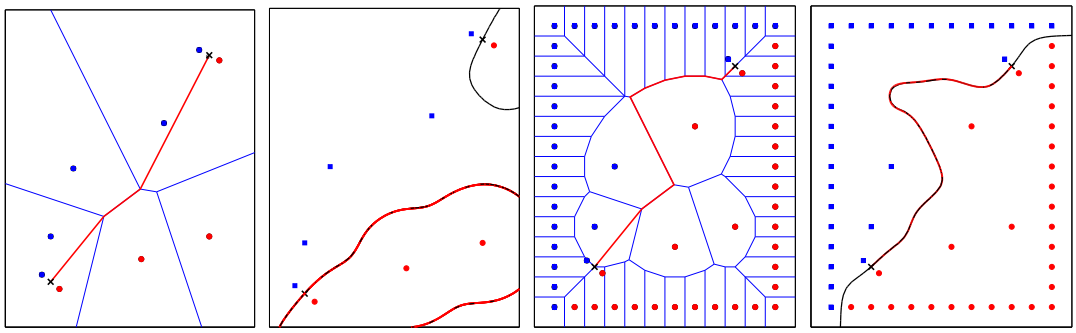
FIGURE 11. P_{Vor} with and without POOF's

Figure 11 shows the P_{Vor} generated with and without the POOF's. Rather than an indirect detour generated without POOF's (Figure 11(a)), a short and direct P_{Vor} is generated with POOF's (Figure 11(b)). Figure 11(b) shows that every natural prototype is enclosed by a fully closed convex polygon due to the surrounding POOF's that guarantee the short and direct P_{Vor}.

Conceptually, it requires more vertices to form the P_{Vor} since short fragments benefit the smoothness of the path in the first stage. Unfortunately, the increasing number of vertices also increases the computation load of the system, which makes it difficult to determinate the number of POOF's. The difficulty is then reduced when we face to the independency between the P_{SVM} and P_{Vor}. No matter how rugged the P_{Vor} is, the eventual smoothness comes from the SVM optimizer itself. Hence, only an adequate number of

POOF's is asserted for the system. For example, the POOF's are generally arranged with a fixed gap $p = 1$ and $\delta = 2$ in Algorithm 1 of this paper.

4.3.2. *Effect to SVM path.* In fact, the P_{SVM} is actually a decision boundary learned from the prototypes. This means that the effectiveness of the P_{SVM} depends substantially on the distribution of the prototypes, especially the natural prototypes. The ideal of distribution is that the adversary labeled prototypes can clearly be separated along the orientation from I to O. The distribution creates repulsive potentials of approximately equal magnitude on the opposite sides to form P_{SVM}. Unfortunately, this requirement may not be adequate due to poor or imbalanced distribution of the prototypes. One example of such an unfavorable condition is shown in Figures 12(a) and 12(b). Due to the sparsity, P_{SVM} is broken into fragmental curves which enclose only a part of the prototypes locally. The broken P_{SVM} eventually fails, although an acceptable Voronoi path has been initially selected. The unfavorable condition can be improved by adding the surrounding POOF's. Figures 12(c) and 12(d) show the improvement of the above example. An adequate continuous smooth P_{SVM} is therefore generated due to the added POOF's. Learning with the added POOF's, the SVM optimizer would urge the generated P_{SVM} to enclose as many adversary prototypes as possible, draw extensively a continuous P_{SVM} across the whole input space, and, hence, reduce the sparsity. Similar to the explanation above, the additional POOF's also prevent the system from the other problem of imbalanced learning set [41].



