PATH PLANNING USING CONSTRAINED PATH

Jalil Rasekh

Department of Biomedical Engineering, Noshirvani University of Technology, Babol, Iran
jalil.rasekhi@gmail.com

ABSTRACT

In this paper, we propose a new method for path planning to a point for robot in environment with obstacles. The resulting algorithm is implemented as a simple variation of Dijkstra’s algorithm. By adding a constraint to the shortest-path, the algorithm is able to exclude all the paths between two points that violate the constraint. This algorithm provides the robot the possibility to move from the initial position to the final position (target) when we have enough samples in the domain. In this case the robot follows a smooth path that does not fall in to the obstacles. Our method is simpler than the previous proposals in the literature and performs comparably to the best methods, both on simulated and some real datasets.

1. INTRODUCTION

Motion planning (also known as the "navigation problem" or the "piano mover’s problem") is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement Howard et al. (2002). Currently Mobile Robot has been widely used in examination and navigation particularly where static and unknown surroundings are involved. For example, consider navigating a mobile robot inside a building to a distant waypoint. It should execute this task while avoiding walls and not falling down stairs. A robotic system capable of some degree of self-sufficiency is the overall objective of an autonomous mobile robot and are required in many fields Gerkey and MatariÄG (2001); Saripalli et al. (2003); Falcone et al. (2003); Schilling and Jungius (1996); Bayestehtashk et al. (2013); Bayestehtashk and Shafran (2013). The motion of mobile robots in an unknown environment where there are obstacles needs an algorithm that could be aware of the math and avoid the collision Abiyev et al. (2010); Gu and Hu (2002). This paper deals with the intelligent path planning of intelligent autonomous systems in an unknown environment. Here the goal is to find best possible path from starting point to target point, that reduces time and distance, in a given environment, avoiding collision. A robotic vehicle is an intelligent mobile machine capable of autonomous operation in structured and unstructured environment, it must be able of sensing (perceiving its environment) thinking (planning, reasoning) and acting (moving and manipulating). However, mobile robots are appropriate tools for investigating optional artificial intelligence problems relating to word understanding and taking a suitable action, such as, planning missions, avoiding obstacles, and fusing data from many sources Gerkey et al. (2002). The algorithms for motion planning (Rastegar et al., 2009; Babaean et al., 2009) include grid-based search, interval-based search , potential fields, geometric algorithms and sampling-based algorithms LaValle (2006). low-dimensional problems can be solved by grid-based algorithm that overlay a grid on top of a configuration space, or geometric algorithm compute the shape and connectivity of the free region. Potential field algorithm are suitable for motion planning in high dimensional systems but stuck in local minima. Sampling-based method are able to handle local minima. Using sampling methods the probability of finding correct path converge to 1 as more time is spent. the sampling method are the state of the art for motion planning (Babaean et al., 2009) in the high dimensional space. Given source points and destination points in an unknown but statistic environment, we propose a markedly different approach based on connecting source and target points via angle-constrained paths. Our method lies in the category of sampling-based algorithms. It can be seen as a constrained variant of Isomap (Tenenbaum et al., 2000). Isomap was specifically designed for dimensionality reduction in the single-manifold setting. The angle constraint on paths is there to prevent connecting points from one cluster to points from a different, intersecting cluster. The resulting algorithm is implemented as a variation of Dijkstra’s algorithm (Dijkstra, 1959). Our method is simpler than the previous proposals in the literature and performs comparably to the best methods, both on simulated and some real datasets. The rest of the paper is organized as follows. In Section 2 we explain the notion of curvature constrained shortest-path and it’s connection with the angle constrained shortest-
path. In Section 3 we present our algorithm for Path Planning in an unknown environment. In Section 4 we present some numerical experiments. In Section 5 we discuss and outline our future work and development of our algorithm.

2. CONSTRAINED PATH

2.1. Curvature constraint

Neighborhood graphs play a central role in manifold learning, exploiting the fact that smooth submanifolds are locally flat. Recall that a neighborhood graph is a graph with vertices the sample points \( x_1, \ldots, x_N \). We consider two types of neighborhood structure (Maier et al., 2009):

- \( \varepsilon \)-ball. \( x_i \) and \( x_j \) are connected if \( \|x_i - x_j\| \leq \varepsilon \), where \( \| \cdot \| \) denotes the Euclidean norm.
- \( k \)-nearest neighbor. \( x_i \) and \( x_j \) are connected if \( x_j \) is among the \( k \)-nearest neighbors of \( x_i \) (in the Euclidean metric), or vice-versa.

