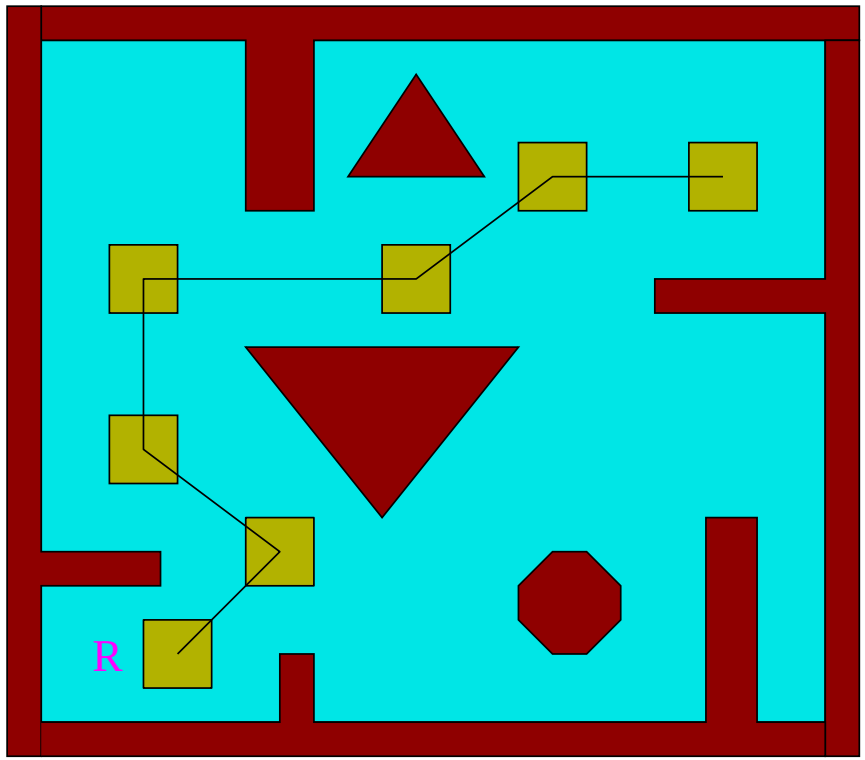
Robot Path Planning: Off-line and On-line Algorithms

Subir Kumar Ghosh

School of Technology & Computer Science
Tata Institute of Fundamental Research
Mumbai 400005, India
ghosh@tifr.res.in

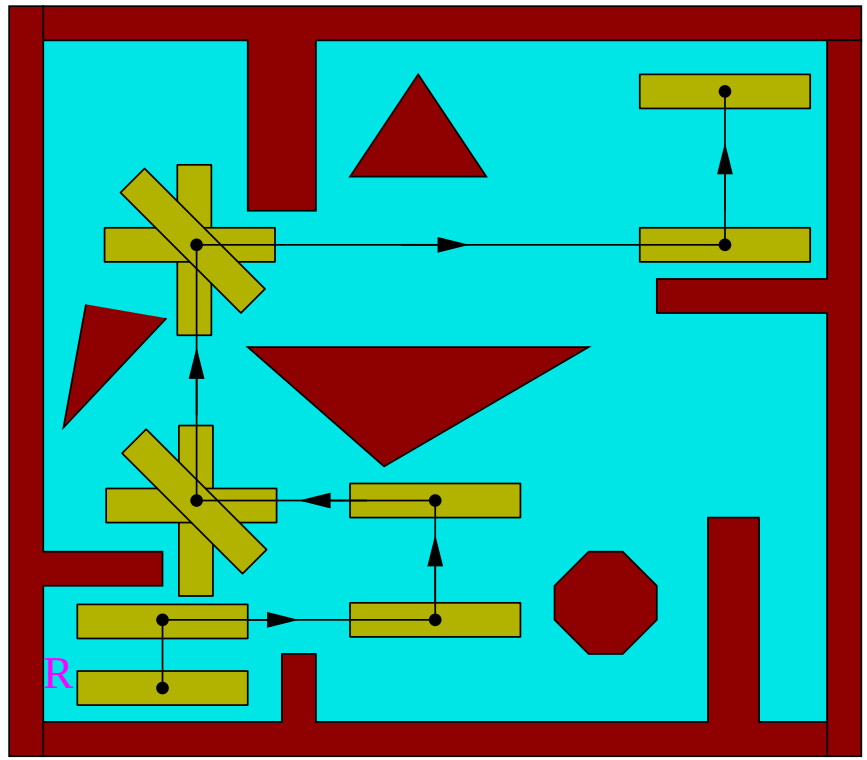
Collision-free Path

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



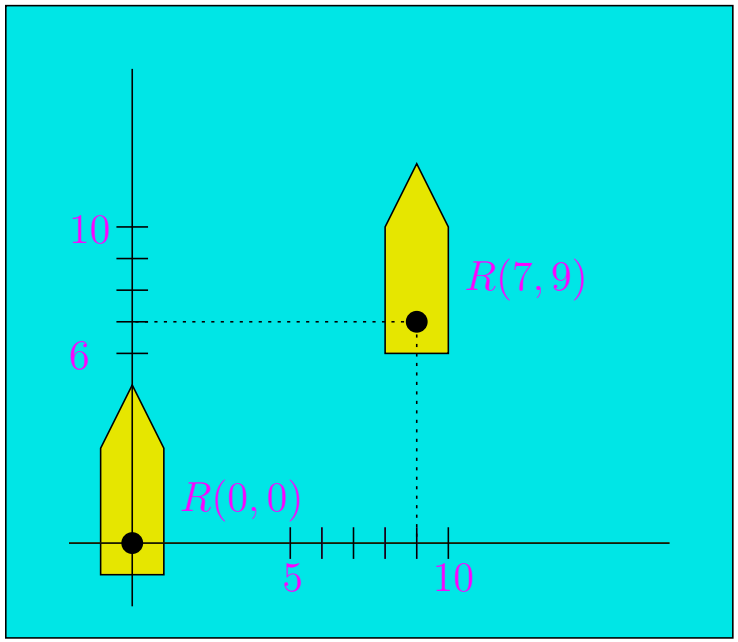
1. J-C Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