(a) without POOF's (b) P_{SVM} from (a) (c) with POOF's (d) P_{SVM} from (c)

FIGURE 12. P_{SVM} with and without POOF's

4.4. **Changing footcuffs.**

4.4.1. *Gap d between footcuffs.* The footcuffs, as defined in Figure 6, are used mainly as constraints to force the P_{SVM} to pass through endpoints I and O. The gap d between the footcuffs hence determines the allowed clearance for the P_{SVM} passing through. As shown in Figure 6, when the allowed clearance shrinks, the curvature of P_{SVM} near the location of I or O is relatively restricted in a narrow range. In general, the P_{SVM} prefers to have less restriction by the footcuffs, but excessive relaxation of the restriction would cause the P_{SVM} to miss endpoint I or O. Thus, the restriction from an adequate gap d gives the P_{SVM} an exact hit to both I and O.

An experiment is made keeping $\theta = \pi/2$ constant and only varying d from 0.2 to 1 for observation. Figure 13 shows that the P_{SVM} is changed slightly by an increasing value of d. As expected, the flatness from a larger d is greater than that from a smaller one. In conclusion, data with a gradual change of d is digested and listed in Table 3. Dramatically, a sufficiently large value of d, e.g., $d = 1$, benefits not only the V_S and path length, but

also the SVM computing time. By observation, the generated P_{SVM} is also relatively smooth, with $d = 1$. This is consistent with the fact that a smoother decision boundary implies less model complexity in the SVM [35], and is preferred by the computing time.

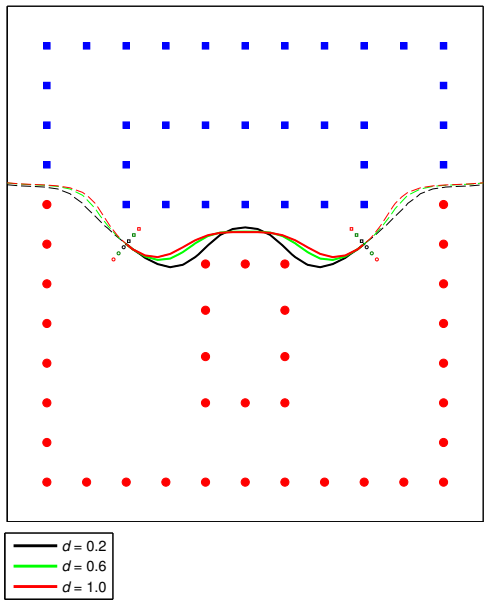


FIGURE 13. Paths by different d

TABLE 3. Effect of changing d

D	V_S	Length	Smoothing Time
0.2	5.23×10^{-1}	6.86	2.82×10^{-1}
0.4	4.19×10^{-1}	6.60	2.67×10^{-1}
0.6	3.62×10^{-1}	6.48	2.52×10^{-1}
0.8	3.28×10^{-1}	6.40	2.50×10^{-1}
1	3.05×10^{-1}	6.35	2.50×10^{-1}

4.4.2. *Orientation θ of footcuffs.* The orientation of footcuffs also affects the appearance of the P_{SVM}. As described in Subsection 3.1.3, a setting of $\theta = \pi/2$ is adopted for default orientation of the footcuffs at I and O for a perpendicular direction of the agent start or stop. In general, the perpendicular direction is often the direction of P_{SVM} in those two positions. In contrast to the most effortless orientation straightforward from I to O, the default direction is not always optimistic for an ideal P_{SVM}. Keeping $d = 1$, Figure 14 illustrates that different settings of θ may slightly change the P_{SVM} using the same obstacle layout of Figure 13. The related V_S value and path length are recorded in Table 4. The table shows that the change of θ effectively alters the path smoothness. We should also notice that an over-adjustment of θ may have a bad effect. As presented in the table, the path of $\theta = \pi/4$ becomes longer due to the over-adjustment, even though the corresponding V_S is still small. Comparing all the paths in Figure 14, the P_{SVM} with $\theta = 0.3\pi$ together with $d = 1$ produces the best smooth path for the agent to pass through.

