Visibility-based Robot Motion Planning

(Lecture slides)

Subir Kumar Ghosh
School of Computer Science
Tata Institute of Fundamental Research
Mumbai 400005, India
ghosh@tifr.res.in

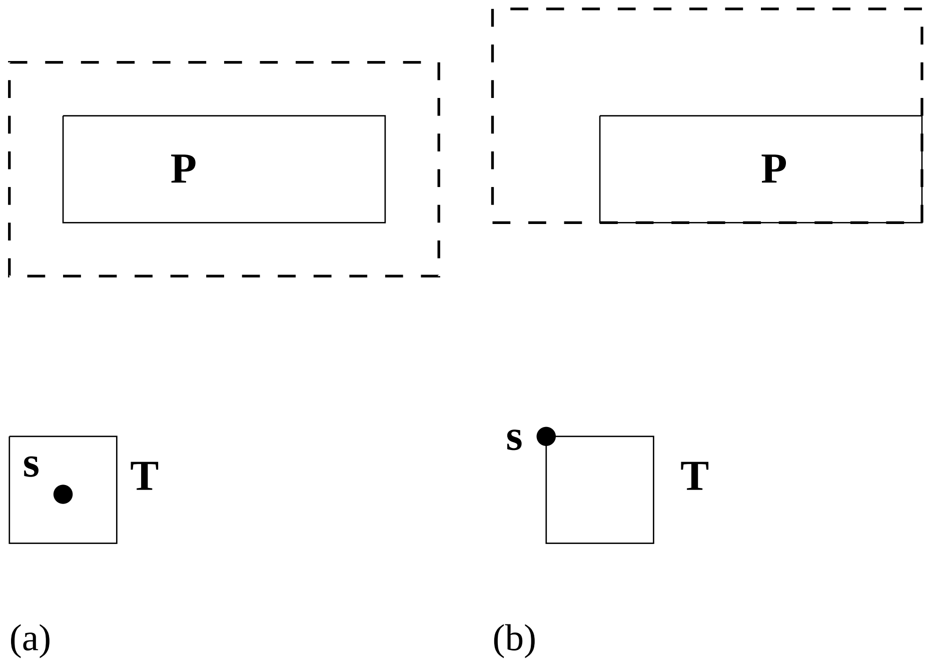
Overview

1. Computing the configuration space
2. Computing Euclidean shortest paths
3. Computing minimum link paths
4. Computing bounded curvature paths
5. Exploring an unknown polygon: Continuous visibility
6. Exploring an unknown polygon: Discrete visibility
7. Exploring an unknown polygon: Bounded visibility

Robot Motion Planning

One of the main problems in robotics, called motion planning, is to find a collision-free path amidst obstacles for a robot from its starting position to its destination.

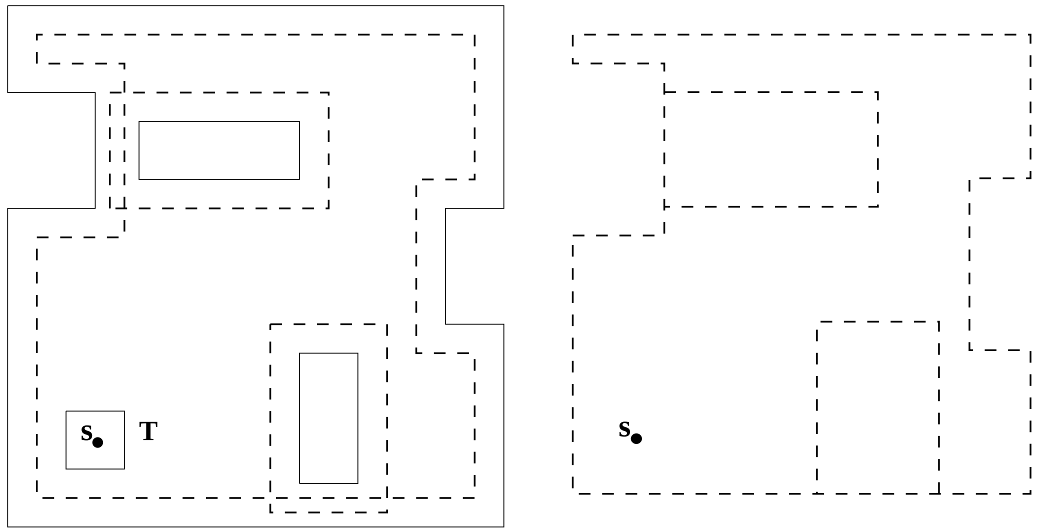
Minkowski sum



Above figures show the Minkowski sums of P and T with s as the reference point (under translation).

1. J-C Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

Computing configuration space



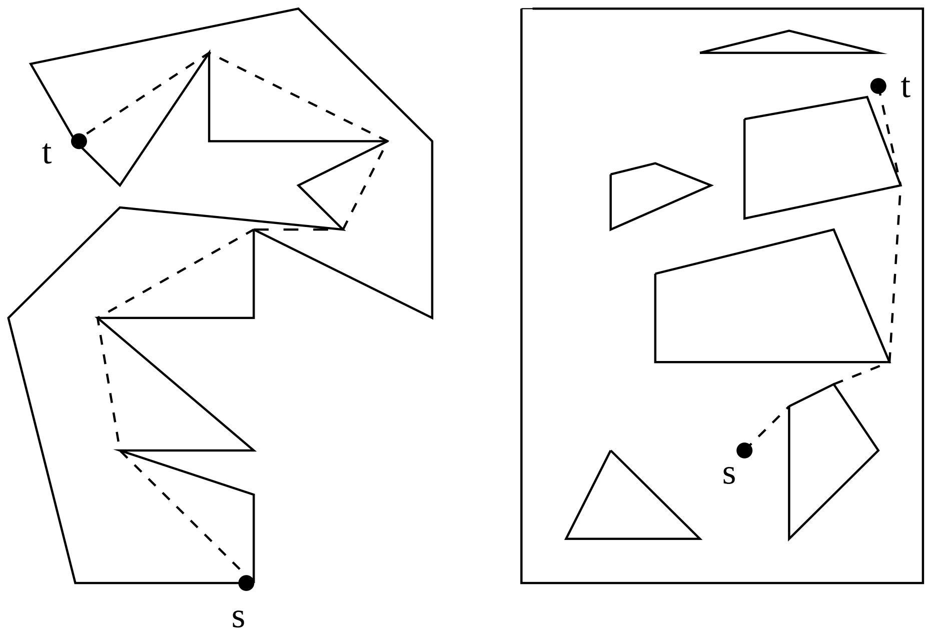
The configuration space can be computed using Minkowski sum and any path inside the free configuration space is a collision-free path.

