Local Path Planning for an Unmanned Ground Vehicle Based on SVM

Regular Paper

Chen Qingyang1,*, Sun Zhenping1, Liu Daxue1, Fang Yuqiang1 and Li Xiaohui1

1 College of Mechatronics and Automation, National University of Defense Technology, Changsha, P.R. China
* Corresponding author E-mail: chy1982_008@nudt.edu.cn

Received 22 Jun 2012; Accepted 5 Oct 2012

DOI: 10.5772/54130

© 2012 Qingyang et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract To resolve the local path generating problem for unmanned ground vehicles (UGV) in unstructured environments, a method combining a basic path subdivision method for topological maps of local environments and a Support Vector Machine (SVM) is proposed in this paper. Based on the basic path subdivision method, topological maps of local environments can be extracted with little expanded nodes, without the constraints of obstacle representation, so meeting the need for autonomous navigation in unstructured environments. Next, to optimize the candidate routes in topological maps and generate a smoother path, an SVM is introduced. The candidate routes boundary points are defined as positive and negative samples, and SVMs are employed to train the separating surface. An original SVM is extended to satisfy extra constraints such as vehicle position and heading constraints. Experimental results show the effectiveness and advantages of the proposed method.

Keywords Unmanned ground vehicle, local path planning, SVM, basic path subdivision

1. Introduction

Unmanned ground vehicles are becoming more and more important in human environments, especially in the military applications such as military reconnaissance, transportation and soldier rescue. In these tasks, an unmanned ground vehicle is usually required to follow a predefined path, avoiding unexpected obstacles. However, with the limitation of sensors on vehicles, perception of environments cannot be very accurate and stable all the time, and positioning errors may sometimes be large. What is more, typical path following methods usually cannot guarantee a vehicle tracking a predefined path precisely due to uncertainty of vehicle dynamics, etc.

To adapt to the above situations, local path planning should be carried out continuously and react to uncertainties in a timely fashion. In addition, to improve the performance of path following, vehicle kinematics and dynamics should be taken into account in the local path planning process. This may make the problem difficult to resolve with the real-time requirement.

To address the complicated problem of local path generation, a number of researchers have joined this group and some significant results have been achieved. Roughly speaking, traditional path planning methods can be divided into three groups [1], namely cell decomposition methods, roadmap methods [2-6] and artificial potential methods [7-10]. These methods can resolve the path planning problem for some applications.
Cell decomposition methods rely on the partition of the configuration space into a finite number of regions. Each cell is denoted as collision free or covered by obstacles. The local path planning problem is then translated into the problem of finding a sequence of collision free neighbouring cells. When the initial and local target states are contained, a feasible path is found.

In roadmap methods, a network of collision free paths is constructed, which spans the free configuration space (i.e., the subset of the configuration space that does not result in collisions with obstacles). When initial and target states are given, the first step is to find paths connecting the initial and target configurations to the roadmap. Based on that, a sequence of paths on the roadmap is chosen. There are several methods for building such a roadmap, among which we can mention visibility graphs and Voronoi diagrams.

In artificial potential field methods, a collision free trajectory is generated by a robot moving locally according to “forces” defined as the negative gradient of a potential function. This function is designed to provide attractive forces toward the goal and repulsive forces which push a robot away from obstacles (the potential function is bowl-shaped with the goal at the bottom, and obstacles are represented by peaks). This class of methods is based on the definition of a feedback control policy (i.e., the control is computed at each instant in time as a function of the current state), as opposed to the open-loop approaches of the preceding two classes. A shortcoming of this formulation is the possible existence of local minima in which a robot might become trapped. An artificial potential function, which does not have local minima, is called a navigation function. But computing such a function is generally as difficult as solving the local path planning problem for all initial states.