4.5. **Case simulations.** Four cases to generally characterize the planned path are presented in Figure 15. The cases with different conditions show that the presented method is effective for producing a safe and smooth path. Two types of obstacle layout, the distributed layout and the blocked layout, are drawn in the case study. As shown first in

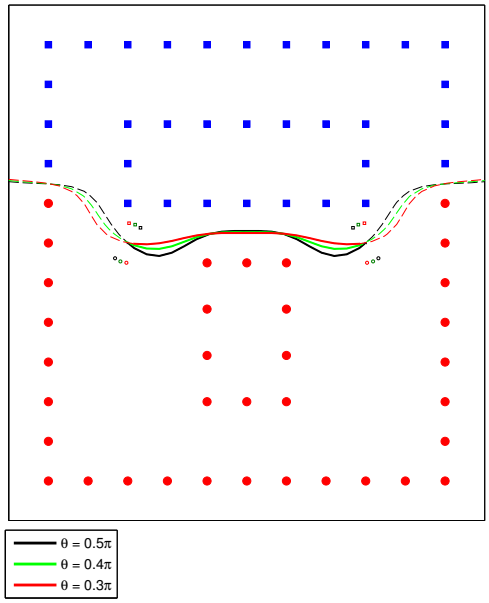


FIGURE 14. Paths by different θ

TABLE 4. Effect of changing θ

θ	V_S	Length	Smoothing Time
$\pi/2$	3.05×10^{-1}	6.35	2.50×10^{-1}
0.45π	2.47×10^{-1}	6.16	2.44×10^{-1}
0.40π	1.87×10^{-1}	6.01	2.36×10^{-1}
0.35π	1.37×10^{-1}	5.89	2.34×10^{-1}
0.30π	1.03×10^{-1}	5.78	2.34×10^{-1}

Figures 15(a) and 15(b), a path is successfully produced with different locations of I and O using the same distributed obstacle layout. Rather than the distributed obstacle layout, appropriate paths of blocked obstacles are also achieved as shown in Figures 15(c) and 15(d). Though the paths fulfilled the safe and smooth criteria, there is still a deficiency when the curve turns around an obstacle that was far from the neighboring POOF's, as shown by the dashed-line encircled area in Figure 15(d). This path made a detour when it ran into such an area. The detour is not an error but only a defect that comes with the wide clearance between the obstacle and its neighboring POOF's. A similar deficiency can also be found in Figure 15(a). The deficiency may be resolved by a tightly magnified silhouette for the POOFs' layout, as mentioned in Subsection 3.1.2. Although silhouetting the POOF's is a potential solution for such a deficiency, it should be strictly considered with the corresponding side effects. For example, a tightly silhouetted POOF's restricts the allowed clearance for path planning, and is disadvantageous for the safety of wide clearance in the paper.

5. **Application Remarks.**

5.1. **Deficiency and limitation.**

Although the planner is studied in detail, it still suffers from the deficiencies of increased computational complexity and increased model parameters for selection up to 4. As shown in Table 1, two orders of computational time are required as tradeoff for the global smoothness of P_{SVM}. As known, the computational complexity of P_{SVM} depends significantly on the number l of input prototypes [39], and would be increased approximately as the order of $\mathrm{O}(l^2)$ comparing to that of $\mathrm{O}(l)$ of

