13th National Conference on Mechanisms and Machines (NaCoMM07),
IISc, Bangalore, India. December 12-13, 2007

NaCoMM-2007-099

Robot path planning using silhouette method

Aneeta Bhattacharyya[1], Ekta Singla[2*], Bhaskar Dasgupta[2]
[1] University of Calcutta, India
[2] Mechanical Department, IIT, Kanpur, India

[*] Corresponding author (email: ektas@iitk.ac.in)

Abstract

This paper presents an implementation of the silhouette method of robot path planning. It is a roadmap based method which consists of generating the silhouette of the work cell, forming a network of simple semi-free paths by connecting the silhouette curves to each other and selecting the path from the start position to the end position, in the configuration space. Dijkstra's shortest path algorithm is used for tracing the path between the two given points out of the connected graph. A few examples are presented to show the efficacy of the algorithm. The value of this method is that it is complete and it is applicable for a general n-dimensional C-space, even though the path found by this method is, in general, just a feasible one. The examples used in the present paper are for two-dimensional workspaces, but its planned continuation into higher dimensions is the actual valuable outcome of this research.
Keywords: Robot motion planning, Silhouette method, Dijkstra's shortest path algorithm.

1 Introduction

In this era of automation where robots and autonomous machines are in use for almost every field, may it be assembly layouts, surgical applications or entertainment, development of efficient robot-path-planners is surely an important requirement. The proficiency and suitability of a planner is evaluated based on the aspects of its completeness, optimality of the path provided by it and its capability to work in higher dimensions. General approaches to solve the basic motion planning problem are Roadmap methods, Cell decomposition methods and Potential field methods [1]. Extensive research has been done on the implementation of these methods.

Roadmap methods, as presented in [2, 3, 4, 5], consists of constructing the network of free paths in C-space and searching the path in this network between the desired locations. These methods are known for their completeness, i.e., they provide a path if one exists and show failure, otherwise. Though the paths provided by the roadmap methods are not, in general, optimal, but assurance to get a feasible path in itself is important for many of the cluttered environments. Cell decomposition methods consist of successive decomposition of the robot's configuration space into cells of some geometric shape. They mainly work on two strategies, one is by decomposing the workspace exactly into cells (refer [6]), and another is the approximate decomposition (see [1, 7]). The former is complete in providing a path for only lower dimensional C-spaces. Higher order workspaces are better dealt with approximate cell decomposition methods, though at the cost of gaurantee to get a path. Absence of optimality in the paths and involvement of memory exhaustion problems limit the utility of these methods. Potential field methods claim for generating the optimal paths, though the path obtained, in general, can be only locally optimal. The successful results of many researchers [8, 9, 10] show the successive improvements in this type of methods, particularly towards global opimality, but still lacking in the aspect of completeness.

All these issues bring out the fact that motion planning for robots of any number of degrees of freedom (DOF's), working in cluttered environments, is still a problem, that invites further work. Necessity to get a reliable path planner which can work with higher dimensions of C-space leads us to the selection of roadmap methods. Besides, once we get a path, it can be improved towards optimality by hybrid methods as discussed in [4, 11].

The roadmap methods which can deal with higher dimensional C-spaces came into existence with the development of Probabilistic Roadmap Method (PRM), presented by Kavraki et al [2]. Successful in providing a feasible path for robots with large DOF's, this two-phase method got enough attention by the researchers. However, the two limitations of unnecessary collision checks and narrow passage problem, raised the need for further research. Sanchez and Latombe [12] presented an improvement over PRM by providing a lazy-in-collision-detection technique SBL (Single-query Bi-directional Lazy in collision detection) PRM. It avoids pre-computing a roadmap and involves searching the robot's free space locally. This helped in reducing the collision checks and hence the time consumption. However, the problem of narrow passage still remained. Isto and Saha [3] and Brock and Kavraki [15] proposed other approaches to work with higher dimensional C-space.