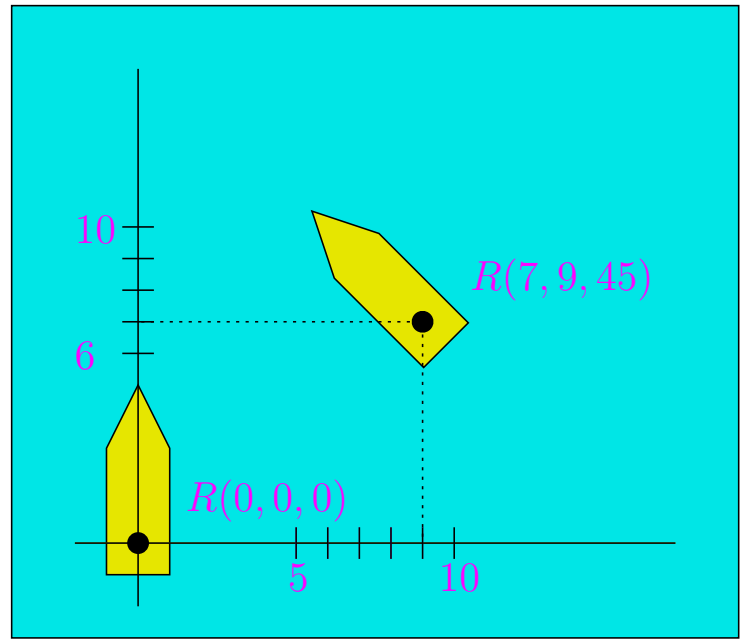
The path of a robot can be viewed as a sequence of translations and rotations from starting position to the destination in the work space.



1. H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, 2005.

Placement of a robot in the work space





By translation, $R(0,0)$ is first placed at $R(7,9)$, and then $R(7,9,0)$ is rotated by 45° to $R(7,9,45)$.

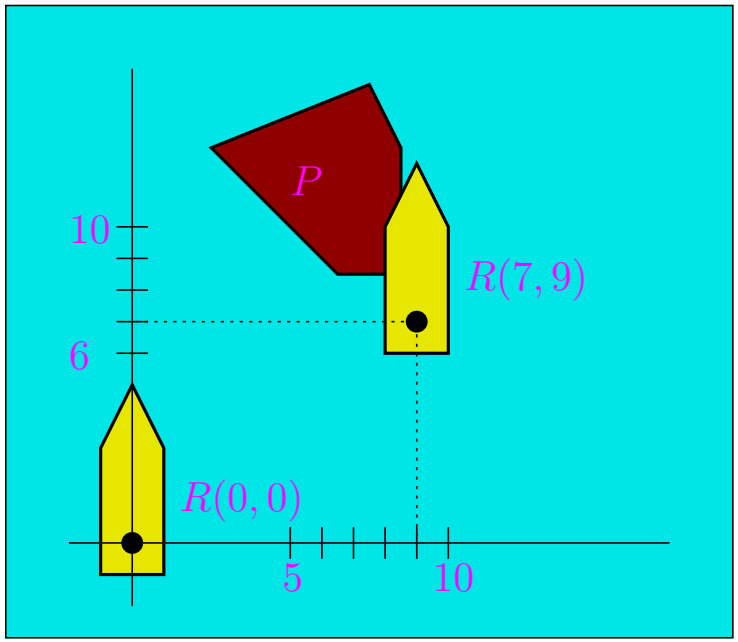
1. M. de Berg, M. Van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, 1997.

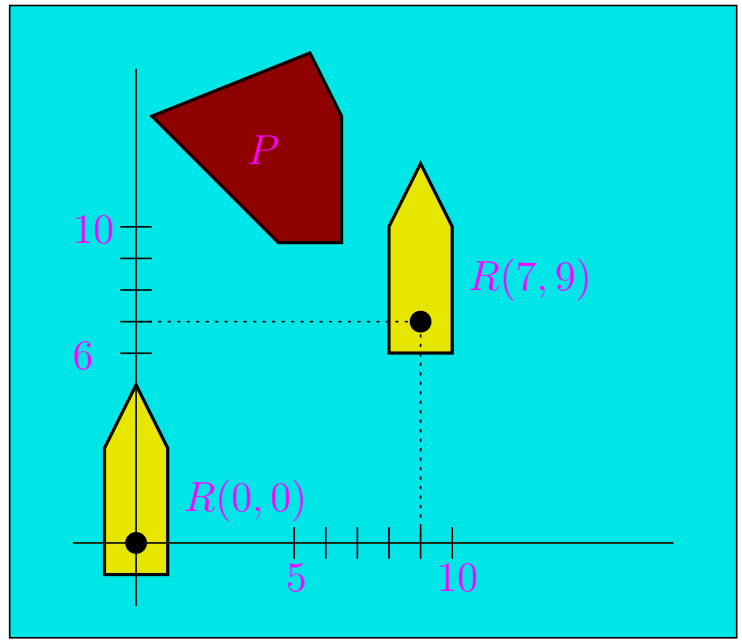
Degrees of freedom

- A placement of a robot is specified by a set of parameters that corresponds to the number of *degrees of freedom* of the robot.
- This number is 2 if planar robots can only translate.
- This number is 3 if planar robots can both translate and rotate.
- In three dimensional space, a robot that can only translate has 3 degrees of freedom.
- In three dimensional space, a robot that is free to translate and rotate has 6 degrees of freedom.

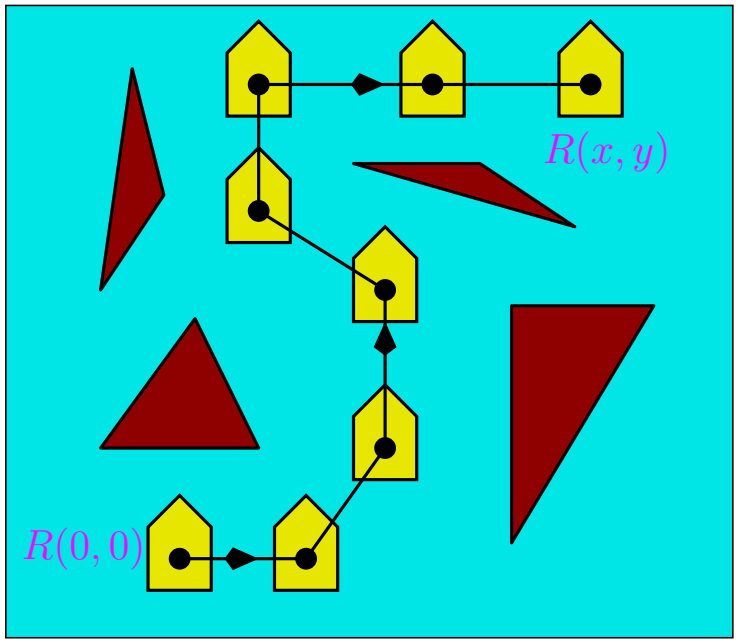
Configuration space

- The parameter space of a robot R is called *configuration space*.
- A point p in the configuration space corresponds to a certain placement $R(p)$ of the robot R in the work space.
- The configuration space of a translating robot R in the plane is the two dimensional Euclidean plane, and therefore identical to the work space.
- The configuration space of a translating and rotating robot R in the plane is the three dimensional space $\mathbb{R}^2 x[0 : 360)$, and therefore is not the Euclidean 3-dimensional space.

