Fast Medial-Axis Approximation via Max-Margin Pushing

Guilin Liu and Jyh-Ming Lien

Abstract—Maintaining clearance, or distance from obstacles and sampling efficient enough configurations on the medial axes are a vital component for successful motion planning. Maintaining high clearance often creates safer paths for robots. Having bias for sampling on medial axis also offers higher possibility to find a path in complex environment where the feasible configuration space only occupies a small proportion of the whole space. Inspired by the similarity between medial axis and max-margin scheme in optimization, especially in Support Vector Machine, we propose a new method to quickly construct the medial axis for the motion planning environment both in low and high dimensional space. However, directly applying the SVM classification on the large volume of uniformly sampled configurations suffers from huge computation and the medial axis is usually not the real medial axis due to SVM’s optimization function’s tolerance to the mis-classification. Instead, we show a method that can quickly push any configuration to the medial axis by using the characteristics of the Max-Margin’s optimization function. Experiments in low and high dimensional space and comparisons with other medial-axis motion planning algorithm are shown.

I. INTRODUCTION

Medial axis (MA), a set of points that locally maximize the distance to obstacles, is known to be an important structure in robotic motion planning [1]. However, computing the exact representation of the MA in configuration space whose dimensionality is usually greater than three is highly nontrivial [2] and can only be approximated.

One type of approximation is sampling. Wilmarth et al. [3] investigated a method, called Medial Axis Probabilistic Road Map (MAPRM), which retracts configurations, free of collision or not, to the medial axis of free configuration space. Because configurations in collision with obstacles can also be pushed to the medial axis, MAPRM provably increases the probability of generating samples in the narrow passage. Lien et al. [4] extended MAPRM to high dimensional spaces by approximating the clearance and penetration depth with a large number of random query rays. Both methods do not provide any guarantee regarding the distribution of samples on the MA. Recently, Yeh et al. [5] presented a method named Uniform Medial-Axis PRM (UMAPRM) that can uniformly samples the MA, thus provide better coverage than MAPRM.

Although these methods effectively addressed the issue of the “narrow passage problem,” they all rely on expensive geometric computation to approximate clearance and penetration depth, particularly in high dimensional space. These expensive proximity queries hinders further improvement of these MA-based sampling strategy. For example, it is highly desirable that the edges connecting the sample can also be on or near the MA to increase both roadmap connectivity and path clearance. However, the edges by these MAPRM-variants are usually not on the MA because making a connection on the MA between a pair of configurations requires a local planner that retracts every discretization of the connection to the MA. This local planner is prohibitively slow for most motion planning problems.

Our Work. This work proposes a solution that allows a configuration, free of collision or not, to be efficiently retracted to the MA. The proposed method achieves this without excessively expensive geometric proximity queries. The main idea is based on an observation that constructing the MA from a set of obstacles can be viewed as a multi-class training and classification problem. Each obstacle is regarded as a label and the labels of samples in the configuration space (C-space) is determined according to which obstacle is the closest. Under this setting, the MA of the free C-space can be viewed as the classification boundary. Therefore, approximating the MA can be transformed to finding the hyperplane that has the largest distance to the nearest training data point of any-class (known as the functional margin).

Fig. 1 illustrates the main steps of the proposed method called SVMA. In machine learning, the widely-used Support Vector Machine (SVM) is a technique that achieves the binary classifier by constructing a hyperplane that has maximum margin. The proposed method SVMA uses the connection between SVM and the medial axis to accelerate the retraction process (discussed in Section III).

The basic SVM only supports binary classification. Extensions to multiple-classes cases are required to handle motion planning problems with more than two obstacles. In Section III-E, we will discuss an efficient multi-class classification approach by combining both the pairwise one-vs.-one SVM and one-vs.-others SVM. As shown in Fig. 1, SVMA has two stages: the training stage and the pushing (retraction)
stage. In the **training stage**, we train two types of classifiers: pairwise one-vs.-one SVM and one-vs.-others SVM. Given \( n \) obstacles, we would have \( n/2 \) one-vs.-other SVM classifiers and \( n(n-1)/2 \) one-vs.-one classifiers. In the **pushing stage**, we first randomly sample a configuration in the environment, and one-vs.-other SVM classifiers are used to determine the closest obstacles, \( m \) and \( n \), for this configuration. Then SVMA uses the pairwise one-vs.-one SVM classifier for \( m \) and \( n \) to push the configuration towards the medial axis. The last two steps are repeated until convergence. Details about training and pushing will be discussed in Section III-E. Our experimental results show that the proposed method provides a fast way to generate dense configurations on the medial axis with bounded approximation error controlled by a user parameter.