The problem of computing collision-free path of a rectangle in the actual space is now reduced to that of a point in the free configuration space.

This concept was introduced in the following paper:

1. T. Lozano-Perez and M. A. Wesley, *An algorithm for planning collision-free paths among polyhedral obstacles*, Communication of ACM, 22 (1979), 560-570.

Computing Euclidean shortest paths

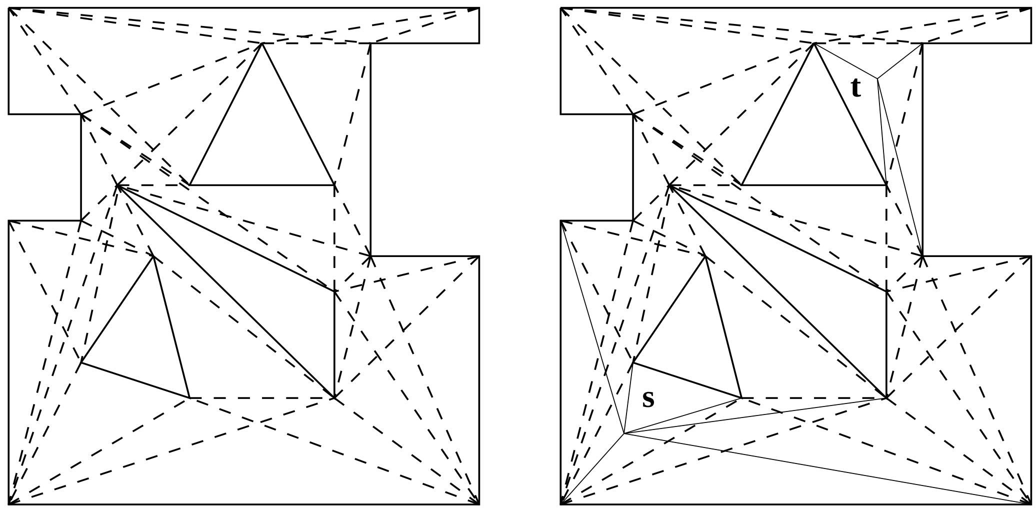


The *Euclidean shortest path* (denoted as $SP(s,t)$) between two points s and t in a polygon P is the path of smallest length between s and t lying totally inside P.

Let $SP(s,t) = (s,\ u_1,\ u_2, \ldots,\ u_k,\ t)$. Then, (i) $SP(s,t)$ is a simple path, (ii) u_1, u_2,..., u_k are vertices of P and (iii) for all i, u_i and u_{i+1} are mutually visible in P.

1. J. Mitchell, *Geometric shortest path and network optimization*, Handbook in Computational Geometry (edited by J.-R Sack and J. Urrutia), Elsevier Science Publishers B.W., Chapter 15, 1999.

Computing $SP(s, t)$ using visibility graph



The visibility graph of a polygon P with non-intersecting polygonal holes or obstacles is a graph whose vertex set consists of the vertices of P and whose edges are visible pairs of vertices.

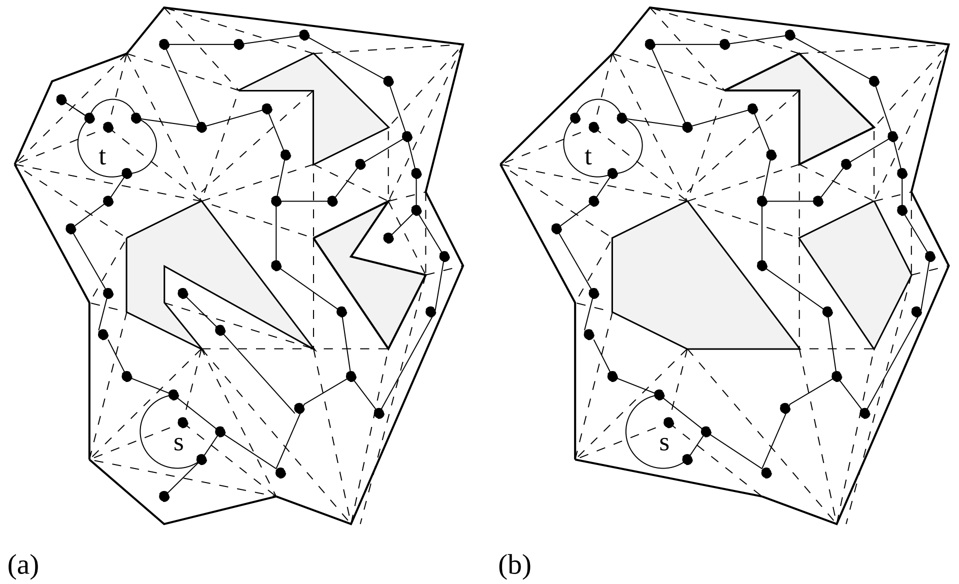
Assign the length of each visible pair as an weight on the corresponding edge in the visibility graph and use the following algorithm to compute $SP(s, t)$.

1. M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, Journal of ACM, 34 (1987), 596-615. Running time: $O(n \log n + E)$, where E is the number of edges in the visibility graph.

Algorithms for computing visibility graphs

1. T. Lozano-Perez and M. A. Wesley, *An algorithm for planning collision-free paths among polyhedral obstacles*, Communication of ACM, 22 (1979), 560-570. Running time: $O(n^3)$.
2. D. T. Lee, *Proximity and reachability in the plane*, Ph. D. Thesis, University of Illinois, 1978. Running time: $O(n^2 \log n)$.
3. M. Sharir and A. Schorr, *On shortest paths in polyhedral spaces*, SIAM Journal on Computing, 15 (1986), 193-215. Running time: $O(n^2 \log n)$.
4. E. Welzl, *Constructing the visibility graph for n line segments in $O(n^2)$ time*, Information Processing Letters, 20 (1985), 167-171. Running time: $O(n^2)$.
5. T. Asano and T. Asano and L. J. Guibas and J. Hershberger and H. Imai, *Visibility of disjoint polygons*, Algorithmica, 1 (1986), 49-63. Running time: $O(n^2)$.
6. S. K. Ghosh and D. M. Mount, *An output sensitive algorithm for computing visibility graphs*, SIAM Journal on Computing, 20 (1991), 888-910. Running time: $O(n \log n + E)$. Space: $O(E)$. E is the number of edges in the visibility graph.

Computing $SP(s,t)$ using partial visibility graph



Above figures show dual graphs of a triangulation of P before and after removing nodes of degree 1.

1. S. Kapoor, S. N. Maheshwari and J. Mitchell, *An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane*, Discrete and Computational Geometry, 18(1997), 377-383. Running time: $O(n+h^2 \log n)$, where h is the number of holes. Space: $O(n)$.
2. H. Rohnert, *Shortest paths in the plane with convex polygonal obstacles*, Information Processing Letters, 23(1986), 71-76. Running time: $O(n \log n + h^2)$.