Another group of path planning methods, such as the probabilistic roadmap (PRM) [11, 12] and rapidly exploring random trees (RRTs) [13-15], aim to use random sample methods to resolve the path planning problem. These methods are implemented by sampling the state space randomly and using feasible trajectory segments to connect these states until the expected target state has been reached. However, the real-time requirement cannot be guaranteed for these methods, which is important for autonomous navigation of vehicles. To make the sampling-based method useful for motion planning of unmanned ground vehicles, an extension of the standard RRT, combining the close-loop prediction control method and several important details was proposed in [16]. The proposed method was applied to Team MIT’s entry in the 2007 DARPA Urban Challenge - their vehicle being one of the six vehicles that completed all missions.

Motion planning with primitives [17] is another popular method developed recently. Mobility constraints and dynamic requirements can be embodied in the motion primitives. With motion primitives, deterministic [18] or randomized [1] planning methods can find a feasible trajectory for robots, which satisfy the dynamical requirements and environmental constraints. Key problems for a local path planning method based on motion primitives may be the generation of a primitive library [19, 20] and on-line search efficiency [21, 22].

In [23], the SVM method is introduced to mobile robot path planning. Obstacles in an environment are labelled as positive and negative samples, and the path planning problem is transformed to the dual problem of finding a smooth separating surface for the two classes. To adapt to dynamic environments, the work is extended in [24, 25]. New obstacles detected on-line are classified using the k-nearest neighbours algorithm in [25]. Based on the classification result, the kernel-based SVM is used to find a smooth path for mobile robots. However, some limitations exist in these methods. Firstly, to guarantee the intermediate region, including the start and goal points, some virtual obstacles are placed around these points. However, this cannot guarantee the separating hyperplane passing through the start and goal points, or the planned path connecting the start and goal points. It also cannot guarantee that the planned path being coincident with the direction of a mobile robot. This is not good when a robot needs to follow the planned path precisely. Second, when searching for a path based on the training process of the SVM, the methods above all use obstacle samples. This may waste much time for kernel matrix computation and may be unnecessary. In fact, only the boundary points of a special route are needed for the training process.

Based on the analysis above and to resolve the local path generating problem, this paper designed a basic path subdivision method for topological maps. When the topological map of a local environment is extracted, route boundary points can be computed and can be used for the training process of the SVM. This can save a lot of time, so meeting the real-time needs of a UGV. Moreover, to guarantee the separating hyperplane, or the planned path connecting the start and goal points, the original SVM was extended and extra constraints incorporated. So the Lagrange function will be different to that in the original SVM. Details of the extended SVM will be given below.

The remainder of this paper is organized as follows: firstly, the original SVM will be extended in Section 2, so making sure the planned path connects the start and goal points. To improve the time performance of the SVM-based method, a basic path subdivision method for topological maps is introduced in Section 3. Based on the
basic path subdivision method, topological maps of local environments can be extracted with little expanded nodes. When the topological map of a local environment is extracted, sample data for the training process of SVM can be resolved, even with irregular obstacles. Experimental results of the proposed method will be given in Section 4, followed by the conclusions of this paper and some future work in Section 5.

2. Extended SVM for local path planning

2.1 Local path planning for UGV

As a key problem for autonomous navigation of UGVs, local path planning is attracting more and more attention. The aim of local path planning is to find a feasible path from a start point to a goal point in a local environment. The path should not intersect with any obstacles. In addition, to improve the tracking performance, vehicle kinematics, or even dynamics, are suggested to be taken into consideration. The typical situation of local path planning for a UGV is demonstrated in Figure 1. In the figure, the black area denotes the feasible zone, while the white area represents the obstacles. The green circle at the bottom of the figure denotes the position of a vehicle and the green arrow denotes its heading. The blue circle at the top of the figure represents the expected position of the vehicle and the blue arrow is its expected heading. Local path planning in this environment is to find a feasible path from the point V to point T, which may as well be coincident with the start and goal directions, curvatures and so on. The obstacles are expanded with the size of the vehicle so the vehicle can be considered as a single point with a special heading.

Figure 1. Demonstration of local path planning for a UGV in an unstructured environment. The green circle and arrow at the bottom of the figure denote the start state of a vehicle. The blue circle and arrow at the top of the figure denote the expected state. The black area denotes the feasible zone, while the white areas are obstacles in the local environment.