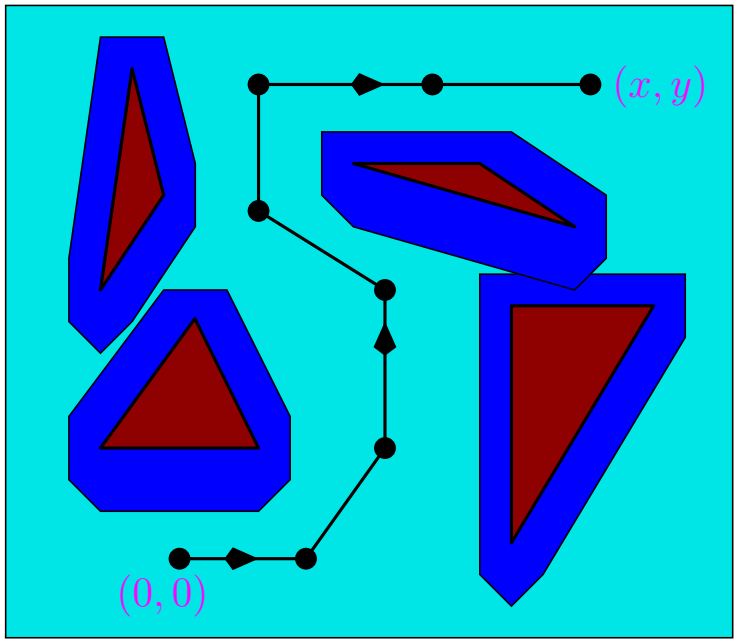




- If a placement of a robot R intersects an obstacle P in the work space, then the corresponding point lies in the *forbidden configuration space*.
- Otherwise, the corresponding point lies in the *free configuration space*.
- Every collision free path of a robot R lies in the free configuration space.



Work Space

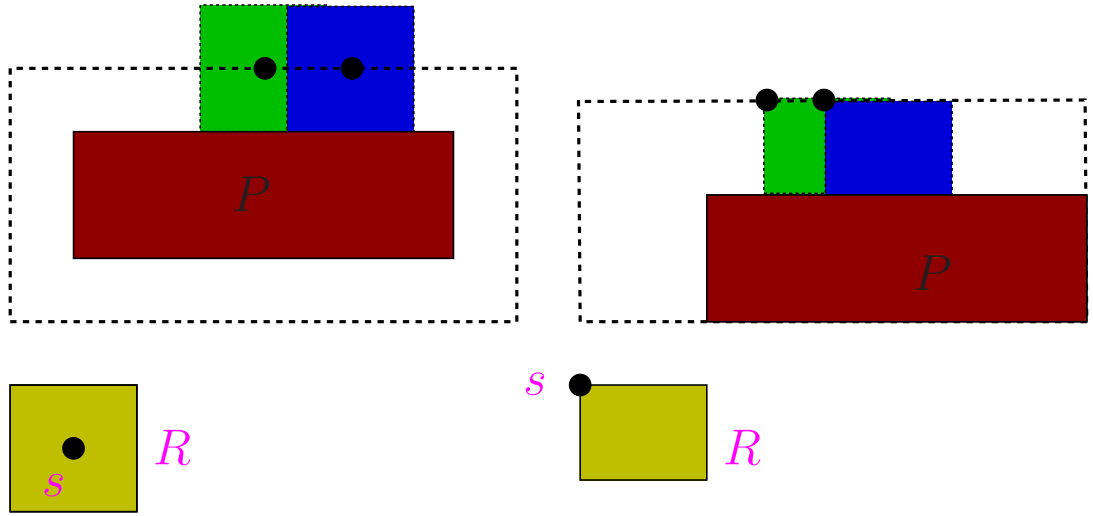


Configuration Space

- A collision-free path of a planar robot under translation in the work space is mapped to a path of a point robot in the free configuration space.
- Inflated obstacles in the configuration space are called *configuration-space obstacles* or *C-obstacles*.

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 C – *obstacles* under translation



- The *C-obstacles* of a obstacle P for a robot R under translation is the set of all points in the configuration space such that the corresponding placements of R in the work space intersects P.
- If R is slided along the boundary of the obstacle P, then the boundary of the *C-obstacle* is the locus of the reference point of R.
- The *C-obstacle* can be computed using the Minkowski sum of P and R.

Minkowski sum

- The Minkowski sum of two sets $S_1 \subset \mathbb{R}^2$ and $S_2 \subset \mathbb{R}^2$ is defined as $S_1 \oplus S_2 := \{p + q : p \in S_1, q \in S_2\}$, where $p + q$ denotes the vector sums of the vectors p and q, i.e., if $p = (p_x, p_y)$ and $q = (q_x, q_y)$, then we have $p + q := (p_x + q_x, p_y + q_y)$.
- The $C - obstacle$ of P for a planar robot R under translation is $P \oplus (-R(0, 0))$.
- The Minkowski sum of two convex polygons with n and m edges can be computed in $O(n + m)$ time.

Algorithms for Minkowski operations

1. L. Guibas, L. Ramshaw and J. Stolfi, *A kinetic framework for computational geometry*, FOCS, Pages 100-111, 1983.
2. M. de Berg, M. Van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, 1997.
3. P.K. Ghosh, *A solution of polygon containment, spatial planning, and other related problems using Minkowski operations*, Computer Vision, Graphics and Image Processing, 49 (1990), 1-35.
4. P.K. Ghosh, *A unified computational framework for Minkowski operations*, Computers and Graphics, 17 (1993), 357-378.
5. R. V. Benson, *Euclidean Geometry and Convexity*, McGraw-Hill, New York, 1966.

An algorithm for computing a collision free path

Step 1: Take a point s of the planar robot R as the reference point.

Step 2: Using Minkowski sums, compute C-obstacles of the set of obstacles $Q = \{P_1, P_2, \ldots, P_k\}$ given in the work space.