**II. RELATED WORK**

**Motion Planning with the Medial Axis** Techniques to construct the medial axis (MA) have been widely explored, such as in graphics and robotics. In this section, we briefly review the use of MA in motion planning literature. Much research has been proposed for motion planning with the workspace medial axis. We are not aware of any motion planning research using the medial axis in arbitrary C-space. Surveys [6], [1] summarize earlier work in this area. Here we will concentrate on more recent work using probabilistic roadmap methods (PRMS).

Foskey et al. [7] propose a hybrid approach which uses a discrete approximation of the generalized Voronoi Diagram of the workspace to find an estimated path. Then they use a randomized path planner to bridge invalid segments along the path.

Hoff et al. [8] present a motion planning method that employs the approximate medial axis computed by graphics hardware. They successfully apply this technique to a 2D dynamic environment composed of more than 140,000 polygons. In [9], Holleman and Kavraki approximate the workspace medial axis by employing a similar method to [10].

Choset and Burdick [11] use the generalized Voronoi graph (GVG) to serve as a basis for sensor based robot motion planning. The GVG is an extension of the medial axis. The medial axis is the set of all points that are equidistant to at least two obstacles. The GVG is the set of all points in \( m \) dimensions that are equidistant to \( m \) obstacles. The GVG is not guaranteed to be connected in dimensions greater than two, so additional structures are introduced to link the disconnected components of the GVG. The resulting structure is the hierarchical generalized Voronoi graph (HGVG). The HGVG is proven to be connected, and can be built incrementally using distance measurements from sensors.

The Equidistance Diagram (EDD) [12] is a roadmap method based on the medial axis of the workspace. Roadmap nodes are the local maxima of a clearance function defined in the workspace.

**III. MAX-MARGIN AND MEDIAL AXIS**

The medial axis of the free space partitions the C-space into Voronoi-cell-like regions. Each region corresponds to a connected component of C-obstacle. All configurations inside that region must be closer to its associated obstacle than to the other obstacles. If we view this property as a classification problem, we can also say that the label of these configurations is their corresponding obstacle. Thus, the MA is where a configuration has similar chance of being labeled from two or more obstacles. Based on this observation, the problem of approximating the medial axis can be transformed to a problem of finding the classification boundary for the labeling (classification) problem.

In this section, we will describe a method called SVMA that uses the characteristics of the max-margin optimization function to push a configuration onto the classification boundary. This is achieved by using the derivative of the SVM Max-Margin classification function as the pushing direction and the classification score as the step size.

**A. Max margin and separating hyperplane**

Support Vector Machine (SVM) finds a linear hyperplane to separate positive and negative data in feature (kernel) space. The hyperplane is parameterized by \( w \) and \( b \), where \( w \) is the normal vector and \( b \) is the offset of the hyperplane. The
distance from the closest point to the hyperplane in feature (kernel) space is called “margin”. SVM, as shown in Fig 2 minimizes $||w||^2$, subject to the constraint:

$$y_i(w^T x_i + b) - 1 >= 0,$$

where $y_i \in \{-1, 1\}$ indicates the label of $x_i$. If the kernel function is used, the $f(x)$ becomes $f(x) = \sum \alpha_i y_i K < x_i, x > + b$ ($K$ is the kernel function). Fig 2 shows the visualization of the linear separating hyperplane for the separable case.

The testing data on the separating hyperplane, corresponding to $f(x) = w^T x + b = 0$, would have equal chance to be assigned with positive label and negative label. In other words, the testing data has the equal “classification distance” to positive support vectors and negative support vectors. Thus, with the similar spirit, the separating hyperplane can be regarded as the “medial axis” between the positive support vectors and negative support vectors. If the positive training data and negative training data are from two geometric objects, the separating hyperplane can be regarded as the approximate medial axis. In motion planning, the positive and negative training data can come from the contact configurations of two obstacles.