Computing $SP(s,t)$ directly

$SP(s,t)$ can be computed directly once the shortest path map is constructed using continuous Dijkstra Method.

This method involves simulating the effect of a 'wavefront' propagating out of s. The *wavefront* at a distance d from s is the set of all points $u \in P$ such that $|SP(s,u)| = d$.

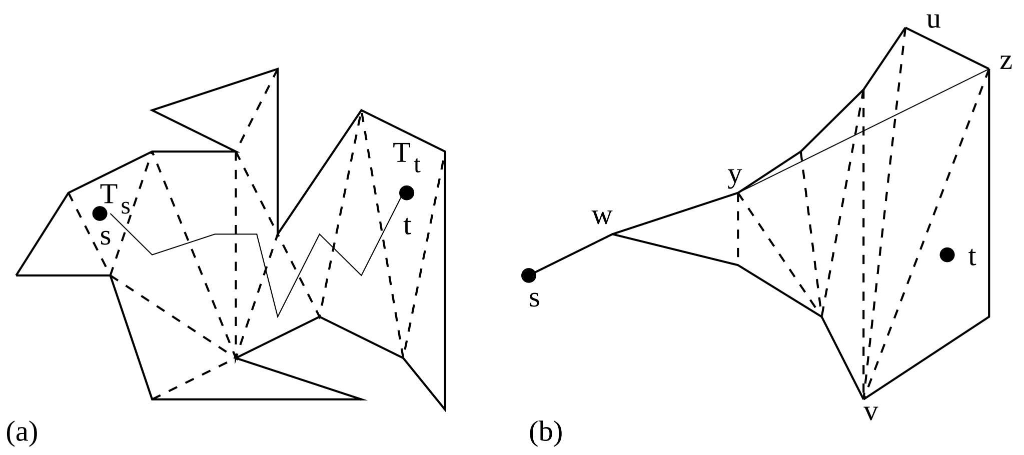
The propagation can be carried out on *cell-by-cell* basis after decomposing the entire region of P into "conforming subdivision".

1. J. Hershberger and S. Suri, *An optimal-time algorithm for Euclidean shortest paths in the plane*, SIAM Journal on Computing, 28(1999), 2215-2256. Running time: $O(n \log n)$. Space: $O(n \log n)$.

Open Problem

Can one compute $SP(s,t)$ in $O(n + h \log h)$ time and $O(n)$ space?

Computing $SP(s,t)$ in a simple polygon



$SP(s,t)$ passes only through the triangles in the path from T_s and T_t in the dual of a triangulation of P.

$SP(s,t)$ is *outward convex* at every vertex on the path.

Let (u,v,z) be a triangle such that $SP(s,u)$ and $SP(s,v)$ have already been computed. Then $SP(s,z)$ can be computed by drawing tangent from z to $SP(s,u)$ or $SP(s,v)$.

1. D. Lee and F. Preparata, *Euclidean shortest paths in the presence of rectilinear boundaries*, Networks, 14 (1984), 303-410. Running time: $O(n)$.

Computing the Euclidean shortest path tree

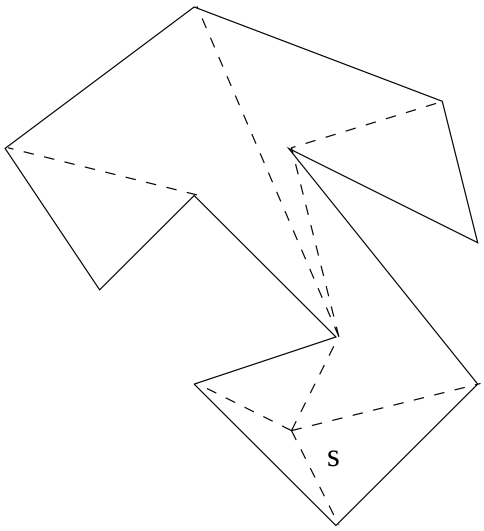
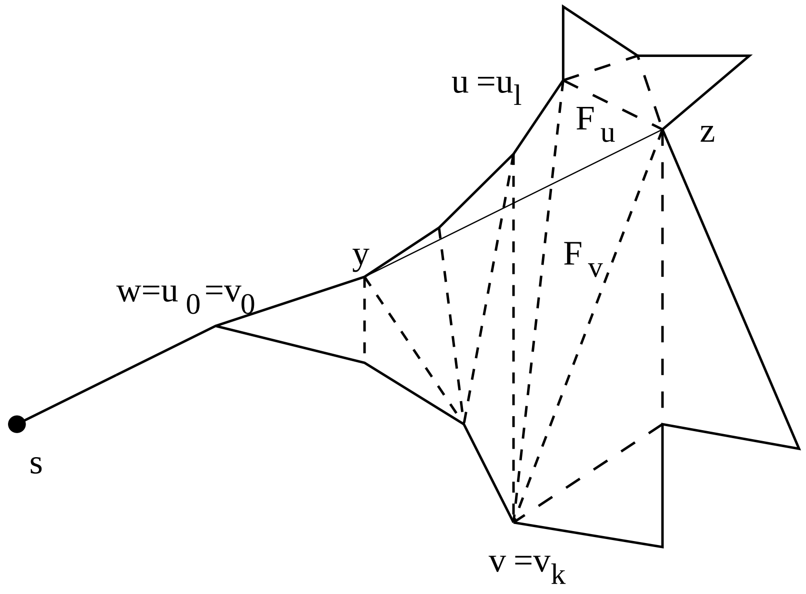


Figure shows the Euclidean shortest path tree from a given point s to all vertices of a simple polygon.

The Euclidean shortest path tree from s is the union of $SP(s, u)$ to all vertices u of the polygon.

1. L. Guibas, J. Hershberger, D. Leven, M. Sharir and R. Tarjan, *Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons*, *Algorithmica*, 2(1987), 209-233. Running time: $O(n)$.