2.2 Support vector machine

2.2.1 Optimal separating hyperplane

As a binary classification method, the SVM method is used to find the optimal separating hyperplane based on the concept of margin maximization [26].

Let \( (x_i, y_i), \ldots, (x_N, y_N), \quad x_i \in \mathbb{R}^m, \quad y_i \in [-1,1] \) be the training samples. The binary classification problem is to find a hyperplane \( w^T x + b = 0 \) separating the samples. If the training samples are linearly separable, there are parameters \( w \) and \( b \) that satisfy [23]:

\[
y_i(w^T x_i + b) \geq 1, \quad (i = 1, \ldots, N)
\]

Using such parameters, the two classes are separated by two hyperplanes:

\[
\begin{align*}
H_1: & \quad w^T x + b = 1 \\
H_2: & \quad w^T x + b = -1
\end{align*}
\]

and no data exist between the hyperplanes. Since the distance between the two hyperplanes is \( \frac{2}{\|w\|} \), the SVM method is used to find the optimal parameters that maximize the distance between the hyperplanes or that maximize the margin between the hyperplanes. Optimal parameters are determined by minimizing the objective function:

\[
\Phi(w, b) = \|w\|^2 / 2
\]

under the constraint represented by (1).

So the original SVM problem is a quadratic programming problem formulated as:

\[
\begin{align*}
\min & \quad \Phi(w, b) = \|w\|^2 / 2 \\
\text{s. t.} & \quad y_i(w^T x_i + b) \geq 1, \quad (i = 1, \ldots, N)
\end{align*}
\]

Resolving the quadratic programming problem by Lagrange function:

\[
L(w, b; \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{N} \alpha_i[y_i((w \cdot x_i) + b) - 1]
\]

where \( \alpha_i \geq 0 \) \( (i = 1, \ldots, N) \) are the Lagrange multipliers. A solution to this problem is given by solving the following dual problem:

\[
\max W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)
\]

s. t.
\[ \sum_{i=1}^{N} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \quad (i = 1, \ldots, N) \quad (8) \]

In the above, \((x_i, x_j)\) are the inner products between samples \(x_i\) and \(x_j\).

The relation between the optimal parameters \((w^*, b^*)\) for the original problem and \(\alpha^*\) for the dual problem can be represented as:

\[ w^* = \sum_{i=1}^{N} y_i \alpha_i^* x_i \quad (9) \]

\[ b^* = \frac{1}{2} [w^* \cdot x^* - w^* \cdot x^+] \quad (10) \]

where \(x^*\) is a random support vector from one class and \(x^+\) is a random support vector from the other class.

According to the Karush-Kuhn-Tucker (KKT) condition, only some \(\alpha_i\) with special training data \(x_i\) are non-zero in (11):

\[ \alpha_i [y_i ((w \cdot x_i + b) - 1)] = 0 \quad (i = 1, \ldots, N) \quad (11) \]

These special training data are on either one of the hyperplanes \(H_1\) and \(H_2\). In the SVM, these training samples are called support vectors because they are the only data that determine the parameters. From these sampling data, a discrimination function is determined as:

\[ \sum_{i \in S} \alpha_i^* y_i (x_i \cdot x) + b^* = 0 \quad (12) \]

where \(S\) indicates the set of indices for support vectors.

### 2.2.2 Non-linear SVM using kernel tricks

An SVM can be used to determine non-linear separating surfaces using kernel tricks [26]. This is based on the idea of transferring non-linearly separable data to a higher dimensional space. To calculate the inner products in a higher dimensional space, kernel functions are adopted.