The central idea in this paper is the use of constrained shortest-path distances in a neighborhood graph. The paths are constrained in order to control their smoothness. The constrained shortest-path distances are used to estimate geodesic distances reliably, even when the surface self-intersects, thus allowing us to mimic Isomap. We use the fact that the constrained and unconstrained shortest-path distances are similar for points belonging to the same submanifold, while usually different for points belonging to different submanifolds.

For an ordered triplet of points \( (x, y, z) \) in \( \mathbb{R}^3 \), we define the curvature as

\[
\text{curv}(x, y, z) = \begin{cases} 
(R(x, y, z))^{-1}, & \text{if } \angle(x, y, z) < \frac{\pi}{2}, \\
\infty, & \text{otherwise}
\end{cases}
\]  

(1)

where \( \angle \) stands for the angle and \( R(x, y, z) \) is the radius of the circle passing through \( x, y, z \).

\[
R(x, y, z) = \frac{\sqrt{||x - y||^2 + ||z - y||^2 + 2||x - y||||z - y|| \cos \angle(x, y, z)}}{\sin \angle(x, y, z)}.
\]  

(2)

with \( R(x, y, z) = \infty \) if \( x, y, z \) are aligned.

Definition. For a curvature \( \kappa > 0 \), we say that a path \( (x_{i_1}, \ldots, x_{i_m}) \) is \( \kappa \)-constrained if \( \text{curv}(x_{i_{t-1}}, x_{i_t}, x_{i_{t+1}}) \leq \kappa, \quad \forall t = 2, \ldots, m - 1 \).

To compute these constrained shortest-path distances we use a simple modification of Dijkstra’s algorithm. See Algorithm 1 below. When applied to a neighborhood graph with maximum degree \( \Delta \), its computational complexity is \( O(\Delta N \log N) \) per source point.

2.2. Angle constraint

For an ordered triplet of points \( (x, y, z) \) in \( \mathbb{R}^D \), define its angle as

\[
\angle(x, y, z) = \angle(xy, yz) = \cos^{-1} \left( \frac{\langle y - x, z - y \rangle}{\|y - x\| \|z - y\|} \right) \in [0, \pi]
\]  

(3)

We say that a sequence of points \( (x_{i_1}, \ldots, x_{i_m}) \) is \( \theta \)-angle constrained if the angles between successive segments are all bounded by \( \theta \), meaning

\[
\angle(x_{i_{t-1}}, x_{i_t}, x_{i_{t+1}}) \leq \theta, \quad \forall t = 2, \ldots, m - 1.
\]  

(4)

Figure 1 shows three D-dimensional points \( x, y, z \) which form vertices of a triangle such that \( x \) and \( z \) belong to the annulus neighborhood of point \( y \). Under above assumption the angle constraint \( \angle(x, y, z) < \theta \) where \( \theta < \pi/2 \) implies curvature constraint \( \text{curv}(x, y, z) < \kappa \) where \( \kappa = 2 \sin(\theta)/\sqrt{\frac{2}{3}(1 + \cos(\theta))} \).

![Fig. 1: x and z lie in annulus neighborhood of point y.](image)

3. ALGORITHM

In this section we present two algorithms to compute the constrained shortest path algorithm between source and target points. Algorithm 1 is a simple modification of dijkstra’s algorithm where we take into the account constrained when we travel from one node to the other nodes in the graph. For two sample points \( x, x' \), their graph shortest-path distance is defined as

\[
\Delta_{x}(x, x') = \inf \left\{ \sum_{s=0}^{m-1} w_{i_s, i_{s+1}} : x_{i_0} = x, x_{i_m} = x' \right\}.
\]  

(5)
where the infimum is over all possible tuples of indices \(i_1, \ldots, i_m \in [N]\), \(m \geq 1\). Define, similarly, their graph \(\kappa\)-curvature constrained shortest-path distance, denoted \(\Delta_{\kappa}(x, x')\), by requiring in addition to (5) that \(\text{curv}(x_{i_{s-1}}, x_i, x_{i+1}) \leq \kappa\) for all \(s \in [m-1]\).

Central to our methodology is the computation of curvature constrained shortest-path distances in the neighborhood graph. A simple modification of Dijkstra’s algorithm allows their computation with the same computational complexity—\(O(\Delta N \log N)\) per source point if \(\Delta\) is the maximum degree of the graph. See Algorithm 1, which applies for any constraint on adjacent pairs of edges, denoted by \(\sim\). We note that this can be implemented by applying the classical Dijkstra’s algorithm to the graph \(G\) with nodes the edges of the original graph \(G\), where two edges \((i, j)\) and \((j, k)\) in \(G\) are neighbors in \(G\) if \((i, j) \sim (j, k)\).