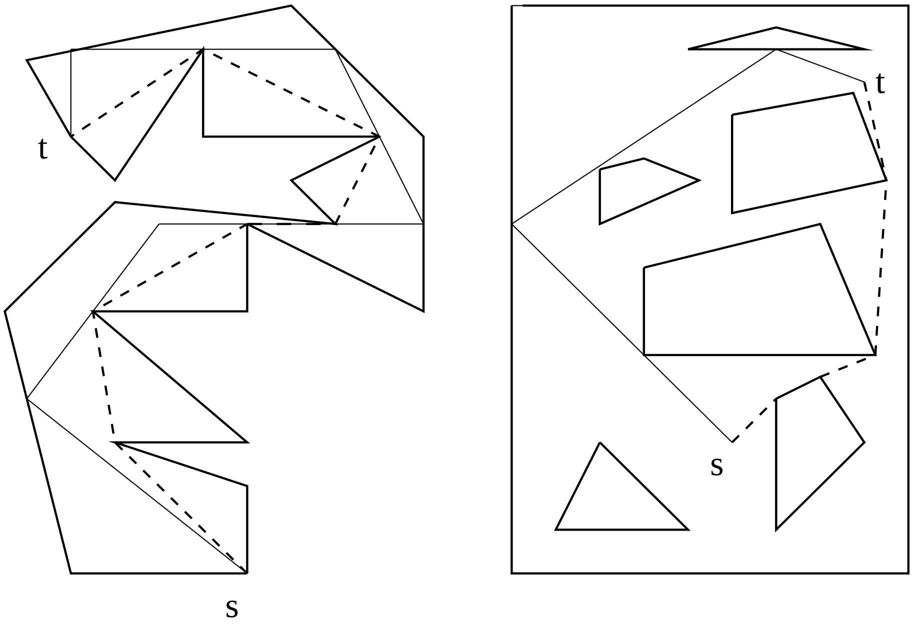
The current funnel can be maintained as a sorted list $[u_l, u_{l-1}, ..., u_1, w, v_1, v_2, ..., v_k]$ where $w = u_0 = v_0$ is the apex of the funnel, $SP(w, u) = [u_0, ..., u_l]$, $SP(w, v)$ $= [v_0, ..., v_k]$ and $u_l = u$, $v_k = v$.

The tangent yz splits a funnel into two funnels and both funnels can be propagated in $O(n)$ time using Finger search tree.

Open Problem

Can one compute shortest path tree from a point in a triangulated simple polygon in $O(n)$ time without using Finger search tree?

Computing minimum link paths

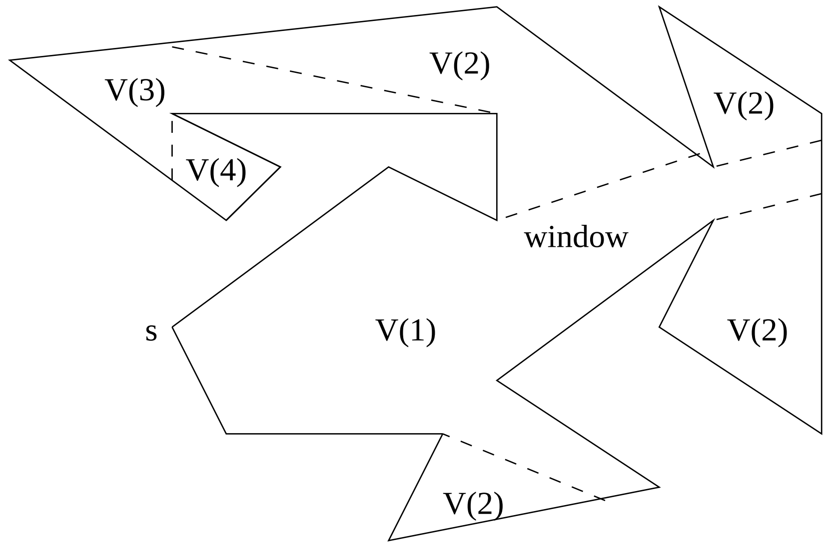


A minimum link path connecting two points s and t inside P (denoted by $MLP(s,t)$) is a polygonal path with the smallest number of turns or links.

1. S. Suri, *A Linear Time Algorithm for Minimum Link Paths Inside a Simple Polygon*, Computer Graphics, Vision, and Image Processing, 35 (1986), 99-110. Running time: $O(n)$.

2. S. K. Ghosh, *Computing the visibility polygon from a convex set and related problems*, Journal of Algorithms, 12 (1991), 75-95. Running time: $O(n)$.

Suri's algorithm for computing $MLP(s, t)$



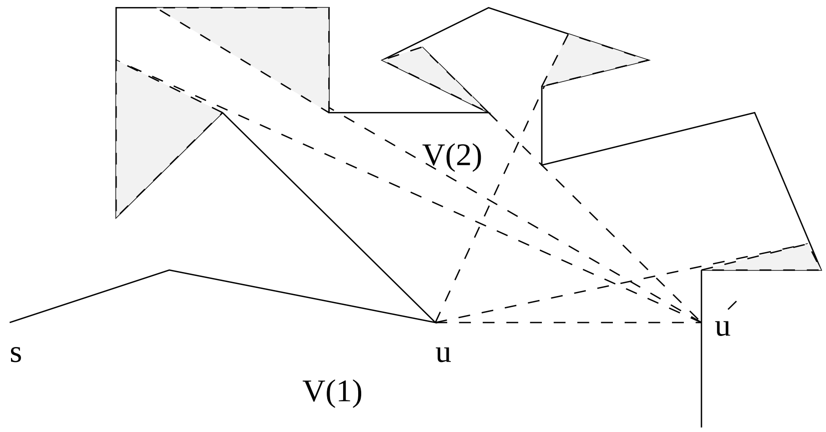
Number of links (called *link distance*) required from s to any point of $V(i)$ is i.

Windows in P are constructed edges on the boundary of $V(i)$ for all i.

$V(1)$ is the visibility polygon of s. For $i > 1$, $V(i)$ is the set of points of P weakly visible from some window w, i.e., any point of $V(i)$ is visible from some point of w.

Decompose P by introducing windows according to the link distance and then compute $MLP(s, t)$ by introducing turning points on the appropriate windows.