### B. Modeling the medial axis

The approximate medial axis (i.e., separating hyperplane) corresponds to the set of $\{x\}$ where $f(x) = w^T x + b = 0$. To obtain a collection of medial axis configurations $\{x\}$, directly solving $x = f^{-1}(0)$ (the inverse function of $f(x) = 0$) is impossible when some more complex kernel function like RBF is introduced.

Another approach is to partition the whole space into uniform grids and determine the labels for each grid cell. Then, the grid cells which have neighboring grid cells with different labels are retained and the rest grid cells are filtered out. The remaining cells must enclose the MA. Unfortunately, this approach requires to test many samples whose size is exponential to the robot’s degrees of freedom, thus impractical for higher dimensional space. In the next section, we introduce an efficient pushing-based approach to approximate the medial axis.

### C. Push configurations to the Medial Axis

Pushing a configuration $x_0$ to medial axis is inspired by the intrinsic idea of maximal margin. The pushing strategy is similar to gradient decent algorithm, but with the step size automatically determined. Given the SVM classification $f(x)$, the pushing is achieved by iteratively updating $x_{n+1} = x_n - \frac{f(x)}{|f(x)|}$ until $|f(x_{n+1})| < \epsilon$. In this paper, we use $\epsilon = 0.02$ throughout all experiments. Configurations generated in this way are approximately on the medial axis. From the view point of SVM classification, the approximation error of these configurations are bounded by this parameter $\epsilon$. Since the classification score field is continuous, given enough samples of such kind, their connecting edges would be on the approximated medial axis too.

We proved the correctness of this gradient-based approach from the perspectives of the intrinsic idea of SVM and from the perspective of gradient decent based optimization. Details of the proofs are shown in the supplemental material.

### D. Handle workspace with more than two obstacles

The SVM is a binary classifier. When there are more than two obstacles in the environment, we are going to discuss the limitations in using either one-vs.-one classifier or one-vs.-others classifier in this section.

**one-vs.-one classifier** Let $x$ be a set of configurations on the separating hyperplane between object $A$ and object $B$. We observe that some configurations in $x$ may be not on the medial axis. For example, some points on the separating hyperplane between $A$ and $B$ can be inside a third object $C$ as shown in Fig. 3.

**one-vs.-others classifier** If we extract the medial axis by find the points where $|g| \approx 0$, for a one-vs.-others classifier $g$. The extracted medial axis underestimates the region that should ‘belong’ to this obstacle. Take Fig 4 as an example. The score field of the one-vs.-others classifier is shown in the left two and top-right images of Fig 4. The bottom right image is the medial axis extracted using one-vs.-others classifier for the bottom obstacle. It can be seen that the medial axis for the one-vs.-others classifiers is too close to the classifier.
Combining one-vs.-one and one-vs.-others classifiers.

From the discussions above, we know that both only using one-vs.-others classifier and one-vs.-one classifier do not extract the correct medial axis. To solve the problem, we need to find the answers for these questions: for a random configuration or a random region in the space, we need to know which obstacles we are closer to, which part of the medial axis we should push to and which classifier we need to use for pushing. And our solution to these question is to combine one-vs.-one classifier and one-vs.-others classifier.

We use one-vs.-others classifiers to identify which obstacles we are closer to and whose one-vs.-one classifier we should use for pushing; then we use the identified one-vs.-one classifier for pushing. To validate this solution, we visualize score field of this combining solution in Fig 5. We first use one-vs.-other classifiers to get the right one-vs.-one classifier \( f_i(x) \) and then compute and visualize the value \( |f_i(x_0)| \) for the current configuration \( x_0 \). The general procedure for generating configurations on medial axis is: a. for a configuration \( x_0 \), we find the "closest" 2 obstacles using all the one-vs.others classifiers’ classification score as the measurement; higher classification score means closer. b. get the one-vs.-one classifier corresponding to those two "closest" obstacles, and apply the pushing strategy discussed in subsection III-C.