Algorithm 1  Constrained Dijkstra’s Algorithm

Input: weighted graph \(G = (V, E, w)\), source node \(s\), constraint on pairs of edges \(\sim\).
Output: constrained shortest-path distances from \(s\) to all other nodes.
Initialize \(p = s\), \(t = s\) and \(m = 0\).
For \(i \in V \setminus \{s\}\): \(\text{dist}[i] = \infty\), \(\text{cost}[i] = \infty\), \(\text{parent}[i] = \text{undefined}\), \(\text{temp}[i] = 0\), \(\text{path}[i] = []\).
For \(j = 1\) to \(N\) do
\(\text{dist}[i] = w_{ij}\) for each \(i \in \text{neighbors}(t)\).
for \(i = 1\) to \(N\) do
if \(\text{dist}[i] + m < \text{cost}[i]\) then
if \(\angle(i, t, p) < \text{constraint}\) or \(t = s\) then
\(\text{parent}[i] = t\) and \(\text{cost}[i] = \text{dist}[i] + m\).
end if
end if
end for
\(I\) is the vertex in \(G\) with min \(\text{cost}[i]\). Exclude \(I\) from the next search.
if \(\text{parent}[I] = \text{defined}\) then
Update \(\text{path}[I]\) by appending vertex \(I\) to the end of \(\text{path}[\text{parent}[I]]\).
end if
\(\text{dist}[i] = \infty\) for all \(i = 1, \ldots, N\).
\(t = I\) , \(p = \text{parent}[t]\).
if \(p = \text{undefined}\) then
break
end if
end for

Algorithm 2 is an exact algorithm where we build a new graph where the nodes of new graph are the edges of original graph See Figure 2 then we apply the simple dijkstra’s algorithm on this new graph where the shortest path distance between sources and target in the new graph is the constrained short path distance between source and target nodes in original graph Algorithm 3.

Algorithm 2  Build a new graph where the nodes of new graph are the edges of original graph

Input: weighted undirected graph \(G = (V, E, w)\), constraint on pairs of edges \(\sim\).
Output: Edge graph.
Form a new undirected graph \(G_e = (V_e, E_e, w_e)\) where \(V_e\), nodes in the new graph, are the edges of original graph \(G\).
for all \(x \in V\) and all pairs \(y, z \in \text{neighbors}(x)\) do
if \(\angle(y, x, z) < \text{constraint}\) then
\(x, z \in V_e\) and \(x, y \in V_e\) are two connected nodes in \(G_e\) with weight \(w(y, z) = (w(x, z) + w(x, y))/2\) where \(w(x, z) \in w_e\).
end if
end for

Algorithm 3  Constrained shortest path algorithm

Input: weighted undirected graph \(G = (V_e, E_e, w_e)\), source node \(s\), destination node \(d\).
Output: constrained shortest-path distance from \(s\) to \(d\).
Add a source nodes \(s_e\) to \(V_e\) and also add a set of edges between \(s_e\) and all the nodes in graph \(G_e\) of form \(s, i\) where \(i \in G\) and \(w(s, i) = \frac{1}{2} w(s, t)\).
Add a destination nodes \(d_e\) to \(V_e\) and also add a set of edges between \(d_e\) and all the nodes in graph \(G_e\) of form \(d, i\) where \(i \in G\) and \(w(d, i) = \frac{1}{2} w(d, t)\).
Apply regular Dijkstra’s algorithm to the undirected graph \(G_e = (V_e, E_e, w_e)\) with source node \(s_e\) and destination node \(d_e\).

![Fig. 2: a) Original graph. b) Edge graph](image)

We analyze the complexity of the algorithm in the
case of sparse graph with \( N \) nodes and each node with \( K \) neighbours. The time complexity of dijkstra’s algorithm on the new graph is \( O(\log(N_e)E_e) \) where \( N_e \) and \( E_e \) are the number of nodes and edges in the new edge graph \( G_e \) respectively and \( N \) and \( E \) are the number of nodes and edges in the original graph. For KNN graph we have \((KN)/2 \) edges and \( N_e = (KN)/2 \) is the number of nodes in \( G_e \), the maximum number of edges in this graph for every node happens when every node is connected to all of it’s neighbors which is \( K \). so the total number of edges for the new graph in the worst case of no constraint on the path would be \( E_e = (((KN)/2)K)/2 \) which gives complexity \( O(\log((KN)/2)(((KN)/2)K)/2) \) = \( O((K^2N/4)\log(KN/2)) \). Approximately complexity for the single source, the single destination is \( O(NK^2) \).