Computing the weak visibility polygon

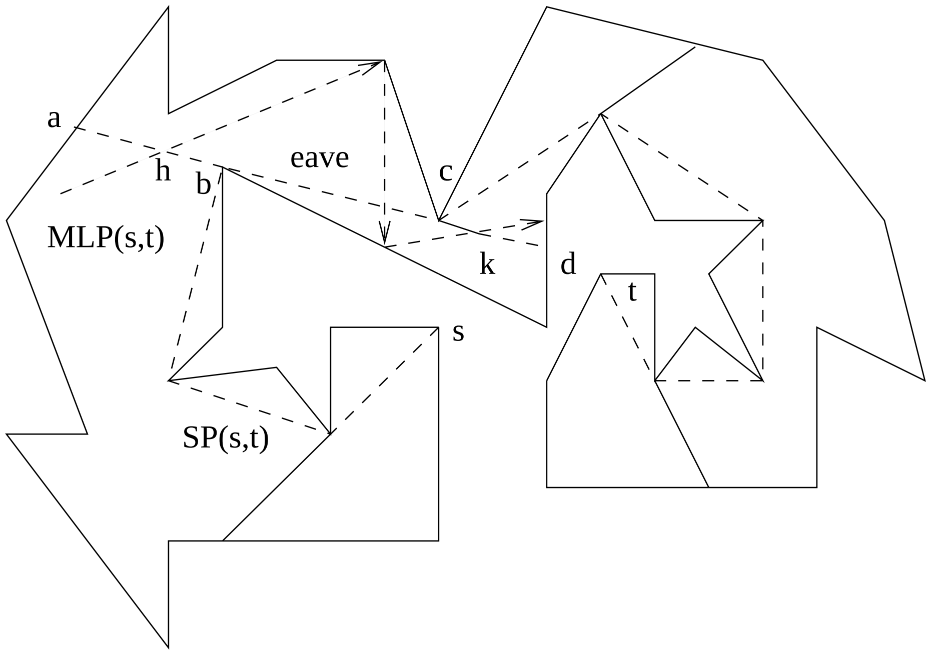


Compute the Shortest path tree rooted at s (denoted by $SPT(s)$). Compute $V(1)$. Let uu' be a window of $V(1)$ where u is a vertex of P and u' is the extension point of su. Construct $SPT(u')$ from $SPT(u)$. Compute Weak visibility polygon of uu' from $SPT(u)$ and $SPT(u')$.

A point $v \in P$ is weakly visible from uu' iff $SP(u, v)$ and $SP(u', v)$ meet only at v.

1. S. K. Ghosh, A. Maheshwari, S. P. Pal, S. Saluja and C. E. Veni Madhavan, *Characterizing and recognizing weak visibility polygons*, Computational Geometry: Theory and Applications, 3 (1993) 213-233.
2. J. Hershberger, *An optimal visibility graph algorithm for triangulated simple polygons*, Algorithmica, 4 (1989), pp. 141-155.

Ghosh's algorithm for computing $MLP(s,t)$

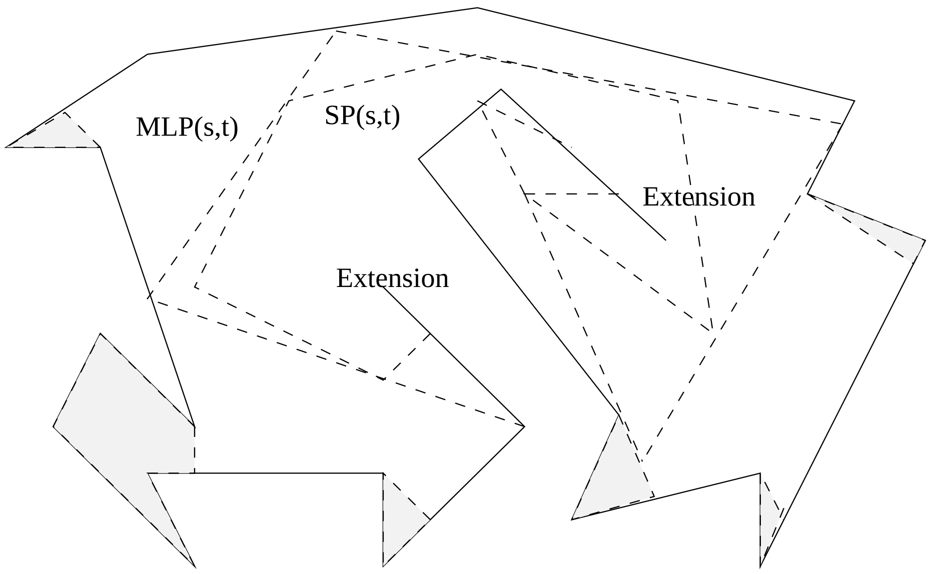


The edge bc of $SP(s,t)$ is an eave and a and d are extension points of bc on the boundary of P.

Any $MLP(s,t)$ must intersect ab, bc, cd. So, another $MLP(s,t)$ can be obtained by replacing $MLP(h,k)$ by hk. So, this altered $MLP(s,t)$ contains the eave bc.

There exists a $MLP(s,t)$ containing all eaves of $SP(s,t)$.

Compute MLP between the extensions of consecutive eaves and connected them using the subsegment containing eaves to construct $MLP(s,t)$.



MLP between the extensions of consecutive eaves is a convex greedy path.

Shortest path trees rooted at s and t can be used to remove some regions of P where convex greedy path does not enter.

1. V. Chandru, S. K. Ghosh, A. Maheshwari, V. T. Rajan and S. Saluja, *NC-Algorithms for minimum link path and related problems*, Journal of Algorithms, 19 (1995), 173-203. Running time: $O(\log n \log\log n)$ (with $O(n)$ processors in CREW PRAM model of computing).

Computing $MLP(s,t)$ in a polygon with holes

The algorithm of Mitchell et al. for computing $MLP(s,t)$ in a polygon with holes follows the same approach as that of Suri by computing the regions $V(1)$, $V(2)$,...Since computing $V(i)$ explicitly, for all i, is very costly, algorithm computes only the envelope of $V(i)$ for all i which is enough to compute $MLP(s,t)$.

Open Problem

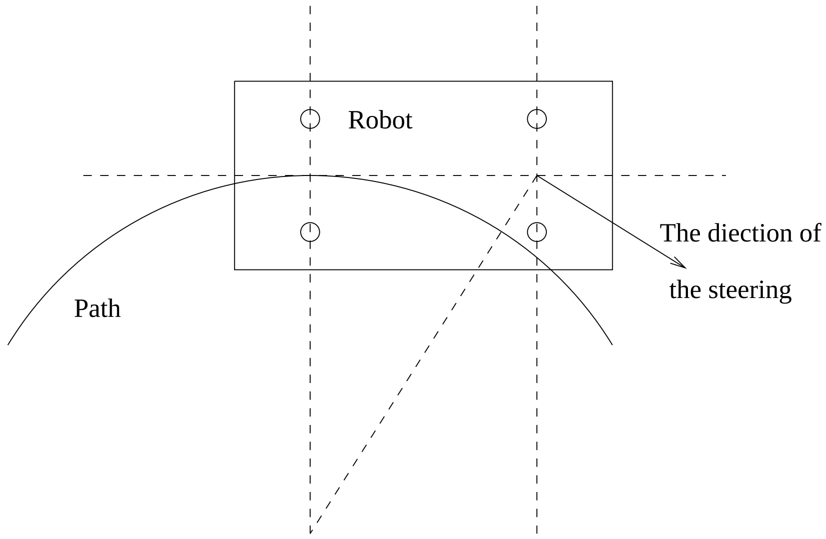
Can one compute $MLP(s,t)$ in a polygon with holes in sub-quadratic time?

1. J. Mitchell and G. Rote and G. Woeginger, *Minimum-link paths among obstacles in the plane*, Algorithmica, 8 (1992), 431-459. Running time: $O(Ea(n)\log^2 n)$ where $a(n)$ is the inverse of the Ackermann function.