Katz and Hutchinson [4] emphasized on the narrow passage problem and proposed a method of integrating the basic probabilistic path planning with specific potential functions near the narrow passages. This helped not only to make the robot pass through the narrow passage but also in reducing the graph searching time. The implementation of the method is shown only for 2-dimensional cases while the definition of potential functions for higher dimensional cases may come out as another problem. Valero and Ceccarelli et al [13, 14] presented their approach for path planning and implemented successfully on 2-dimensional redundant manipulators. Silhouette method, one of the roadmap methods, presented by Canny [16], is however free from any narrow passage problem. This method is considered difficult in implementation, but its completeness, ability to deal with higher-dimensional configuration and avoidance of narrow passage problem motivate us to attempt its implementation. This paper presents the schematic for implementation of the silhouette path planning method and presents the results for 2-dimensional C-spaces.

Section 2 describes the Silhouette method. The algorithm to implement the method is given in section 3. Section 4 presents some results with their descriptions. The conclusions of the work are summarized in section 5.

2 Silhouette Method

The word 'silhouette' literally means a dark shadow or outline of an object against a light background. As the name suggests silhouette path planning technique consists of generating the silhouette of the work cell and developing the roadmap by connecting these silhouette curves to each other. This is a network of semi-free paths which is searched to select the path from the start configuration to the goal configuration.

The n-dimensional configuration space is operated upon by the recursive approach of sweeping with lower dimensional hyperplanes. At the zero recursion level, a hyperplane of dimension $(n-1)$ is swept across the C-space in a direction perpendicular to the hyperplane, by constant steps. At each step the extremal points of C-space are marked which when joined together gives rise to algebraic curves called as silhouette curves. The extremal points are also connected to the silhouette curves formed at the previous step, if any.

The network of silhouette curves constructed till this point is not yet complete. There may remain some of the sets of extremal points which are not yet included in the network. These are represented by critical points, the points where the connectivity of the silhouette curves change. During the sweep process, whenever a critical point is encountered, a hyperplane at this point is also included in the set of $(n-1)$ dimensional hyperplanes collected at this recursion level. All these slices are swept by the next-lower-dimensional hyperplanes in the next recursion level. The recursion ends when there is no critical point to connect to the silhouette curves in the set that is currently swept out, or when the hyperplane being swept becomes one dimensional. This leads to the formation of the silhouette of the entire workspace and of the obstacles within it.

To find a path from the initial configuration to the goal configuration, these two points are also considered as critical points. The path between the two points is traced out by considering the roadmap as a connected graph and using a suitable shortest path algorithm. The edge set of the graph would consist of the algebraic curve segments of the roadmap and the vertex set would consist of the endpoints of these curve segments.

3 Implementation

In this paper, an implementation of the silhouette method for 2-dimensional C-space is presented. For this case, sweeping hyperplane, at zeroth recursion, reduces to a sweep-line. An axis, say x-axis, is selected as the sweeping direction and the sweep-lines are parallel to the another coordinate axis. It is possibly the simplest version of this method but it can turn out to be computationally quite involved.

The workspace is input as a set of line segments which, when connected sequentially, forms the configuration space of the robot. The algorithm is implemented using a point classification procedure which accepts, as input, a specific x-coordinate value and determines whether the line drawn through that point and perpendicular to the x-axis contains any extremal points. If such extremal points are found on the slice, which is a straight line for a 2-dimensional workspace, the procedure connects those points to the extremal points of the previous slice (if any). On complete execution of this procedure, the fully connected silhouette of the workspace is obtained. Next, the two end points are connected to the silhouette curve by considering them as critical points.

Dijkstra's algorithm [17] is used to find the shortest path between the two desired points. Dijkstra's algorithm works by visiting each of the vertices in a graph beginning with the starting vertex. It then repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined. It expands outwards from the starting vertex until it reaches the goal vertex. The advantage of using Dijkstra's algorithm is that it is guaranteed to find a shortest path from the starting vertex to the goal vertex, as long as a path exists.

3.1 Roadmap development

In the first phase of network development, the x-axis is selected as the sweeping direction and a step-size of one unit is chosen to cut the slices. The slices are the lines parallel to y-axis. The algorithm enumerated below is executed for each slice.

1. Check for intersection of the slice with the workspace boundary. Store the points of intersection.
2. Connect points of intersection of the new slice with those of the previous slice.
3. Check for critical slices.
4. Connect all critical points to the rest of the silhouette curves.