E. SVMA Push Algorithm

In this section, we summarize the whole algorithm of SVMA pushing. Given a random configuration \( x \), the pushing is an iterative process. For each iteration, we use all one-vs.-others classifiers to get two obstacles \( s, t \) with the two top classification scores; then use the one-vs.-one classifier \( f_{st}(x) \) to push the two obstacles to get pushing direction \( f'_{st}(x) \) and pushing step size \( f_{st}(x) \); repeat the pushing until the step size is slower than the threshold. The overall procedure can be described as Algorithm 1:

\[
\text{while } d > \epsilon \text{ do } \\
\quad x = x - \frac{f'_{st}(x)}{|f'_{st}(x)|} \cdot d \\
\quad \text{find } s \text{ and } t \text{ such that } g_s \geq g_t \geq g_i, \forall i \in \{1..n\} \setminus \{s, t\} \\
\text{step size } d = f_{st}(x) \\
\text{end}
\]

IV. STEP SIZE OPTIMIZATION

We have discussed using the SVM score as pushing step size. In this section, we will discuss its limitation and alternative approaches that use workspace clearance to obtain better step size.

The training process of SVM to minimize the overall optimization function. So if the positive and negative sample are not linearly separable, some may be misclassified as compromise. As a consequence, the classification boundary is not the real medial axis, If we directly use the SVM’s score as the step size and stopping criteria for pushing, some of the pushing result is not on the medial axis. For example, Fig. 7(a) is the constructed medial axis with SVM classification score as the step size. However, part of the medial axis goes through the obstacles.

A. Workspace clearance difference as step size

The distance measurement and the SVM score are in different domains and have different units. In order to utilize the distance difference for the step size, first we need to find the relationship between the distance and SVM score.
Support vectors are used for computing it. The procedure is below: 1. for each positive support vector \( p \), we find the nearest negative support vector \( q \). 2. place the robot on that support vector (the support vector is a configuration); compute the distance \( d_{pq} \) when the robot is placed on these two configurations. 3. repeat 1 and 2 for all the positive support vectors. The relation between score difference and distance is 

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{(f(p_i) - f(q_i))}{d_{pq_i}}.
\]

To determine the pushing step size for any configuration, shown as Fig 7(b), we place the robot onto that configuration; then we use PQP [18] to get the point that is the closest to the robot for those two obstacles; then get the perpendicular bisector \( l \) of these two points; later compute the distance \( b \) between the robot center and \( l \); the pushing step size is 

\[
b \frac{1}{n} \sum_{i=1}^{n} \frac{(f(p_i) - f(q_i))}{d_{pq_i}}.
\]

And in the same way, the pushing stopping criteria is when the step size is smaller than a threshold \( \epsilon \).

### B. Combining SVM score and clearance difference

We also provide another option that is to use the SVM score as the step size to roughly push the configuration to the medial axis and then change the way to determine the step size by using clearance difference (PQP distance difference). That is to combine the SVM score and clearance difference for pushing. However, we observed that sometimes the improvement of combination of two over only using clearance is insignificant. One reason is that for some very complex environment, SVM classification would have a lot of misclassification. Thus, for the combination approaching, the initial pushing results would be farther from the real medial axis.

### V. Experiments and results

In order to validate our method, we implemented SVMA in C++. All experimental results are collected from a Linux Virtual Machine with 4G RAM hosted on Windows 7 with Intel Core i7-2640M CPU 2.8G. The SVM implementation is adapted from libsvm [15].

#### A. Training data

The training data are obtained from two sources: 1. the configurations from the contact space, and 2. the configurations that are in-collision with the obstacle. We used Minkowski-sum to generate configurations on contact space. Generating configurations on contact-space is more time-consuming compared to doing it in collision space. Intuitively, configurations on contact space would be better for max-margin. In 2D and 3D, generating the sample configurations on the contact-space does not cost too much time comparing to training the SVM models. Thus, we used the configurations in contact space as our training data. For higher dimension, we can use either contact space or obstacle space.

#### B. Experiment Setting

As shown in Fig. 8, we tested four environments with 2D, 3D, 6D C-space. The first environment contains 354 3D buildings from Oklahoma City. In our 2D experiment, we use their 2D footprints on the ground. We also use these models to construct the medial axis in 3D. Another 3D model is a skeleton model of a house, which is made up of 148 cylinders. The environment with 6D has a robot made up of 3 parts and can translate and rotate, enabling the total degree of freedom to be 6D. Figure 8 shows the built medial axis configurations for those environments. For the 2D and 3D examples, our goal is to have enough samples that can approximate the medial axis of all the obstacles in the space. The robot for these two environment is a point robot. However, for the 6D environment, our goal is to find the path for the robot through the narrow passage. The robot’s geometry contains three parts that are perpendicular to each other and needs to rotate when it goes through the narrow passage. From the Table I, we found that the main time cost are computing Minkowski-sum in 6D and connecting the configurations. While for this environment, two obstacles are separable, thus, training the SVM is very fast.