2. A. Maheshwari, J-R. Sack and H. Djidjev, *Link distance problems*, Handbook in Computational Geometry (edited by J.-R Sack and J. Urrutia), Elsevier Science Publishers B.W., Chapter 12, 1999.

3. T. Asano, S. K. Ghosh and T. C. Shermer. Visibility in the plane. *Handbook in Computational Geometry (ed. J.-R. Sack and J. Urruta)*, Elsevier Science Publishers B. W., Chapter 19, 1999.

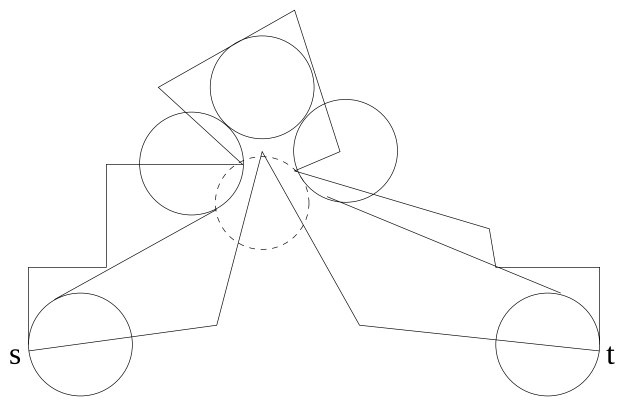
Non-holonomic Robot Motion Planning



A robot is said to be *non-holonomic* if some kinematics constraints (for example, velocity/acceleration bounds, curvature bounds) locally restricts the authorized directions for its velocity.

A typical example of a non-holonomic robot is that of a car: assuming no slipping of the wheels on the ground, the velocity of the midpoint between the two rear wheels of the car is always tangential to the path.

Bounded curvature path problem



Compute a path of minimum length inside a polygon between two given points s to t consisting of straight-line segments and circular arcs such that

(i) the radius of each circular arc is at least 1,

(ii) each segment on the path is the tangent between the two consecutive circular arcs on the path,

(iii) the given initial direction at s is tangent to the path at s, and

(iv) the given final direction at t is tangent to the path at t.

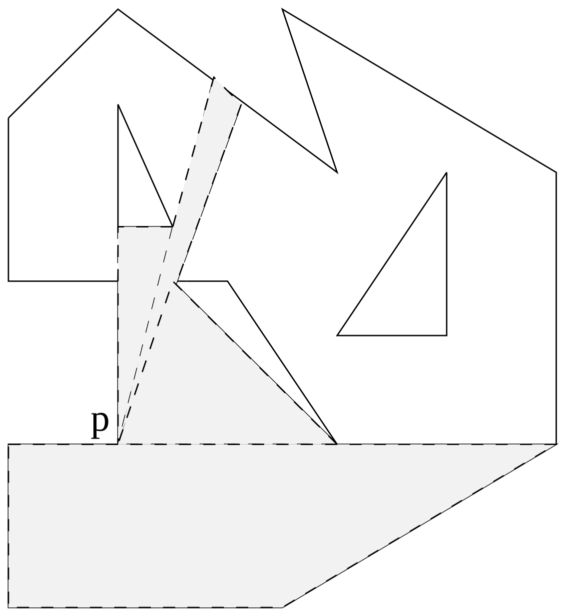
Open Problem

Above problem is open except when given polygon is a convex polygon without holes.

Algorithms for bounded curvature paths

1. S. Fortune and G. Wilfong, *Planning constrained motion*, STOC, pp. 445-459, 1988.
2. P. Jacobs and J. Canny, *Planning smooth paths for mobile robots*, Proc. of the IEEE Conf. on Robotics and Automation, pp. 2-7, 1989.
3. P.K. Agarwal, P. Raghavan, and H. Tamaki, *Motion Planning for a steering-constrained robot through moderate obstacles*, STOC, pp. 343-352, 1995.
4. J-D. Boissonnat and S. Lazard, *A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles*, In Proceedings of the Annual ACM Symposium on Computational Geometry, pages 242-251, 1996.
5. P.K. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri, and S. Whitesides, *Curvature-constrained shortest paths in a convex polygon*, In Proceedings of the 14th Annual ACM Symposium on Computational Geometry, pages 392-401, 1998. Running time: $O(n^2 \log n)$.
6. J-D. Boissonnat, S. K. Ghosh, T. Kavitha and S. Lazard, *An algorithm for computing a convex and simple path of bounded curvature in a simple polygon*, submitted for publication. Running time: $O(n^4)$.

Exploring an unknown polygon: Continuous visibility



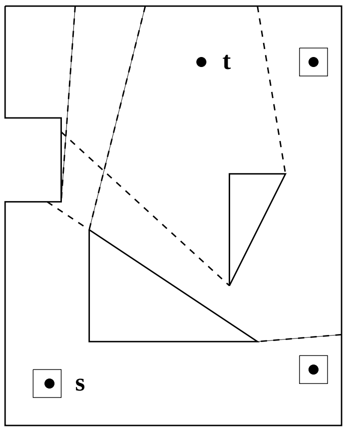
Suppose the polygon P is not known apriori and the point robot can compute the visibility polygon of P from its current position using visual sensors. The robot wants to see all points of P with minimum cost. Cost can be the length or the number of links in the path that the robot has traveled starting from its initial position.

Efficiency of the on-line algorithm is defined in terms of *Competitive ratio*.

$$Competitive\, ratio = \frac{cost\ of\ the\ on-line\ algorithm}{cost\ of\ the\ off-line\ algorithm}$$

.

Algorithms for exploring in unknown environment



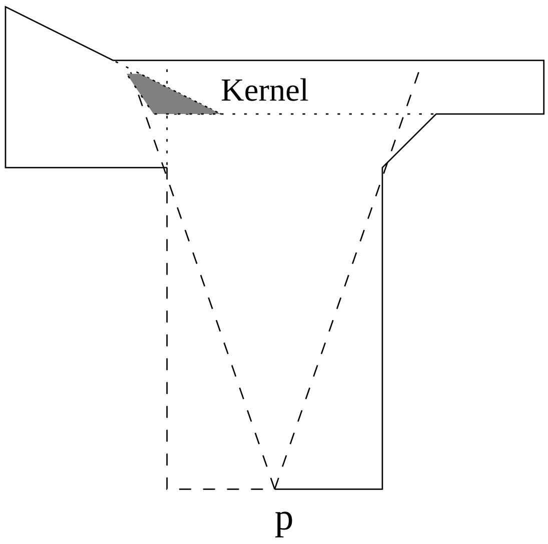
1. A. Blum and P. Raghavan and B. Schieber, *Navigating in unfamiliar geometric terrain*, SIAM Journal on Computing, 26 (1997), 110-137.