In step 1, the procedure determines whether the line segment, just formed, intersects any of the line segments, given in the input set, i.e., the workspace. Step 2 collects all the extremal points of the present slice and connects them properly to the extremal points of the previous slice (if any). For this purpose, it takes the points of intersection of the input line segments with the slice passing through the middle of these two slices. If any of the intersection points on the middle slice matches with the mid-point of a probable line segment then that probable segment is considered to be genuine, else it is considered to be non-existent. Step 3 checks for critical slices by examining the intersection points on each slice and detecting the points on a slice that have not been connected to either its previous slice or its next slice. If such points are found then they are considered to be critical and the corresponding slices are considered to be critical slices. The slices passing through the initial and final points are also included in the crticle slices set. Step 4 connects the critical points to the silhouette curves formed in step 2 using the corresponding critical slices to obtain a fully connected network of silhouette curves of the entire workspace.

3.2 Path search

The network constructed in the previous phase is searched for the desired path between the two configuraions given. In this paper, the Dijkstra's shortest path algorithm [17] is used which is enumerated below.

1. Create adjacency matrix for the silhouette curve.
2. Label start node as permanent and all other nodes as temporary.
3. While cost of path to goal node is not found,
    - select least cost edge that connects a permanent node to a temporary node.
    - update cost of path to temporary node as the sum of edge cost and cost of path to permanent node.
    - mark selected temporary node as permanent, save a pointer from newly created permanent node to the previous permanent node.
4. Trace back the path from initial point to final point.

Step 1 creates an adjacency matrix for the silhouette curve that was found during point classification. In the adjacency matrix, nodes represent the end points of the algebraic curve segments and the cost of the edges represents the linear distance between the end points of the segments. Step 2 begins the Dijkstra's algorithm by labeling the starting node as permanent and all the other nodes as temporary. Step 3 iteratively searches the graph to find the least-cost edge that connects a permanent node to a temporary node. This temporary node is then marked as permanent and the cost of the path from the source to the present node is updated to the sum of the cost of the path to the permanent node and the cost of the edge between the permanent node and the present node. The iteration terminates when the destination node becomes permanent, i.e., the cost of the shortest path from the source to the destination has been calculated. Step 4 traces back the path from the destination to the source by following back the links.

4 Results and Discussions

The algorithm, implemented in C language, is tested on a number of imaginary workspaces having geometric boundaries. The results obtained for a few of the cases are presented in this section to show the promise of the implemented path planning algorithm. The execution of the algorithm needs to be supplied the step-size for the hyperplanes to cut the slices in the sweeping direction, along with the coordinates of the initial and the goal configurations.

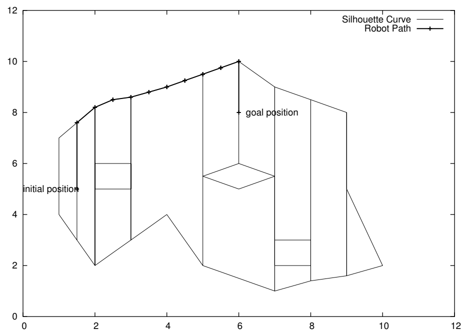

Figure 1: Path between (1.5,5) and (6,8)

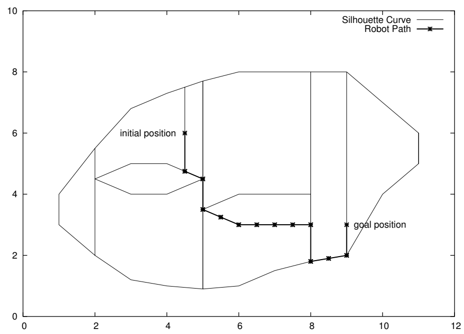

Figure 2: Path between (4.5,6) and (9,3)

Fig. 1 shows the path planned by the developed algorithm for the two given terminal locations. Initial point $(1.5, 5)$ is to connect the goal point $(6, 8)$. The path, consisting of semi-free straight lines, is highlighted with bold lines. The crossmarks over it show the points of stepwise-connection of the silhouette while its making. The path consists of the parts of the two critical slices passing through the start point and the end point and a portion of the silhouette of the workcell cut by these two slices.

Another case with end-points $(4.5, 6)$ and $(9, 3)$ in a different workspace is fed to the algorithm and the resulting path is shown in fig. 2. The path passes conveniently through the passage between the two obstacles. As the algorithm provides the paths which touch the workspace boundaries, a small offset (tolerance) can be added to the boundaries, as protection walls, before supplying the workspace to the algorithm.