Let \(x_i\) and \(x_j\) be sample data in an original space, and \(\phi\) be the mapping from the original space to a higher dimensional space. To calculate the inner products of \(\phi(x_i), \phi(x_j)\) in the higher dimensional space, some classes of kernel functions \(K\) that satisfy:

\[ K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad (13) \]

are employed. With the introduction of kernel functions, the objective function of (7) is represented as:

\[ \max W(\alpha) = \frac{1}{2} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (14) \]

under the constraints:

\[ \sum_{i=1}^{N} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \quad (i = 1, \ldots, N) \quad (15) \]

Then the discrimination function of (12) is modified as:

\[ \sum_{i \in S} \alpha_i^* y_i K(x_i, x) + b^* = 0 \quad (16) \]

Since all necessary calculations in the higher dimensional space are replaced by the calculation of the kernel function in the original space, we can determine non-linear separating surfaces without explicitly representing the higher dimensional space.

### 2.3 Extended SVM for local path planning

To make use of an SVM for local path planning and thus guarantee the separating surface simply connects the start and goal points, the original SVM is extended in this paper. Details of the extended algorithm will be given below.

Let \(v_t \in \mathbb{R}^m\) be a start point of a vehicle and \(v_f \in \mathbb{R}^m\) be a goal point. As referred to above, \((x_1, y_1), \ldots, (x_N, y_N), x_i \in \mathbb{R}^m, y_i \in \{-1, +1\}\) are the training samples. The extended SVM for local path planning is used to find a hyperplane \(w^T x + b = 0\) separating the samples - at the same time satisfy:

\[ w^T v_t + b = 0 \quad (17) \]

\[ w^T v_f + b = 0 \quad (18) \]

So the extended SVM for local path planning is formulated as a quadratic programming problem:

\[ \Phi(w, b) = \|w\|^2 / 2 \quad (19) \]

s. t.

\[ y_i (w^T x_i + b) \geq 1, \quad (i = 1, \ldots, N) \quad (20) \]

\[ w^T v_t + b = 0 \quad (21) \]

\[ w^T v_f + b = 0 \quad (22) \]

Solving the original problem above by using the Lagrange function:

\[ L(w, b, \alpha, \beta) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i ((w \cdot x_i + b) - 1)] \]

\[ - \sum_{j=1,2} \beta_j (w \cdot v_j + b) \quad (23) \]

In the above, \(v_t\) and \(v_f\) are denoted as \(v_j\) \((j = 1, 2)\), so to represent the problem in a compact form.
\[ \alpha_i \geq 0 \quad (i = 1, \ldots, N), \quad \beta_j \in \Re (j = 1, 2) \] are the Lagrange multipliers.

The dual problem can be formulated as:

\[
\text{max} \quad W(\alpha, \beta) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i (x_i \cdot x_j) \\
- \sum_{i=1}^{N} \sum_{j=1}^{2} \alpha_i \beta_j y_i (x_i \cdot v_j) - \frac{1}{2} \sum_{i,j=1}^{2} \beta_i \beta_j (v_i \cdot v_j) \\
\text{s.t.} \quad \sum_{i=1}^{N} \alpha_i y_i + \sum_{j=1}^{2} \beta_j = 0, \quad \alpha_i \geq 0 (i = 1, \ldots, N)
\] (24)

The relation between the optimal parameters \((w^*, b^*)\) for the original problem and \((\alpha^*, \beta^*)\) for the dual problem can be represented as:

\[
w^* = \sum_{i=1}^{N} y_i \alpha_i x_i + \sum_{j=1}^{2} \beta_j v_j
\] (26)

\[
b^* = \frac{1}{2} [w^* \cdot x^* - w^* \cdot x^-]
\] (27)

As referred to above, \(x^*\) is a random support vector from one class and \(x^-\) is a random support vector from the other class. In addition, according to the Karush-Kuhn-Tucker (KKT) condition, only some \(\alpha_i\) with special training data are non-zero. With these special data, a discrimination function is determined as:

\[
\sum_{i \in S} \alpha_i^* y_i (x_i \cdot x) + \sum_{j=1}^{2} \beta_j^* (v_j \cdot x) + b^* = 0
\] (28)

where \(S\) indicates the set of indices for support vectors.