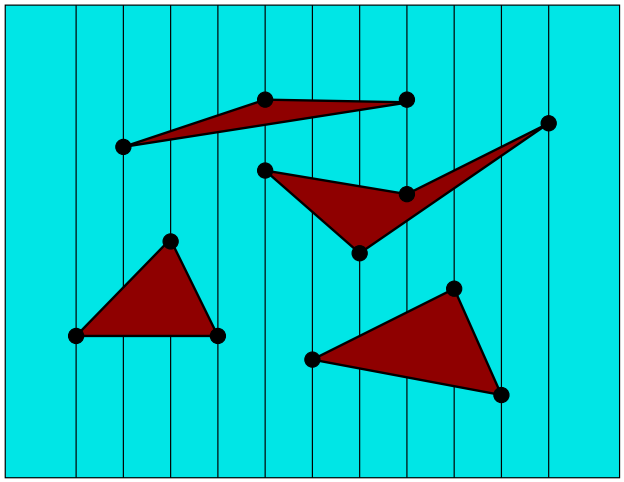
Step 3: Compute the free configuration space by constructing trapezoids.

Step 4: Construct a road map through the free configuration space using trapezoids.

Step 5: Using the road map, compute a path from the starting position $(0, 0)$ of s to the destination point (x, y).

1. K. Kedem, R. Livne1, J. Pach and M. Sharir, *On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles*, Discrete and Computational Geometry, 1 (1986), 59-71.
2. M. de Berg, M. Van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, 1997.

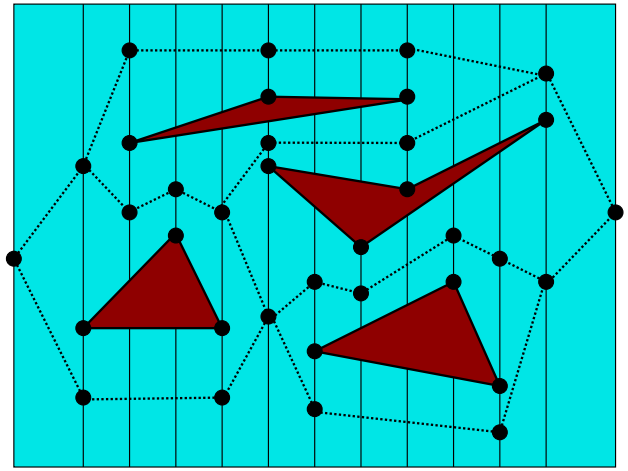
Trapezoid decomposition



Configuration Space

- Draw vertical lines through each vertex of all obstacles.
- Remove the portions of the vertical segments that are intersecting obstacles.
- These segments decompose the free configuration space into trapezoid.
- Trapezoid decomposition can be done in $O(n \log n)$ time.

Road map



Configuration Space

- For each trapezoid, take the midpoints of two vertical segments as nodes, and connect them by an edge.
- This graph gives a road map in the free configuration space.
- Locate the starting point in the trapezoid containing it, and connect it to a node in the trapezoid.
- Similarly, locate the destination point in the trapezoid containing it, and connect it to a node in the trapezoid.
- The road map can be constructed in $O(n)$ time.

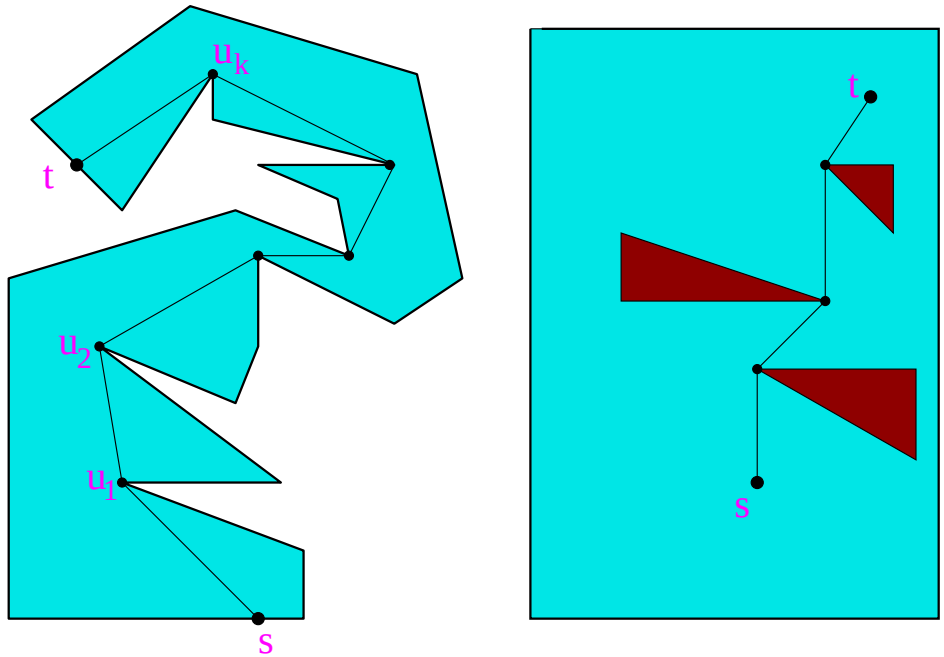
Computing shorter path

- In the previous algorithm, no claim was made on the type of a collision-free path; the path can be very long or it can have a large number of turns.
- In general, if a collision-free path is long, robot naturally takes longer time to reach destination.
- It is, therefore, desirable to compute the collision-free path of minimum length.
- We use the visibility graph of the configuration space as the road map for computing the Euclidean shortest path of the robot from the given stating point to the destination.

1. S. K. Ghosh, *Visibility Algorithms in the Plane*, Cambridge University Press, Cambridge, UK, 2007.

Computing Euclidean shortest paths

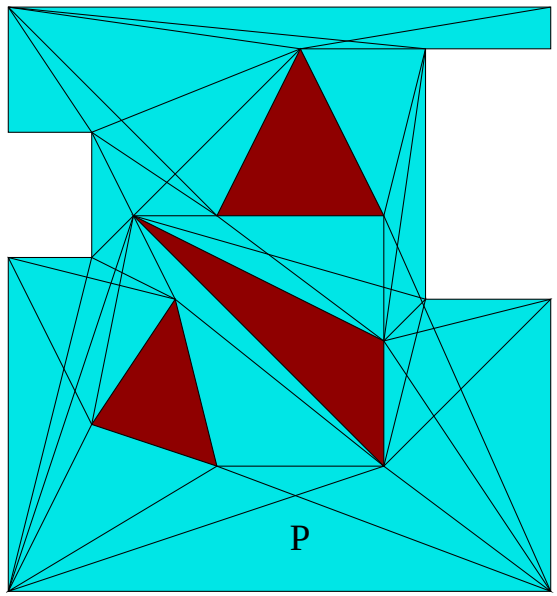
The *Euclidean shortest path* (denoted as $SP(s, t)$) between two points s and t in a polygon P is the path of smallest length between s and t lying totally inside P.