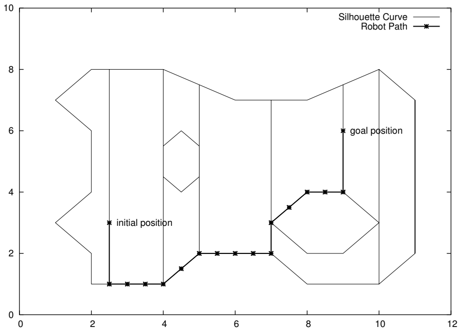

Figure 3: Path between (2.5,3) and (9,6)

Initial and final positions in the next case are $(2.5,3)$ and $(9,6)$, for another environment with two obstacles. The planned path uses the portions of the two slices passing through the start and goal positions and parts of both the workcell and the obstacle boundaries, as shown in fig. 3. It can be seen that, provided the silhouette formed during the first phase of network development is correctly connected, a feasible path is assured, as long as it exists.

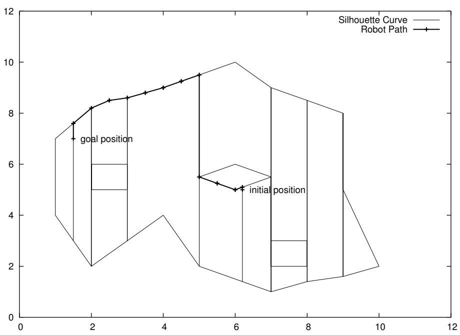

Figure 4: Path between (1.5,7) and (6.2,5)

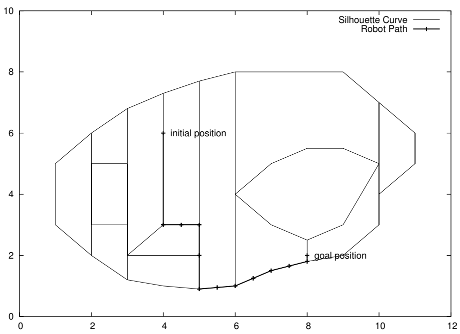

Figure 5: Path between (4,6) and (8,2)

Figs. 4 and 5 are the examples of the cases when the initial and/or the final point is very near to the boundaries of the workcell or obstacles. The path formed reaches the desired locations easily and, unlike potential field methods, is free from the problem of defining the potential functions to avoid the trapping in local minima. We understand that the path found in the network, using Dijkstra's algorithm, is one of the feasible paths possible between the two desired locations.

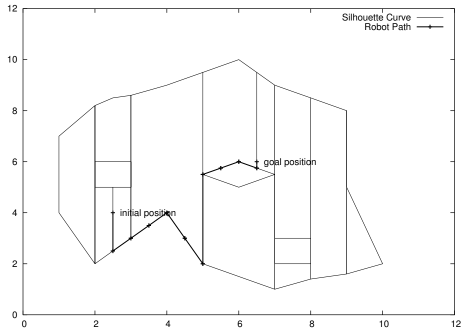

Figure 6: Path between (2.5,4) and (6.5,6)

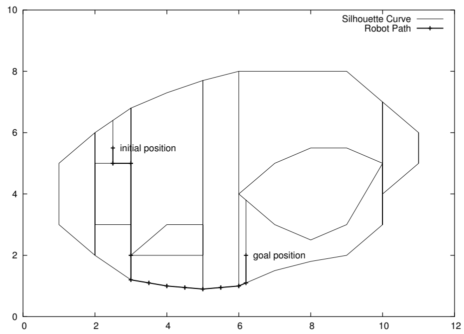

Figure 7: Path between (2.5,5.5) and (6.2,2)

Figs. 6 and 7 show paths for some new sets of end-points in the above environments. All these results highlight the successful implementation of the algorithm and shows the efficiency of the Silhouette method.

It is noticed that the silhouette curves formed in all the cases above have polygonal geometry but if the boundary of the workcell or obstacle is actually curved then that can be better represented by taking a smaller step size. Another important observation is that, if any boundary segment of the workcell or an obstacle coincides with the sweeping hyperplane, as occured in some of the presented cases, it leads to degeneracy and needs special attention while tracing the silhouette curves.