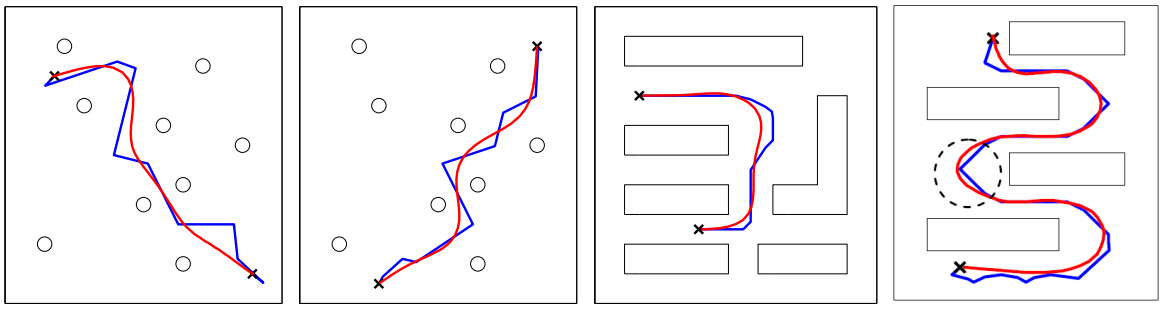


FIGURE 15. Case simulations. P_{Vor}: blue lines, P_{SVM}: red lines. (a) & (b) With distributed obstacles, setting $\lambda = 10$, $\sigma = 2$, $d = 0.5$, $\theta_{\text{I}} = \pi/2$, and $\theta_{\text{O}} = \pi/2$. (c) With blocked obstacles, setting $\lambda = 10$, $\sigma = 2.5$, $d = 0.5$, $\theta_{\text{I}} = \pi/2$, and $\theta_{\text{O}} = \pi/2$. (d) With blocked obstacles, setting $\lambda = 10$, $\sigma = 2$, $d = 0.75$, $\theta_{\text{I}} = 0.44\pi$, and $\theta_{\text{O}} = 0.17\pi$.

a cubic spline smoothing. Although we can generally manipulate with a finite number of input prototypes, the computational time is still high due to the extra computation required for optimization progress and iterative parameter selection. To reduce the deficiencies, future studies are suggested, such as: using interior point algorithms to solve the convex optimization problem [42], adopting sequential minimal optimization with subset division to reduce the computational complexity [43], re-using the solution obtained from specified parameter setting to speed up the parameter selection [36], and adopting early stopping criterion for optimum searching [36]. In addition, the classification-based scheme limits the planner to serving only in a 2D space. To overcome the limitation, future study of adopting support vector regression to substitute support vector classification is also suggested. It should be noted that the computational improvement is optimistic and may be limited since the P_{SVM} is intrinsically derived from the solution of a parametric optimization problem.

5.2. **Practical demonstration.** To demonstrate the applicability of the proposed planner, practical validations by a Lego NXT wheeled-robot have been implemented. The implementation was completed by a common system consisting of an Intel mobile Celeron dual-core T1400 (at 1.73GHz) processor based computer, a higher place USB webcam to detect the distribution of obstacles and monitor the behavior of the robot, and Bluetooth transmit for wireless commands to the Lego NXT. Samples of the validation have been posted in the YouTube websites: www.youtube.com/watch?v=Hu345Kz8jlg, and v = dduRzfN-rI. Although the system is not powerful in computation and communication, this preliminary validation shows the system is still practical for path planning and achieves the expected robot passage.

6. **Conclusions.** An SVM trajectory smoother has been successfully developed to provide a safe and smooth path for a mobile agent. With the most crucial property of induced smooth and wide clearance, the path planner is uniquely qualified for this purpose. The demonstrated two-stage model, merging SVM and Voronoi tessellation, realizes the planner as an admissible model to achieve the goal of this paper. Instead of the Voronoi preprocessor, it is noteworthy that the proposed SVM smoother may potentially be extended more generally to be combined with a preprocessor, which can provide a similar rough reference for partitioning and labeling the prototypes.

Acknowledgment. The authors would like to express their gratitude to the National Science Council of Taiwan, for the support through project NSC 98-2221-E-149-001, NSC 98-2221-E-002-160-MY3, and the Ministry of Education, Taiwan. They also want thank the graduated students Feng-Yi Chou and Jhih-Hao Ma for their consecutive help in the developing period.

REFERENCES

[1] W. C. Frank, J.-C. Thill and R. Batta, Spatial decision support system for hazardous material truck routing, *Transportation Research Part C*, vol.8, pp.337-359, 2000.

[2] I. Waheed and R. Fotouhi, Trajectory and temporal planning of a wheeled mobile robot on an uneven surface, *Robotica*, vol.27, no.4, pp.481-498, 2009.