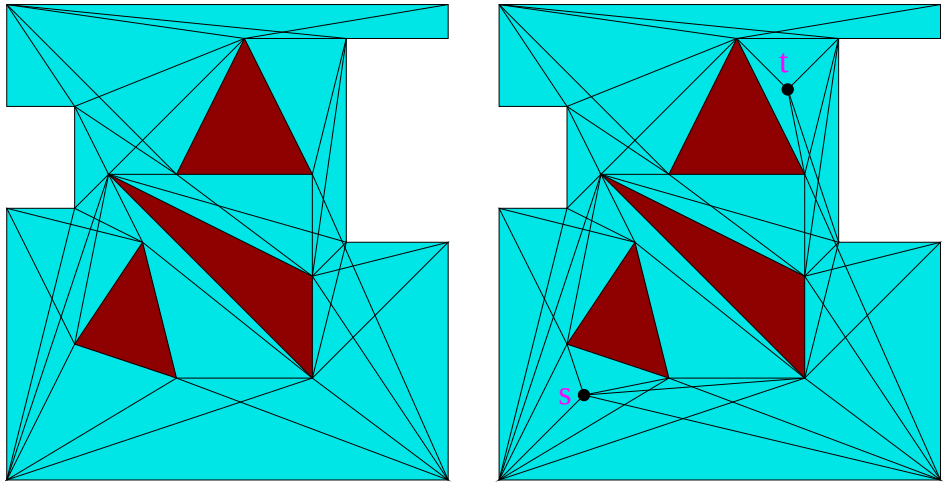
Let $SP(s, t) = (s, u_1, u_2, \ldots, u_k, t)$. Then, (i) $SP(s, t)$ is a simple path, (ii) $u_1, u_2, \ldots, u_k$ are vertices of P and (iii) for all i, u_i and u_{i+1} are mutually visible in P. $SP(s, t)$ is *outward convex* at every vertex on the path.

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

Two vertices of a polygon (with or without polygonal holes) is said to be *visible* if the line segment joining them lie totally inside the polygon.



The visibility graph of a polygon P is a graph whose vertex set consists of the vertices of P and whose edges are visible pairs of vertices.



- Compute the visibility graph of P.
- Connect s to vertices of P that are visible from s.
- Connect t to vertices of P that are visible from t.
- Assign the length of each visible pair as an weight on the corresponding edge in the visibility graph.
- Compute $SP(s, t)$ in the weighted visibility graph.

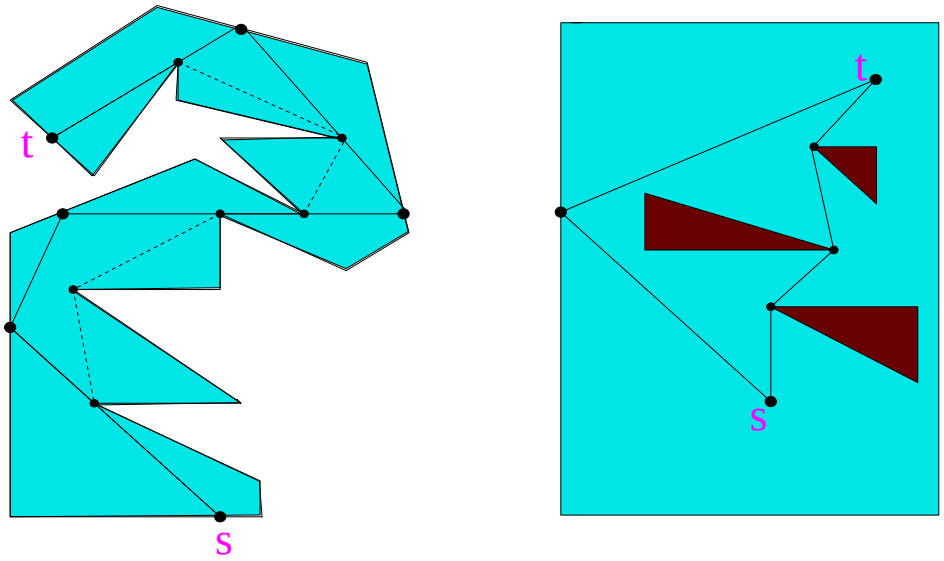
1. M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, Journal of ACM, 34 (1987), 596-615. Running time: $O(n \log n + E)$, where E is the number of edges in the visibility graph.

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. M. Overmars and E. Welzl, *New methods for constructing visibility graphs*, in Proc. 4th ACM Symposium on Computational Geometry, 164-171, 1988. Running time: $O(E \log n)$, where E is the number of edges in the visibility graph.
7. 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: $O(E)$.
8. M. Pocchiola and G. Vegter, *Topologically sweeping visibility complexes via pseudo-triangulations*, Discrete and Computational Geometry, 16(1996), 419–453. Running time: $O(n \log n + E)$. Space: $O(n)$.
9. S. Kapoor and S. N. Maheshwari, *Efficiently Constructing the Visibility Graph of a Simple Polygon with Obstacles*, SIAM Journal on Computing, 30(2000), 847-871. Running time: $O(h \log n + T + E)$, where T is the time for triangulation and h is the number of holes.

Computing minimum link paths



A minimum link path connecting two points s and t inside a polygon P with or without holes (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: $O(n)$.

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 $MLP(s,t)$.

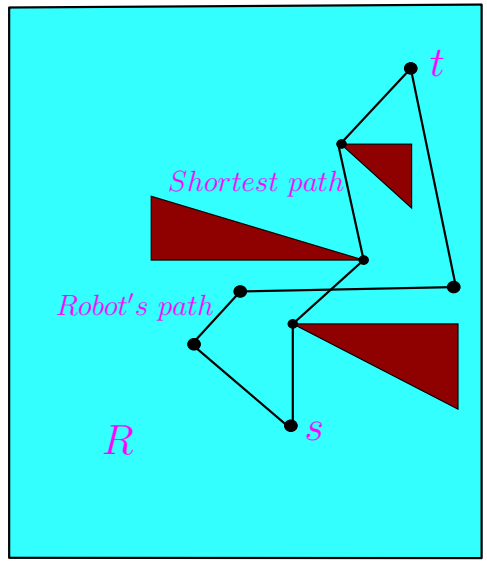
Open Problem: Can $MLP(s,t)$ be computed in a polygon with holes in sub-quadratic time?