Obviously, a linear SVM cannot adapt to many kinds of situations for local path planning. Usually a non-linear SVM using kernel tricks is adopted. The separating surface is determined as:

\[
\sum_{i \in S} \alpha_i^* y_i K(x_i, x) + \sum_{j=1}^{2} \beta_j^* K(v_j, x) + b^* = 0
\] (29)

where \(K(\ , \ )\) is the adopted kernel function. Usually used kernel functions may include a radial basis function (RBF) kernel:

\[
K(x_i, x_j) = \frac{1}{\sigma^2} e^{-\frac{|x_i - x_j|^2}{\sigma^2}}
\] (30)

polynomial kernel:

\[
K(x_i, x_j) = [1 + (x_i \cdot x_j)]^d
\] (31)

and B-spline kernel:

\[
K(x_i, x_j) = \prod_{k=1}^{m} B_{2p+1}(x_i - [x_j]^k)
\] (32)

In the B-spline kernel function, \(B_{2p+1}(x)\) is a B-spline function with degree of \(2p + 1\). \([x_j]^k\) is the \(k\) th component of \(x_j\).

Based on the analysis above and the extended SVM, simulation results can be seen in Figure 2. In the figures, the red points denote samples of one class and the blue points belong to the other class. In Figure 2(a), no start point and goal point is required to connect, while in Figure 2(b), the start and goal point constraints are introduced. The green points in Figure 2(b) denote the start and goal points. With the different requirements, the separating surfaces also have some differences. But obviously the planned path in Figure 2(b) is more suitable than the result in Figure 2(a) for local path planning, as it passes through the start and goal points. One point to mention is that the function adopted in the simulation is the RBF kernel. This is widely used in SVMs with kernel tricks.
3 A basic path subdivision method for topological maps

The extended SVM above can find a smooth path connecting the start and goal points when the sampling data are given; however, in unstructured environments, acquiring the sampling data representing the obstacles is not an easy task. What is more, when all obstacles in an unstructured environment are formulated as sampling data, computation of the SVM may waste much time. This can be seen from Figure 1. In the figure, the number of sampling data for obstacles representation may be large and the obstacles are irregular, so the extraction of sampling data may not be an easy task.

To address the above problem, this paper designed a basic path subdivision method for topological maps of local environments. A basic path is generated by the vehicle state constraints. The vehicle state in the generating process for a basic path is defined as \((v_x, v_y, v_\theta)\) in a configuration space, where \((v_x, v_y)\) is the position and \(v_\theta\) is the heading. Details of the basic path subdivision method will be given below. Based on the method, topological maps of local environments can be extracted with little expanded nodes, so we can get the result in an acceptable time. With the topological maps, sampling data for SVM training can also be extracted by finding the boundary points of the topological routes.

3.1 Basic path generating method

In this paper, a basic path is generated by using a geometrical method. This is easy to implement and as it is required to find the topological maps of local environments, it is sufficient.

As shown in Figure 3, let \(XOY\) represent the temporal coordinate of the vehicle, where \(O\) is its position. The heading of the vehicle is along the \(Y\) axis. \(O(x,y,\alpha)\) is a local target state that comes from the global path planning stage. \((x, y)\) is the position of \(O\) and \(\alpha\) is the angle between the heading of \(O\) and the \(X\) axis.

![Figure 3. Local coordinate for the basic path generation.](image)

From geometry we can ascertain a single arc that simultaneously satisfies the initial state \(O(0,0,90)\) and the local target state \(O(x,y,\alpha)\) is impossible in many situations. An asymptotic approaching method will be adopted, so meeting the initial state and the target state constraints. Based on geometry, a single arc that satisfies the initial state \(O(0,0,90)\) and the weaker target state \(O^*(x,y)\) can be realized as (33), (34), (35) and (36).