1) **Comparison with MAPRM:** And another observation is that the medial axis from MAPRM is generated with the help of approximation using rays. The pushed result is less accurate than the pushing result of our method. In other words, our medial axis configurations are more concentrated. In this way, the length of the connecting edges would be shorter too. Thus, less interpolation is needed to check whether this edge is valid or not. As a result, the connecting time of our method is also shorter than the MAPRM. And pushing using our method is faster than MAPRM.

#### C. Step Size Strategy

As mentioned above, the constructed medial axis using clearance difference as step size has better quality than the SVM score. We will discussing the running time difference.

We set the maximum pushing steps for each method to be the same. However, 80% to 90% of the configurations can be pushed to the MA in 10 steps. The running time is usually related to the density of obstacles and the SVM model’s complexity. The former will influence the speed of collision checking and computing clearance difference, while the
latter relates to the time cost for classifying a configuration. Generally, using SVM as the step size is faster than using clearance difference as step size, because running one time of the classification is often faster than doing one time collision detection. This is also one advantage of our method, especially when the geometry structure of the obstacle is complex. However, when the geometry of the obstacle is simple and only approximated by few vertices, it is not an advantage any more. For example the obstacles in 2D and 3D examples in Fig 8, the obstacles are simple geometry object. On the other hand, for 2D examples, the obstacles have very different scales and some of them are very close at their concave parts. This makes the SVM models more complex. Thus, the time-cost difference between using SVM score as step size and using clearance difference as step size is very small. The same to the combination of them. While in 6D examples, two obstacles doesn’t touch each other. As a result, the learned SVM model is less complex. Using SVM score as step size is faster than using clearance difference as step size. The combination of them is between them.

VI. Conclusion

In summary, we presented a method that can generate approximate medial axis method based on SVM model. The building processing is based on pushing, where the pushing direction and step size can be automatically determined. We showed the experiments in 2D, 3D and 6D.

REFERENCES


**TABLE I**

**COMPARISON WITH MAPRM**

<table>
<thead>
<tr>
<th>Env</th>
<th>ObN</th>
<th>Method</th>
<th>Number MKSum Num Train Num</th>
<th>Pushing Num</th>
<th>Time(s) MKSum Train Push</th>
<th>Conn Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>354</td>
<td>MAPRM</td>
<td>70800</td>
<td>70800</td>
<td>40000</td>
<td>0.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVM</td>
<td>70800</td>
<td>70800</td>
<td>40000</td>
<td>0.33</td>
</tr>
<tr>
<td>3D</td>
<td>148</td>
<td>MAPRM</td>
<td>69469</td>
<td>59200</td>
<td>20000</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVM</td>
<td>69469</td>
<td>59200</td>
<td>20000</td>
<td>0.59</td>
</tr>
<tr>
<td>6D</td>
<td>2</td>
<td>MAPRM</td>
<td>27478</td>
<td>3000</td>
<td>20000</td>
<td>3.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVM</td>
<td>27478</td>
<td>3000</td>
<td>20000</td>
<td>3.23</td>
</tr>
</tbody>
</table>

**TABLE II**

**THE RUNNING TIME COMPARISONS FOR STEP SIZE USING SVM SCORE, CLEARANCE DIFFERENCE AND THEIR COMBINATION. THE COMBINATION APPROACH WOULD HAVE 30% STEPS USING SVM SCORE AND THE REST 70% USING CLEARANCE DIFFERENCE.**

<table>
<thead>
<tr>
<th>Env</th>
<th>ObN</th>
<th>Sample N</th>
<th>SVM</th>
<th>Clearance</th>
<th>Combine</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>354</td>
<td>10000</td>
<td>SVM</td>
<td>94.09</td>
<td>100.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32.27</td>
<td>101.03</td>
</tr>
<tr>
<td>3D</td>
<td>148</td>
<td>10000</td>
<td>SVM</td>
<td>56.84</td>
<td>49.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.03</td>
<td>52.08</td>
</tr>
<tr>
<td>6D</td>
<td>2</td>
<td>10000</td>
<td>SVM</td>
<td>20.74</td>
<td>29.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.15</td>
<td>22.01</td>
</tr>
</tbody>
</table>