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, pp. 519-558, 2000.
3. S. K. Ghosh, *Visibility Algorithms in the Plane*, Cambridge University Press, 2007.

Online algorithms

- Suppose, a robot does not have the complete knowledge of the geometry of R apriori.
- The robot also does not know the location of the target t, but the target can be recognized by the robot.
- In such a situation, the robot is asked to reach t from its starting position s using its sensory input provided by acoustic, visual, or tactile sensors of its on-board sensor system.
- The problem here is to design an efficient *online algorithm* which a robot can use to search for the target t.
- Observe that any such algorithm is 'online' in the sense that decisions must be made based only on what the robot has received input so far from its sensor system.

Efficiency of online algorithms



One of the difficulties in working with incomplete information is that the path cannot be pre-planned and therefore, its global optimality can hardly be achieved.

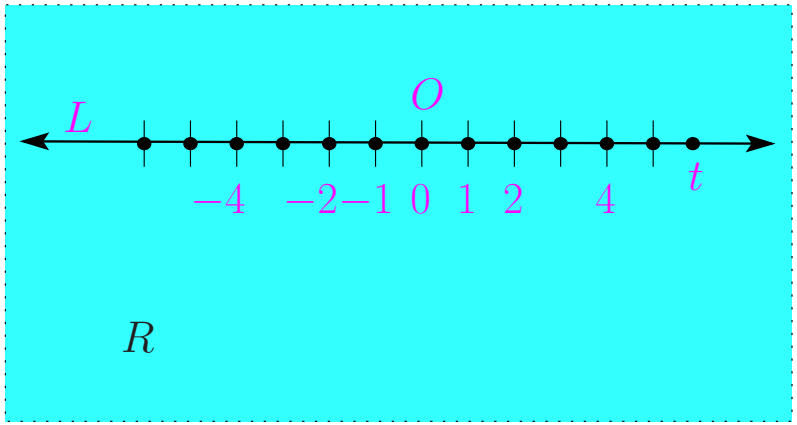
Instead, one can judge the online algorithm performance based on how it stands with respect to other existing or theoretically feasible algorithms.

The efficiency of online algorithms for searching and exploration algorithms is generally measured using their competitive ratios.

$$\textit{Competitive ratio} = \frac{\textit{Cost of the online algorithm}}{\textit{Cost of an optimal offline algorithm}}$$

1. P. Berman, *On-line searching and navigation*, Lecture Notes in Computer Science 1442, pp. 232-241, Springer, 1996.
2. D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, Communication of ACM, 28: 202-208, 1985.

Searching for a target on a line

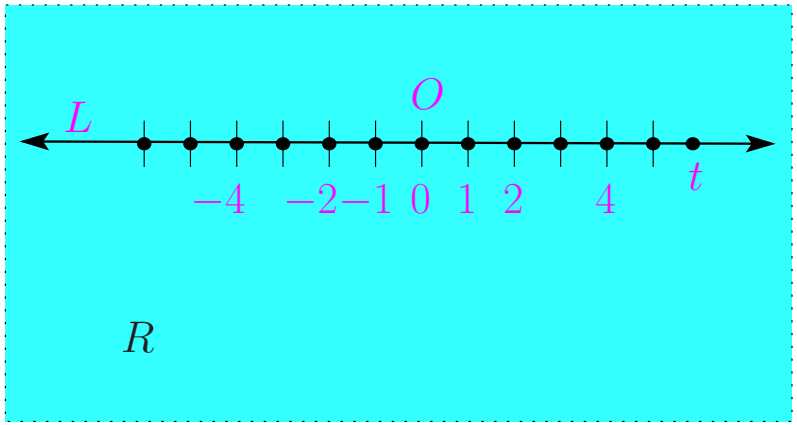


- Suppose, the target point t is placed on a line L in an unknown location.
- Starting from a given position O on L, the problem is to design an online algorithm for a point robot for locating t.
- It is assumed that the robot can detect t if it stands on top of t or reaches t.
- The problem may be viewed as an autonomous robot is facing a very long wall and it wants go to the other side of the wall through a door on the wall but it does not known whether the door is located to the left or right of its current position.

- Suppose the robot knows that t is located exactly d distance away from O.
- Then the robot first walks d distance to the right.
- If t is not found, then the robot returns to O and then walks d distance to the left.
- So, the competitive ratio of this straightforward on-line algorithm is 3.

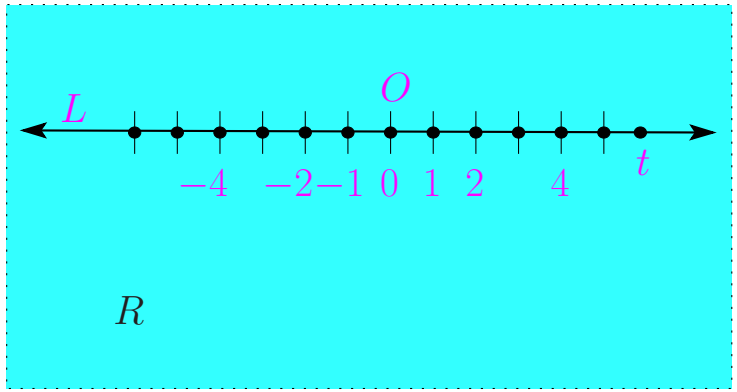
What is the competitive ratio of the search if d is not known apriori?

Alternate walk



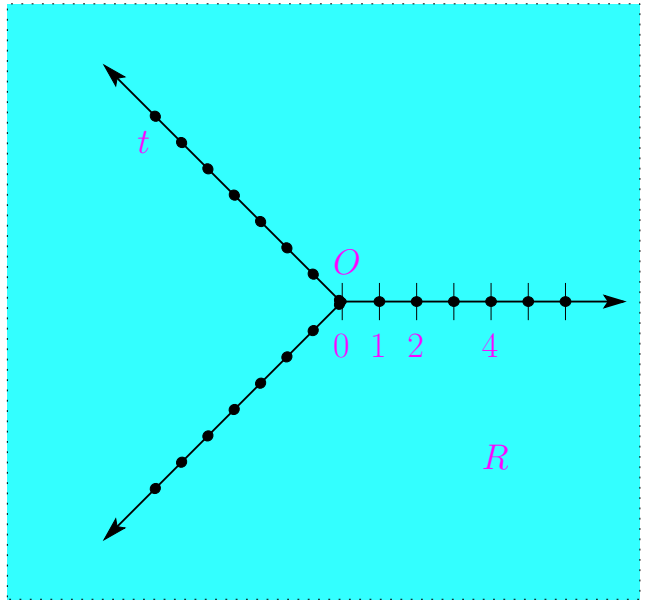
- The robot walks one unit to the right along L. If t is not found, then it returns to its starting point O.
- In the next step, the robot walks two units to the left of O along L. If t is not found again, the robot returns to O.
- In the next step, the robot walks four units to the right along L and if it is again unsuccessful to locate t, it returns to O.
- After some steps, the robot locates t.