2. K. Chan and T. W. Lam, *An on-line algorithm for navigating in an unknown environment*, International Journal of Computational Geometry and Applications, 3 (1993), 227-244.

3. X. Deng and T. Kameda and C. Papadimitriou, *How to learn an unknown environment I: The rectilinear case*, Journal of ACM, 45 (1998), 215-245.

4. F. Hoffmann, C. Icking, R. Klein, K. Kriegel, *A competitive strategy for learning a polygon*, In Proceedings of the eighth ACM-SIAM Symposium on Discrete Algorithms, Pages 166-174, 1997.

Searching for the kernel

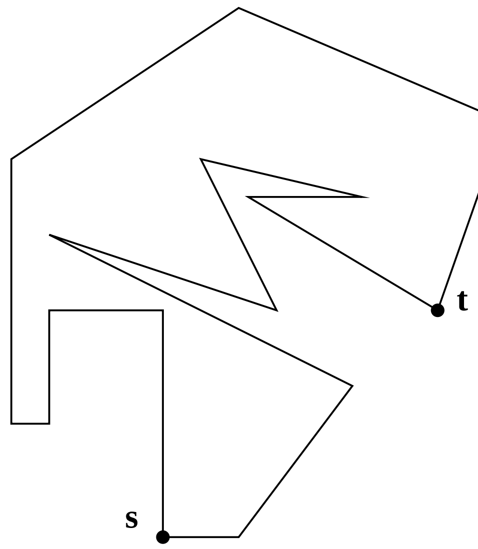


The *kernel of P* is the set of all points of P from which entire P is visible. The problem is to design a competitive strategy to walk into the kernel of P.

1. C. Icking and R. Klein, *Searching for the Kernel of a Polygon—A Competitive Strategy*, In Proceedings of the Annual ACM Symposium on Computational Geometry, pages 258-266, 1995. Competitive ratio:5.331.
2. J.-H. Lee, C.-S. Shin, J.-H. Kim, S. Y. Shin and K.-Y. Chwa, *New competitive strategies for searching in unknown star-shaped polygons*, In Proceedings of the Annual ACM Symposium on Computational Geometry, pages 427-432, 1997. Competitive ratio: 3.828.

Open Problem: The problem is open in link metric.

Searching for a target in a street



A *street* (also called *LR-polygon*) is a polygon for which two boundary chains from start to target are mutually weakly visible.

Entire street is visible from any path connecting s and t.

1. G. Das, P. J. Heffernan and G. Narasimhan, *LR-visibility in polygons*, Computational Geometry: Theory and Applications, 7 (1997), 37-57.
2. B. Bhattacharya and S. K. Ghosh, *Characterizing LR-visibility polygons and related problems*, Computational Geometry: Theory and Applications, 2001 (to appear).

Algorithms for searching in a street

1. R. Klein, *Walking an unknown street with bounded detour*, Computational Geometry: Theory and Applications, 1 (1992), 325-351. Competitive ratio: 5.72.

2. J. Kleinberg, *On line search in a simple polygon*, In Proceedings of the fifth ACM-SIAM Symposium on Discrete Algorithms, Pages 8-15, 1994. Competitive ratio: 2.83.

3. C. Icking, R. Klein and E. Langetepe, *An optimal competitive strategy for walking in streets*, LNCS vol: 1563, Springer, Pages 110-120, 1999. Competitive ratio: 1.41.

4. A. Datta and C. Icking, *Competitive searching in a generalized street*, Computational Geometry: Theory and Applications, 13 (1999), 109-120. Competitive ratio: 9.06.

5. S. K. Ghosh and S. Saluja, *Optimal on-line algorithms for walking with minimum number of turns in unknown streets*, Computational Geometry: Theory and Applications, 8 (1997), 241-266. Competitive ratio: 2.

6. C. Icking and R. Klein, *Competitive strategies for autonomous systems*, In Modelling and Planning for Sensor Based Intelligent Robot Systems (H. Bunke. et al., editor), World Scientific, pages 23-40, 1995.

Exploring an unknown polygon: Discrete visibility

Most on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?

1. Autonomous robots can only carry a limited amount of on-board computing capability. At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming. The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot's trajectory.

2. For good visibility, the robot's camera will typically be mounted on a mast. Such devices vibrate during the robot's movement, and hence for good precision the camera must be stationary while computing visibility polygon.

It seems feasible to compute visibility polygons only at a discrete number of points.

The computational geometry literature has typically assumed that the cost associated with a robot's physical movement dominates all other associated costs. Is this assumption reasonable?

Exploration cost

Each move will have two associated costs as follows.

1. There is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, r, during a move, the total time required for motion will be $r\ D$, where D is the total path length.

2. In an exploratory process each move must be planned immediately prior to the move so as to account for the most recently acquired geometric information. The robot will be stationary during this process, which we assume to take time t_M. Since the robot is stationary during each sensing operation, we assume that it takes time t_S.

Let N_M and N_S be respectively the number of moves and the number of sensor operations. Hence the total cost required to explore P:

$$T(P) = t_M\ N_M + t_S\ N_S + r\ D \tag{1}$$

Now, $(t_M\ N_M + t_S\ N_S)$ can be viewed as the time required for computing and maintaining visibility polygons and it is indeed a significant fraction of $T(P)$ because computer vision algorithms consume significant time on modest computers in a relatively cluttered environment.

The criteria for minimizing the cost for robotic exploration is now to reduce the number of visibility polygons.

Exploration algorithm of Ghosh and Burdick

Step 1: $i := 1$; $T(P) := \emptyset$; $S := \emptyset$; Let p_1 denote the starting position of the robot and $V(P, p_i)$ denote the visibility polygon of P from a point p_i.