4. EXPERIMENTAL RESULTS

4.1. Finding path

In this experiments we sample 3000 points from the domain of the square with several holes in it. The parameter \( K \) to build the KNN graph is equal to 30 and angle constraint is chosen 30. Here the goal for robot is to find the shortest path from point \( A \) which is our starting point to the points \( B, C, D \) and \( E \). We apply Algorithm 1 to this domain. The constrained shortest path from point \( A \) to each of points \( B, C, D \) and \( E \) is shown in Figure 3. As it can be seen our algorithm is able to find the shortest path from \( A \) to other points where it avoids holes in the domain where in motion path planning problem are considered as obstacles.

In another experiment with the same setting as previous experiment, We apply Algorithm 3 to this domain. The result is identical to the approximate Algorithm 1 which has lower complexity and simpler implementation works as good as the exact Algorithm 3 to compute the constrained shortest path between source and destination points.

4.2. Robust to outlier

In another experiments to test the performances of our method in presence of outlier we draw a uniform random sampling from a box, then from a curve in the box with different rates. Then Applying constrained dijkstra’s algorithm Algorithm 1 to this data we compute the constrained shortest path between all the pairs. Figure 4 shows the longest constrained path between all the points, which actually path through the curve and detect the curve. In this experiment the total number of points is 1600 with 100 of them on the curve. This is an applications of motion planing for finding the correct path of the robot in presence of outliers.

5. CONCLUSION

We proposed an algorithm based on constrained shortest path to find the correct shortest path of robot between source and destination points in the presence of obstacles. The proposed method uses the contained version of shortest path algorithm to avoid obstacles. Both approximate and exact algorithms are very robust to the outliers and are able to detect the correct shortest path for moving robots.
References

Abiyev, R., D. Ibrahim, and B. Erin (2010). Navigation of mobile robots in the presence of obstacles. *Advances in Engineering Software* 41(10-S11), 1179 – 1186.

Babaeian, A., S. Rastegar, M. Bandarabadi, and M. Erza (2009, March). Modify kernel tracking using an efficient color model and active contour. In *System Theory, 2009. SSST 2009. 41st Southeastern Symposium on*, pp. 59–63.

Babaeian, A., S. Rastegar, M. Bandarabadi, and M. Rezaei (2009, March). Mean shift-based object tracking with multiple features. In *System Theory, 2009. SSST 2009. 41st Southeastern Symposium on*, pp. 68–72.

Bayestehtashk, A., M. Asgari, I. Shafran, and J. McNames (2013). Fully automated assessment of the severity of parkinson’s disease from speech. *Computer Speech & Language*.

Bayestehtashk, A. and I. Shafran (2013, May). Parsimonious multivariate copula model for density estimation. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 5750–5754.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik* 1(1), 269–271.

Falcone, E., R. Gockley, E. Porter, and I. Nourbakhsh (2003). The personal rover project: The comprehensive design of a domestic personal robot. *Robotics and Autonomous Systems* 42, 245–258.

Gerkey, B. and M. MatariÄ“G (2001). Principled communication for dynamic multi-robot task allocation. In D. Rus and S. Singh (Eds.), *Experimental Robotics VII*, Volume 271 of *Lecture Notes in Control and Information Sciences*, pp. 353–362. Springer Berlin Heidelberg.

Gerkey, B. P., M. J. Matarić, and G. S. Sukhatme (2002). Exploiting physical dynamics for concurrent control of a mobile robot.

Gu, D. and H. Hu (2002). Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems* 39(2), 73 – 86.

Howard, A., M. J. Matarić, and G. S. Sukhatme (2002, September). An incremental self-deployment algorithm for mobile sensor networks. *Auton. Robots* 13(2), 113–126.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. Available at http://planning.cs.uiuc.edu/.

Maier, M., M. Hein, and U. von Luxburg (2009). Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Theoretical Computer Science* 410(19), 1749–1764.

Rastegar, S., A. Babaeian, M. Bandarabadi, and Y. Toopchi (2009, March). Airplane detection and tracking using wavelet features and svm classifier. In *System Theory, 2009. SSST 2009. 41st Southeastern Symposium on*, pp. 64–67.

Saripalli, S., G. Sukhatme, and J. Montgomery (2003). An experimental study of the autonomous helicopter landing problem. In B. Siciliano and P. Dario (Eds.), *Experimental Robotics VIII*, Volume 5 of *Springer Tracts in Advanced Robotics*, pp. 466–475. Springer Berlin Heidelberg.

Schilling, K. and C. Jungius (1996). Mobile robots for planetary exploration. *Control Engineering Practice* 4(4), 513 – 524.

Tenenbaum, J. B., V. de Silva, and J. C. Langford (2000). A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500), 2319–2323.