\[
tg(\beta) = \frac{y}{x} \quad \text{(33)}
\]

\[
r = 0.5 \ast \frac{l_{OO^*}}{\cos(\beta)} \quad \text{(34)}
\]

\[
\theta_s = i \ast \frac{180 - 2 \ast \beta}{N} \quad \text{(35)}
\]

\[
\begin{align*}
x_s &= r \ast (1 - \cos(\theta_s)) \\
y_s &= r \ast \sin(\theta_s)
\end{align*} \quad \text{(36)}
\]

where \(\beta\) is the angle between \(OO^*\) and the \(X\) axis, \(l_{OO^*}\) is the distance between \(O\) and \(O^*\). \(N\) is the expected number of sampled points on the arc, while \(\theta_s\) corresponds to the \(i\) th sampled point. \((x_s,y_s)\) is the position of the \(i\) th sampled point.

From (33), (34), (35) and (36), an expected arc \(\widehat{OO^*}\) from \(O(0,0,90)\) to \(O(x,y)\) can be achieved. Similarly, another expected arc \(\widehat{OO^*}\) from the target state \(O^*(x,y,\alpha)\) to the weaker initial state \(O^*(0,0)\) can be obtained. From the corresponding sampled points on the arcs, an asymptotic approaching method is adopted as (37).

\[
\begin{align*}
x'_s &= e \ast x_s + \eta \ast x'_s \\
y'_s &= e \ast y_s + \eta \ast y'_s
\end{align*} \quad \text{(37)}
\]

where \((x_s, y_s)\) and \((x'_s, y'_s)\) are the corresponding sampled points on the arcs, and \((x''_s, y''_s)\) is the final result from the initial state \(O(0,0,90)\) to the local target state \(O(x,y,\alpha)\). \(e\) and \(\eta\) are the weights corresponding to \(O^*\) and \(O\), which is defined as (38) in this paper.

\[
\begin{align*}
e &= 1 \ast \frac{i - 1}{N - 1} \\
\eta &= \frac{i - 1}{N - 1} \quad (i = 1, 2, \ldots, N)
\end{align*} \quad \text{(38)}
\]

The asymptotic approaching path from the initial state \(O(0,0,90)\) to the local target state \(O(x,y,\alpha)\) is defined as a basic path. A result of the basic path generating method is shown in Figure 4. As shown in Figure 4(a), the arc denoted by blue points is computed by the weaker initial state and the local target state, while the
arc denoted by green points comes from the initial state and the weaker target state. The curve denoted by red points is the asymptotic transferring path from the initial state to the local target state, realized by the method above, and Figure 4(b) shows the curvature of the basic path.

![Arc and Curve](image)

(a) A basic path generated by the vehicle state constraints.

(b) The curvature of the basic path.

**Figure 4.** A simulation result of the basic path generating method. In figure (a), the blue and green points denote the computed arcs, while the red points denote the asymptotic approaching result from the initial state to the local target state. Figure (b) is the curvature of the basic path.

3.2 Basic path subdivision method

Based on the basic path generating method above, this paper designed a basic path subdivision method for topological maps of local environments. One point that should be emphasized is the difference between a route and a path in this paper: a route in this paper denotes a feasible channel from an initial state to a local target state. It does not equal a path, which denotes a curve connecting the initial state and the local target state. The topological map is focusing on all candidate routes in a local environment. When all routes are found and the optimal route is chosen, the optimal path is to be generated in the optimal route.

As shown in Figure 5, with an initial state of the vehicle and a local target state, a basic path can be generated based on the method above. Collision detection is carried out over the generated path and two possible results may appear. A lucky case is that the basic path is collision free. In this case, the process is over. Although in this manner only one route is achieved, and there may be other routes for the vehicle to navigate amongst obstacles and reach the local target, it does not matter seriously. There is very little possibility that any other routes are more optimal than the route from the basic path, with the smoothness criterion of a curve. This can be shown from Figure 4 and Figure 5 below. Another common case to meet is just as shown in Figure 5, where the basic path collides with an obstacle $O_1$. This case can be dealt with using the subdivision method below.