The process of doubling the length is known as *doubling strategy*.



- Assume that t is located at a distance d from the origin on the positive axis.
- Assume that $2^{k-1} < d \leq 2^{k+1}$ for some k.
- The total distance traveled during the alternative walk is $(2.1 + 2.|-2| + 2.4 + 2.|-8| + \ldots + 2.2^{k-1} + 2.|-2^k| + d = 2.2^{k+1} + d)$.
- If the location of t is known apriori, then it is a straight walk of length d from the origin to t.
- So, the competitive ratio of the alternate walk is $(2.2^{k+1} + d)/d = 1 + 2.2^{k+1}/d$ which is at most $1 + (2.2^{k+1}/2^{k-1}) = 9$.

Searching for a target on m rays



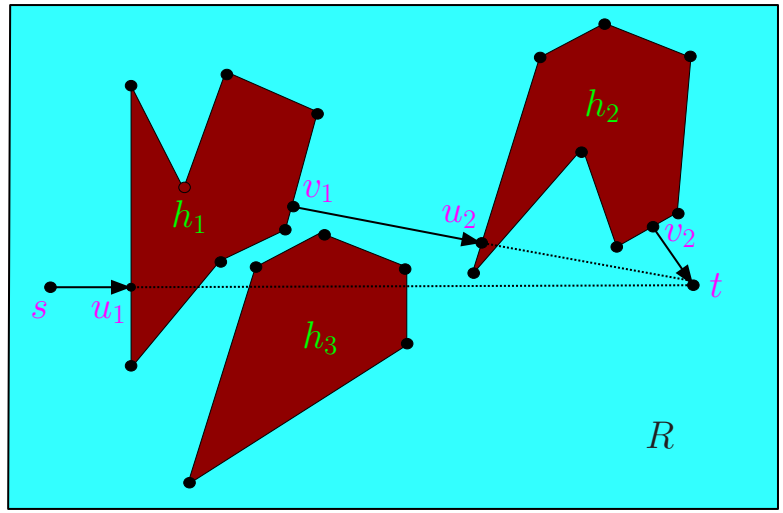
A beautiful young cow Ariadne is at the entrance of a simple labyrinth which branches in $m \geq 2$ corridors. She knows that the handsome Minotaur is waiting somewhere in the labyrinth. What is the best searching strategy for Ariadne to locate Minotaur?

1. S. Gal, *Minimax solutions for linear search problems*, SIAM Journal on Applied Mathematics, 27:17-30, 1974.
2. S. Gal, *Search games*, Academic Press, New York, 1980.

- Visit $m \geq 2$ rays in a cyclic order starting with an initial walk of length one.
- Increase the length of the walk each time by a factor of $m/(m-1)$ till t is located.
- This strategy gives the competitive ratio of $1 + 2m^m/(m-1)^{m-1}$, which is optimal.

1. R. A. Baeza-Yates, J. C. Culberson and G. J. E. Rawlins, *Searching in the plane*, Information and Computation, 106:234-252, 1993.
2. A. Eubeler, R. Fleischer, T. Kamphans, R. Klein, E. Langetepe and G. Trippen, *Competitive online searching for a ray in the plane*, Robot Navigation, Schloss Dagstuhl, Germany, 2006.
3. E. Langetepe, *On the optimality of spiral search*, Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, 2010.

Searching for a target in an unknown region

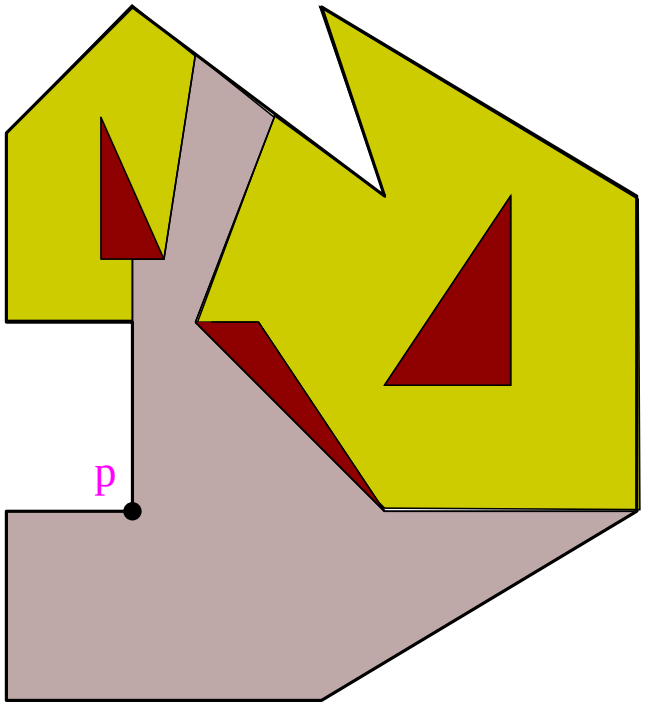


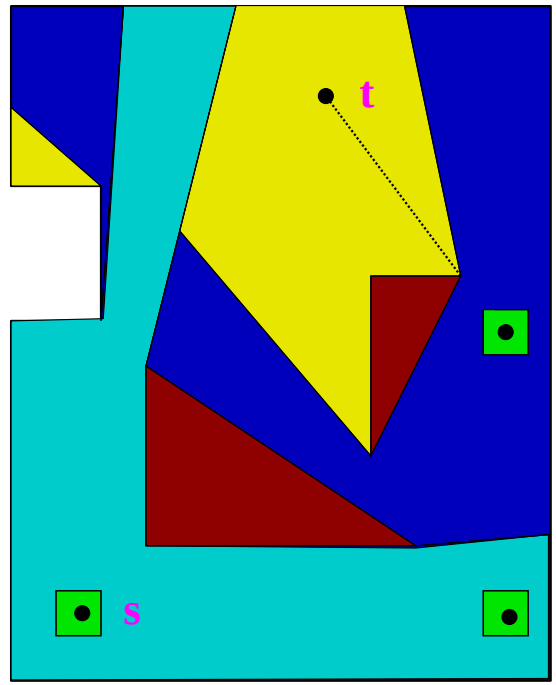
- Assume that the point robot knows the exact location of t but does not know the positions of unknown polygonal obstacles $h_1, h_2, \ldots, h_k$.
- The robot starts from s, and moves towards t following the segment st till the robot detects by its tactile sensor that it has hit a polygonal obstacle (say, h_i) at a some point u_i.
- Then the robot goes around the boundary of h_i to locate the boundary point of h_i (say, v_i) which is closest to t.
- Then the robots moves from u_i to v_i following the shorter of the two paths from u_i to v_i along the boundary of h_i .