5 Conclusion

This paper proposes a possible implementation of the silhouette method for robot motion planning in 2-dimensional workspaces. The main reason for particularly focussing on this method is that it is complete and it works for any n-dimensional C-space, even though the path found by this method is only a feasible one. But in higher dimensions, due to high computational complexity, quite often it is important to trace a possible path than to find the most optimal path. This algorithm recursively reduces the problem of constructing a roadmap in n-dimensions to several sub-problems of constructing roadmaps in sets of dimension $n-1$.

It is clear, however, that apart from its theoretical utility there is no practical utility of silhouette method implemented in 2-dimensions. But it does pave the way for future implementation of the method in higher dimensions that can then be used for actual path planning of a robot in configuration spaces of higher dimensions.

References

[1] J. C. Latombe, "Robot Motion Planning", Boston, MA: Kluwer Academic Publishers, 1991.

[2] L. E. Kavraki, P. Svestka , J. C. Latombe and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces", *IEEE Trans. on Robotics and Automation*, Vol. 12, No. 4, August 1996, pp. 166-171.

[3] P. Isto and M. Saha, "A Slicing Connection Strategy for Constructing PRMs in High-Dimensional Cspaces", *Proc. IEEE International Conference on Robotics and Automation*, Ornaldo, Florida, May 2006, pp. 1249-1254.

[4] R. Katz and S. Hutchinson, "Efficiently Biasing PRMs with Passage Potentials", *IEEE International Conference on Robotics and Automation*, Ornaldo, Florida, May 2006, pp. 889-894.

[5] L. E. Kavraki, M. N. Kolountzakis and J. C. Latombe, "Analysis of Probabilistic Roadmaps for Path Planning", *IEEE Trans. on Robotics and Automation*, Vol. 14, No. 1, February 1998, pp. 566-581.

[6] F. Avnaim, J. D. Boissonnat and B. Faverjon, "A Practical Exact Motion Planning Algorithm for Polygonal Objects amidst Polygonal Obstacles ", Technical Report No. 890, INRIA, Sophia-Antipolis, France, 1988, pp. 1656-1661.

[7] D. Zhu and J. C. Latombe, "New Heuristic Algorithms for Efficient Hierarchical Path Planning", *IEEE Trans. on Robotics and Automation*, Vol. 7, No. 1, Febuary 1991, pp. 9-12.

[8] T. Cecil and D. E. Marthaler, "A Variational Approach to Path Planning in Three Dimension using Level Set Methods", *Journal of Computational Physics*, Vol. 211, 2006, pp 179-197.

[9] E. S. Conkur, "Path Planning using Potential Fields for Highly Redundant Manipulators", *Robotics and Automation Systems*, Vol. 52, 2005, pp 209-228.

[10] J. Wang, X. B. Wu and Z. L. Xu, "Decentralized Formation Control and Obstacles Avoidance Based on Potential Field Method", *Proc. of fifth International Conference on Machine Learning and Cybernetics*, Vol. 2, Dalian, August 2006, pp 803-808.

[11] J.C. Latombe, "Motion Planning: A Journay of Robots, Molecules, Digital Actors and Other Artifacts", *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 11, 1999, pp. 1119-1128.

[12] G. Sanchez and J. Latombe, "A Single-query Bidirectional Probabilistic Roadmap Planner with Lazy Collision Checking", *International Symposium on Robotics Research*, Springer, 2001, pp. 403-417.

[13] M. Ceccarelli, F. Valero, V. Mata, I. Cuadrado, "Generation of Adjacent Configurations for a Collision-free Path Planning of Manipulators", *Robotica (Cambridge print)*, Vol. 14, 1996, pp. 391-396.

[14] F. Valero, V. Mata, I. Cuadrado and M. Ceccarelli, "A formulation for Path Planning of Manipulators in Complex Environments by Using Adjacent Configurations", *Advanced Robotics*, Vol. 11, No. 1, 1997, pp. 33-56.

[15] O. Brock and L. E. Kavraki, "Decomposition-based Motion Planning: A Framework for Real-time Motion Planning in High-Dimension", *IEEE Trans. on Robotics and Automation*, Vol. 2, 2001, pp. 1469-1474.

[16] C. F. Canny, "The Complexity of Robot Motion Planning," MIT Press, Cambridge, MA, 1988.

[17] E. W. Dijkstra, "A Note on Two Problems in Connexions with Graphs", Numerische Mathematik 1, 1959, pp. 269-271.