[3] T. Lozano-Pérez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Communications of the ACM*, vol.22, no.10, pp.560-570, 1979.

[4] D. G. Kirkpatrick, Efficient computation of continuous skeletons, *Proc. of the 20th IEEE Annual Symp. Foundations of Computer Science*, San Juan, Puerto Rico, pp.18-27, 1979.

[5] L. Kavraki, P. Svestka, J. Latombe and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE Trans. Robotics and Automation*, vol.12, no.4, pp.566-580, 1996.

[6] S. M. LaValle, Rapidly-exploring random trees: A new tool for path planning, *Tech. Rep. 98-11*, Department of Computer Science, Iowa State University, 1998.

[7] T. Ohuchi, T. Ohtomo, N. Satoh and N. Yamada, Transfer movement control of mobile robot using two circular arcs, *Systems and Computers in Japan*, vol.19, no.9, pp.10-20, 1988.

[8] J. A. Reeds and L. A. Shepp, Optimal paths for a car that goes both forwards and backwards, *Pacific J. Mathematics*, vol.145, no.2, pp.367-393, 1990.

[9] Y. Kanayama and B. I. Hartman, Smooth local-path planning for autonomous vehicles, *Int. J. Robotics Research*, vol.16, no.3, pp.263-284, 1997.

[10] T. Fraichard and A. Scheuer, From Reeds and Shepp's to continuous-curvature paths, *IEEE Trans. Robotics*, vol.20, no.6, pp.1025-1035, 2004.

[11] D.-H. Shin and A. Ollero, Mobile robot path planning for fine-grained and smooth path specifications, *J. Robotic Systems*, vol.12, no.7, pp.491-503, 1995.

[12] W. Nelson, Continuous curvature paths for autonomous vehicles, *Proc. of IEEE Conf. Robotics and Automation*, Scottsdale, AZ, USA, vol.3, pp.1260-1264, 1989.

[13] B. Nagy and A. Kelly, Trajectory generation for car-like robots using cubic curvature polynomials, *Proc. of the Int. Conf. Field and Service Robots*, Helsinki, Finland, 2001.

[14] E. Papadopoulos, I. Papadimitriou and I. Poulakakis, Polynomial-based obstacle avoidance techniques for nonholonomic mobile manipulator systems, *Robotics and Autonomous Systems*, vol.51, pp.229-247, 2005.

[15] M. Hentschel, D. Lecking and B. Wagner, Deterministic path planning and navigation for an autonomous fork lift truck, *Proc. of the 6th IFAC Symp. on Intelligent Autonomous Vehicles*, Toulouse, France, 2007.

[16] K. Komoriya and K. Tanie, Trajectory design and control of a wheel-type mobile robot using B-spline curve, *Proc. of Int. Workshop on Intelligent Robots and Systems*, Tsukuba, Japan, pp.398-405, 1989.

[17] A. Segovia, M. Rombaut, A. Preciado and D. Meizel, Comparative study of the different methods of path generation for a mobile robot in a free environment, *Proc. of the 5th Int. Conf. Advanced Robotics*, Pisa, Italy, vol.2, pp.1667-1670, 1991.

[18] H. Eren, C. C. Fung and J. Evans, Implementation of the spline method for mobile robot path control, *Proc. of Instrumentation and Measurement Technology Conf.*, Venice, Italy, pp.739-744, 1999.

[19] E. Dyllong and A. Visioli, Planning and real-time modifications of a trajectory using spline techniques, *Robotica*, vol.21, pp.475-482, 2003.

[20] E. Koyuncu and G. Inalhan, A probabilistic B-spline motion planning algorithm for unmanned helicopters flying in dense 3D environments, *Proc. of IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Nice, France, pp.815-821, 2008.

[21] C. G. L. Bianco, A. Piazzi and M. Romano, Smooth motion generation for unicycle mobile robots via dynamic path inversion, *IEEE Trans. Robotics*, vol.20, no.5, pp.884-891, 2004.