In general, for the points on the basic path colliding with obstacles, some substitutable expanded states can be generated. So by simply searching some expanded states along the direction perpendicular to the heading of the collided point, expanded states can be found - just as the green points 1 and 2 in Figure 5(a). Here the heading of the collided point is computed by the positions of the collided point and the preceding one. Based on the expanded nodes, the basic path $VT$ from the initial state to the local target state is subdivided. $V1$ and $1T$ form one route; $V2$ and $2T$ form the other route. However, as the generated basic path $2T$ still collides with an obstacle $O_2$, it is subdivided again. Based on the iterative subdivision process, a topological map of local environments can be achieved with little expanded nodes, just as shown in Figure 5(a). In addition, from Figure 5(b) we can see the curvature of routes in the topological map may have a sharp turn with the expanded nodes. This is because the basic path subdivision method only considers the position and direction attributes, but without the curvature attribute of the nodes. So the routes with expanded nodes may be less optimal than the basic path, based on the smoothness criterion, as evidenced above.

When using the basic path subdivision method for a complex environment, as shown in Figure 6, the method can still work well. As shown in the figure, many convex obstacles are placed in the local environment. The path subdivision method can obtain main routes in the local environment. Some very narrow passages and curving routes are discarded due to the constraints of the unmanned ground vehicle. Finally, 11 routes are found in the local environment, as shown in Figure 7.
The topological map of a local environment with the basic path subdivision method.

(b) The curvature of routes in the topological map.

Figure 5. An experimental result of the basic path subdivision method. In figure (a), when the basic path $VT$ collides with an obstacle $O_1$, it is subdivided, and $V1T$ and $V2T$ are obtained. The subdivision process is iteratively implemented based on the collision detection results, and candidate routes $V1T$ and $V23T$, $V24T$ can be achieved. Figure (b) illustrates the curvature of routes in the topological map. From figure (b), the curvature of routes with expanded nodes may have sharp turns.

Figure 6. Experimental results of the basic path subdivision method in a complex environment. The green points denote the expanded nodes in the basic path subdivision process.

Figure 7. SVM training results for the complex environment in Figure 6. The red points denote the separating surface or the planned path. The green points on the planned path are the extra constraints of vehicle position and heading. Sampling data for the training process are the green points on the left of the planned path and the blue points on the right of the planned path. The yellow points denote the support vectors. The extended SVM training process is based on the B-spline kernel.
3.3 Extraction of route boundary data

When the topological map of a local environment is achieved, the boundary points of candidate routes can be extracted. Searching along the direction perpendicular to the heading of the routes and sampling data for SVM training can be obtained as shown in Figure 8. In the figure, the red points denote the candidate routes in a local environment. The distance between the sampled points on the routes is about 1m empirically. This is sufficiently reliable for the path generation and may not need much time for SVM training when in local environments for local path planning. The green points are the left boundary points of the routes and the blue points are the right boundary points of the routes. With the boundary data, the extended SVM above can be implemented for local path planning.

![Figure 8. Demonstration of extracting route boundary data for SVM training. The red points denote the candidate routes in a local environment. The green points are the left boundary points of the routes and the blue points are the right boundary points. These boundary data can be employed for SVM training.](image)

4. Experimental results

Based on the extended SVM and basic path subdivision method, experiments were carried out on a modified unmanned ground vehicle as shown in Figure 9. The vehicle is a modified Toyota Land Cruiser. Steering, brake and throttle are all controlled by computers. Sensors for obstacle recognition include a Velodyne HDL-64 lidar, a Riegl LMS-Q1201, three SICK LMS 291 and two SICK LMS 111 lasers. These sensors mainly detect obstacles and road curb in a local environment. Three SONY colour cameras are equipped for ground segmentation and traversability analysis. A NovAtel SPAN-CPT GPS-aided inertial navigation system (INS) is also incorporated for vehicle position and posture estimation.