Step 2: Compute $V(P, p_i)$; $T(P) := T(P) \cup T'(P)$ where $T'(P)$ is the triangulation of $V(P, p_i)$; $S = S \cup p_i$;

Step 3: While $V(P, p_i) - T(P) = \emptyset$ and $i \neq 0$ then $i := i - 1$;

Step 4: If $i = 0$ then goto Step 7;

Step 5: If $V(P, p_i) - T(P) \neq \emptyset$ then choose a point z on any constructed of $V(P, p_i)$ lying outside $T(P)$;

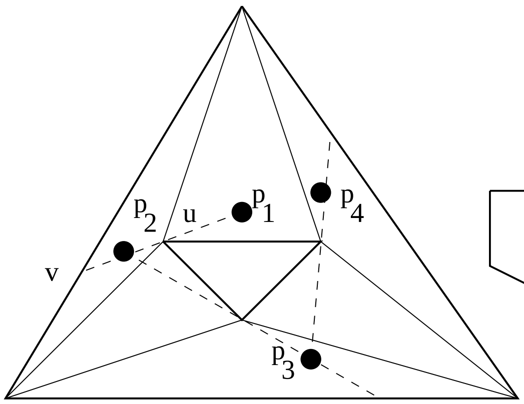
Step 6: $i := i + 1$; $p_i := z$; goto Step 2;

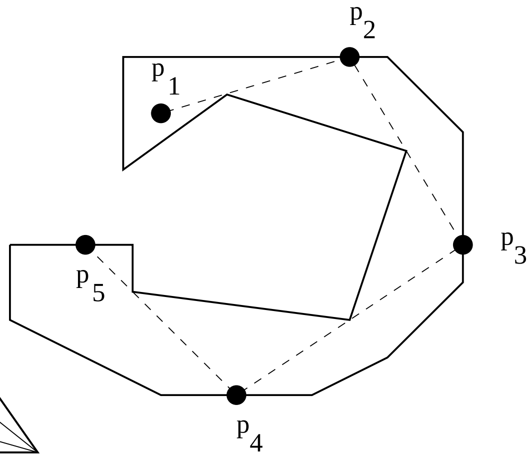
Step 7: Stop.

1. S. K. Ghosh and J. W. Burdick, *An on-line algorithm for exploring an unknown polygonal environment by a point robot*, In Proceedings of the Ninth Canadian Conference on Computational Geometry, pp. 100-105, 1997.

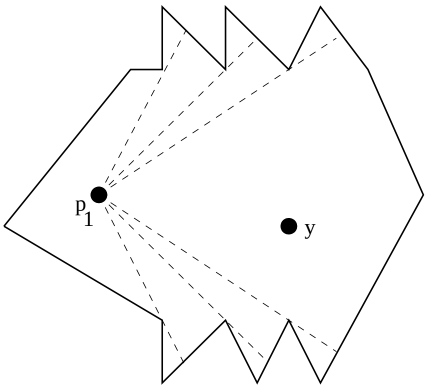
2. S. K. Ghosh and J. W. Burdick, *Exploring an unknown polygon environment with a sensor based strategy*, submitted for publication.

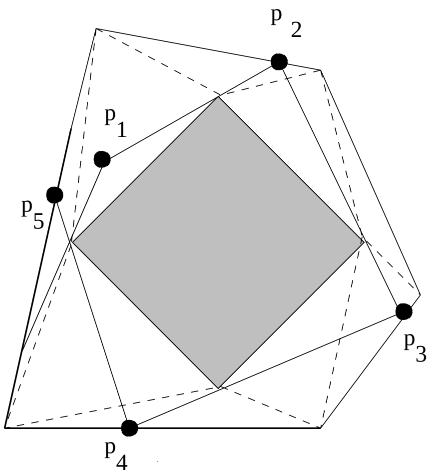
Competitive ratio





The algorithm needs $r+1$ views.





Competitive ratio is $(r+1)/2$, where r denotes the number of reflex vertices of P.

Open Problem: Can one improve the bound?

Exploring an unknown polygon: Bounded visibility

Computer vision range sensors or algorithms, such as stereo or structured light range finder, can reliably compute the 3D scene locations only up to a depth R. The reliability of depth estimates is inversely related to the distance from the camera. Thus, the range measurements from a vision sensor for objects that are far away are not at all reliable.

Therefore, the portion of the boundary of a polygonal environment within the range distance R is only considered to be visible from the camera of the robot.

The maximum number of views needed to explore the unknown polygon P with obstacles of size n is bounded by

$$\left\lfloor \frac{2\times \text{Perimeter}(P)}{R} \right\rfloor + \left\lfloor \frac{2\times \text{Area}(P)}{R^2} \right\rfloor + 3n.$$

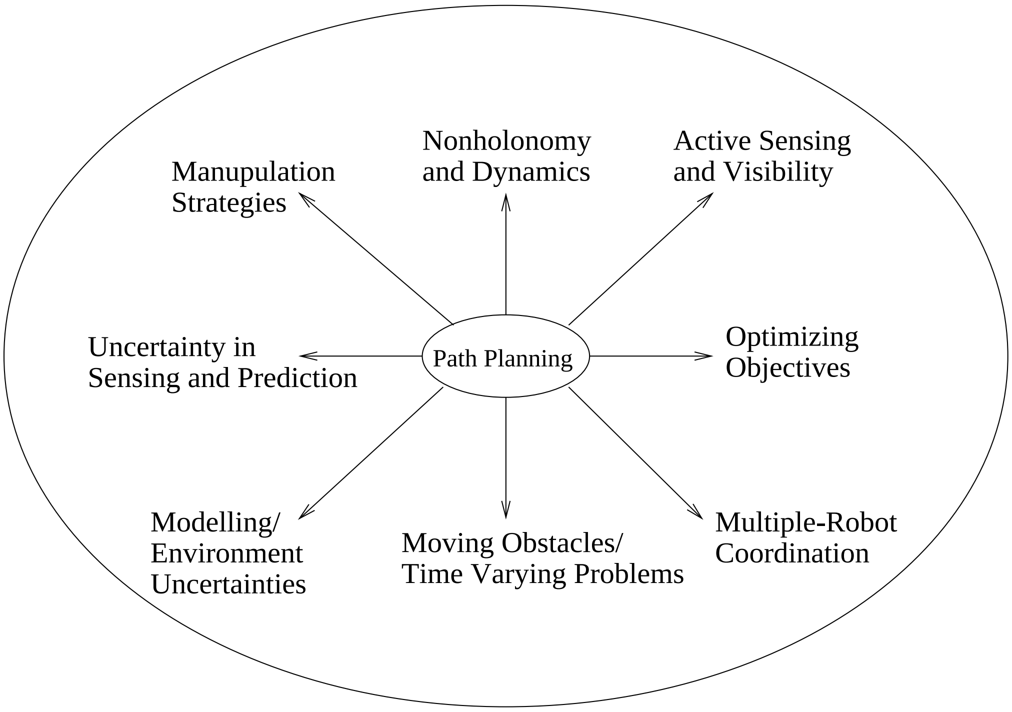
The competitive ratio of the algorithm is

$$6\pi + \frac{4r+3n}{max(\left\lceil \frac{\text{Area}(P)}{\pi R^2} \right\rceil, \left\lceil \frac{\text{Perimeter}(P) - 2Rr}{2\pi R}) \right\rceil}.$$

1. A. Bhattacharya, S. K. Ghosh and S. Sarkar, *Exploring an Unknown Polygonal Environment with Bounded Visibility*, manuscript, 2000.

Concluding remarks

In this talk, we have reviewed a few algorithms for robot path planning in the plane which are based on visibility computations and have suggested a few open problems.



Basic path planning problems.

.