[22] A. Piazzi, C. G. L. Bianco and M. Romano, $\eta 3$ splines for the smooth path generation of wheeled mobile robots, *IEEE Trans. Robotics*, vol.23, no.5, pp.1089-1095, 2007.

[23] S. Macfarlane and E. A. Croft, Jerk-bounded manipulator trajectory planning: Design for real-time applications, *IEEE Trans. Robotics and Automation*, vol.19, no.1, pp.42-52, 2003.
[24] J. Yang, J. Kim, E. P. Pitarch and K. Abdel-Malek, Optimal trajectory planning for redundant manipulators based on minimum jerk, *Proc. of ASME Int. Design Engineering Technical Conf.'s and Computers and Information in Engineering Conf.*, New York, USA, pp.1141-1150, 2008.
[25] G. Bugmann, P. Robinson and K. L. Koay, Stable encoding of robot paths using normalized radial basis networks: Application to an autonomous wheelchair, *Int. J. Vehicle Autonomous Systems*, vol.4, no.2-4, pp.239-249, 2006.
[26] M. E. Flores and M. B. Milam, Trajectory generation for differentially flat systems via NURBS basis functions with obstacle avoidance, *Proc. of American Control Conf.*, Minneapolis, Minnesota, pp.5769-5775, 2006.
[27] Y. Li, C. Li and R. Song, A new hybrid algorithm of dynamic obstacle avoidance based on dynamic rolling planning and RBFNN, *Proc. of IEEE Int. Conf. Robotics and Biomimetics*, Sanya, China, pp.2064-2068, 2007.
[28] M. Ghatee and A. Mohades, Motion planning in order to optimize the length and clearance applying a Hopfield neural network, *Expert Systems with Applications*, vol.36, pp.4688-4695, 2009.
[29] E. Masehian and M. R. Amin-Naseri, A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning, *J. Robotic Systems*, vol.21, no.6, pp.275-300, 2004.
[30] P. Bhattacharya and M. L. Gavrilova, Roadmap-based path planning – Using the Voronoi diagram for a clearance-based shortest path, *IEEE Robotics & Automation Magazine*, vol.15, no.2, pp.58-66, 2008.
[31] J. Miura, Support vector path planning, *Proc. of IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Beijing, China, pp.2894-2899, 2006.
[32] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, New York, 2006.
[33] L. Jiang, M. Deng and A. Inoue, Obstacle avoidance and motion control of a two wheeled mobile robot using SVR technique, *International Journal of Innovative Computing, Information and Control*, vol.5, no.2, pp.253-262, 2009.
[34] L.-L. Wang and W.-H. Tsai, Collision avoidance by a modified least-mean-square-error classification scheme for indoor autonomous land vehicle navigation, *J. Robotics Systems*, vol.8, pp.677-698, 1991.
[35] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, Berlin, Germany, 1995.
[36] B. Schölkopf and A. J. Smola, *Learning with Kernels*, The MIT Press, Cambridge, MA, 2002.
[37] P. Bartlett, B. Schölkopf, D. Schuurmans and A. Smola, *Advances in Large Margin Classifiers*, The MIT Press, Cambridge, MA, 2000.
[38] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, The MIT Press, Cambridge, MA, 2004.
[39] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines*, 3rd Edition, http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2010.
[40] R. H. Bartels, J. C. Beatty and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*, Morgan Kaufmann, San Francisco, CA, 1998.
[41] C.-Y. Yang, J.-S. Yang and J.-J. Wang, Margin calibration in SVM class-imbalanced learning, *Neurocomputing*, vol.73, pp.397-411, 2009.
[42] M. H. Wright, The interior-point revolution in optimization: History, recent developments, and lasting consequences, *Bulletin of the American Mathematical Society*, vol.42, pp.39-56, 2005.
[43] S. S. Keerthi, O. Chapelle and D. DeCoste, Building support vector machines with reduced classifier complexity, *J. Machine Learning Research*, vol.7, pp.1493-1515, 2006.