![Figure 9. Unmanned ground vehicle.](image)

Experiments were carried out in unstructured environments. Typical scenes can be seen in Figure 10. From the figures we can see objects in the environment are irregular or it is difficult to define a single object. So the perception results may be more jumbled than in Figure 1. Experimental results of the extended SVM for Figure 8 can be seen in Figure 11. The function used in the experiment is the RBF kernel. In the figure, the red points denote the separating surface or the planned path by the extended SVM above. The extra constraints are represented by the green points on the separating surface which denote the position and heading constraints. The green points on the left of the planned path are the left boundary points and the blue points on the right of the planned path are the right boundary points. The yellow points in the figure denote the support vectors.

![Figure 10. Typical scene for the experiments.](image)
In addition, when changing the RBF kernel function with other kernel functions, comparative experimental results can be realized in Figure 13. Experimental results in Figure 13(a) and 13(b) are based on the polynomial kernel function, while 13(c) and 13(d) are based on the B-spline kernel function. From these experimental results, the extended SVM can generate rational paths for the situations. But how to choose a rational kernel function for local path planning is a challenging problem for the future.

Next, to compare the proposed extended SVM with the method in [23], experiments based on the method in [23] are carried out in Figure 8. Experimental results can be seen in Figure 14. In the figure, some sampling data are placed on the left (green points) and right (blue points) of the start and goal positions. Experimental results in Figure 14(a) and 14(b) are based on the RBF kernel function, in 14(c) and 14(d) they are based on the polynomial kernel function, while in 14(e) and 14(f) they are based on the B-spline kernel function. From these results we can see the method in [23] cannot guarantee the planned path satisfying the vehicle position and heading constraints well, especially in 14(d) and 14(f). This is not good when trying to achieve a vehicle following the planned path.
Figure 14. Experimental results for Figure 8 with the method in [23]. In the figures, some sampling data are placed on the left (green points) and right (blue points) of the start and goal positions. The results in (a) and (b) are based on the RBF kernel function, in (c) and (d) they are based on the polynomial kernel function, while in (e) and (f) they are based on the B-spline kernel function. From these results we can see the method in [23] cannot guarantee the planned path satisfying the vehicle position and heading constraints well. This is not good for local path planning.

Finally, to test the proposed method for local path planning, different experimental results with the RBF kernel function can be found in Figure 15. From these results we can see the proposed method can generate smooth paths most of the time and adapt well to different situations. When these smooth paths are achieved, the optimal path in local environments can be chosen for path following.

Figure 15. Experimental results of the proposed method in different situations with the RBF kernel function. From these results we can see the method can generate smooth paths most of the time and adapt well to different situations.

5. Conclusions and future work

This paper proposed a local path planning method based on an SVM for unmanned ground vehicles in unstructured environments. The original SVM for classification problems is extended, so to satisfy extra constraints such as vehicle position and heading constraints. Based on the extended SVM, smooth paths connecting the start and goal points can be generated.
In addition, to satisfy the real-time need for autonomous navigation, a basic path subdivision method is designed. The method subdivided the basic path connecting the start and goal states based on collision detection results. When no collision is encountered, the process is over, so adapting to different kinds of situations automatically. Topological maps of local environments can be extracted with little expanded nodes. When the topological map is extracted, the route boundary points can also be extracted, so providing pre-requisite sampling data for the SVM training process.

Based on the proposed method, experiments were carried out on a modified unmanned ground vehicle. Experimental results illustrate the effectiveness and advantages of the proposed method.

Even though the proposed method can generate smooth paths most of the time, some work remains to be done. For example, how to choose a rational kernel function for local path planning, and how to incorporate the vehicle dynamics into consideration, are important problems to be researched. It should also be noted that this paper mainly discusses the local path planning problem in static environments. Therefore, when extended to dynamic environments, some work ought to be incorporated. For example, predicting the future state of dynamic obstacles with the sensed speed and acceleration should be resolved. Finally, in an environment with non-convex obstacles, the basic path subdivision method may have some problems and this is to be researched in future work.

6. Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under grant no. 90820302. The authors would like to thank all persons in the laboratory for their assistance during the data collection. Finally, we would like to thank anonymous reviewers for their constructive comments and suggestions that improved the paper.

7. References