- Then the robots moves from u_i to v_i following the shorter of the two paths from u_i to v_i along the boundary of h_i .
- Treating v_i as s, the robot repeats the same process of moving towards t following the segment $v_i t$ till t is reached.
- The length of the path traversed by the robot is bounded by the length of st and 1.5 times the perimeters of those polygonal obstacles that are hit by the robot.

1. V. Lumelsky and A. Stepanov, *Dynamic path planning for a mobile automaton with limited information on the environment*, IEEE Transactions on Automatic Control, AC-31:1058-1063, 1986.
2. V. Lumelsky and A. Stepanov, *Path planning strategies for point automation moving amidst unknown obstacles of arbitrary shape*, Algorithmica, 2:402-430, 1987.

Continuous and discrete visibility

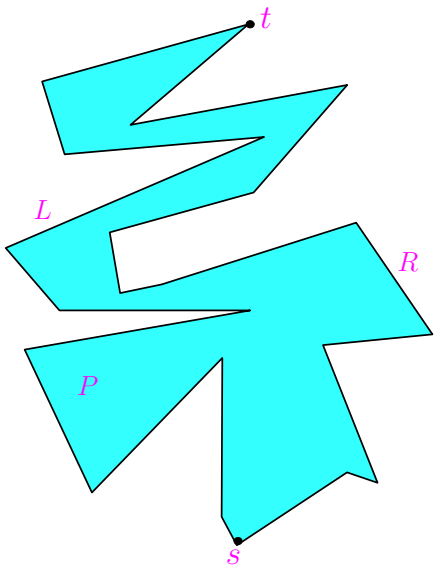


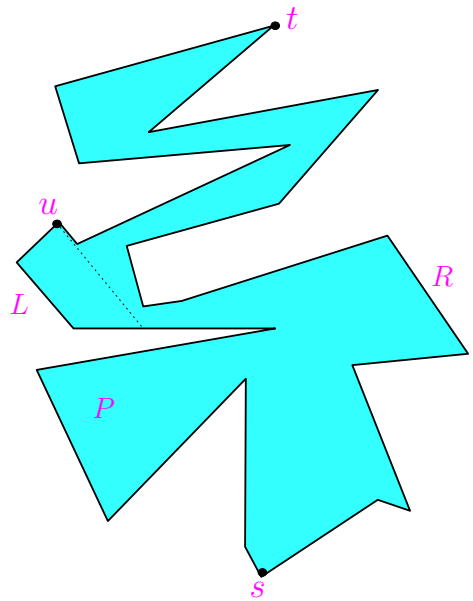


If the robot computes visibility polygons from each points on its path, we say that P is explored under continuous visibility.

If the robot computes visibility polygons from a selected set of points on its path, we say that P is explored under discrete visibility.

Searching for a target in an unknown street





A simple polygon P is said to be a *street* (also called *LR-visibility polygon*) if there exists two points s and t on the boundary of P such that every point of the clockwise boundary from s to t of P (denoted as L) is visible from some point of the counterclockwise boundary of P from s to t (denoted as R) and vice versa.

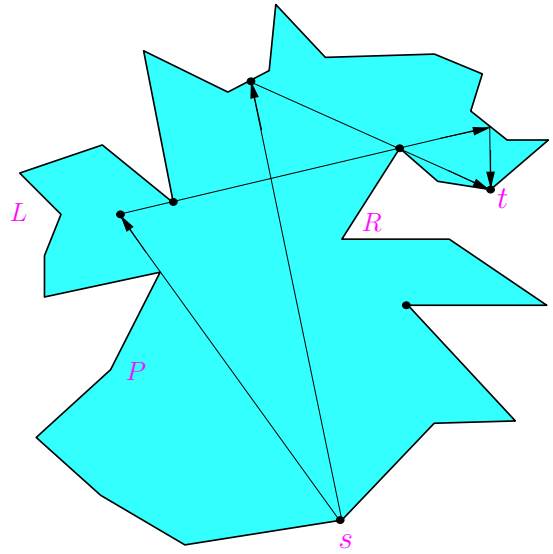
Observe that if a point robot moves along any path between s and t inside the street P, it can see all points of P.

Algorithms for target searching in an unknown 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. C. Icking, *Motion and visibility in simple polygons*, Ph.D. Thesis, FernUniversität, 1994. Competitive ratio: 4.44.
3. 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: 2.61.
4. A. López-Ortiz and S. Schuierer, *Going home through an unknown street*, Proceedings of Algorithms and Data Structures, LNCS 955, pp. 135-146, Springer-Verlag, 1995. Competitive ratio: 2.05.
5. A. López-Ortiz and S. Schuierer, *Walking streets faster*, Proceedings of the 5th Scandinavian Workshop on Algorithm Theory, LNCS 1097,pp. 345-356, Springer-Verlag, 1996. Competitive ratio: 1.73.

6. P. Dasgupta and P. Chakrabarti and S. De Sarkar, *A new competitive algorithm for agent searching in unknown streets*, Proceeding of the 16th Symposium on FSTTCS, LNCS 1180, pp. 32-41, Springer-Verlag, 1995. Competitive ratio: 1.71.
7. I. Semrau, *Analyse und experimentelle Untersuchung von Strategien zum Finden eines Ziels in Str$\alpha\beta$enpolygonen*, Diploma Thesis, FernUniversität, 1996. Competitive ratio: 1.57.
8. E. Kranakis and A. Spatharis, *Almost optimal on-line search in unknown streets*, Proceedings of the 9th Canadian Conference on Computational Geometry, pp. 93-99, 1997. Competitive ratio: 1.498.
9. C. Icking, R. Klein, E. Langetepe and S. Schuierer, *An optimal competitive strategy for walking in streets*, SIAM Journal on Computing, 33(2004), 462-486. Competitive ratio: 1.41.

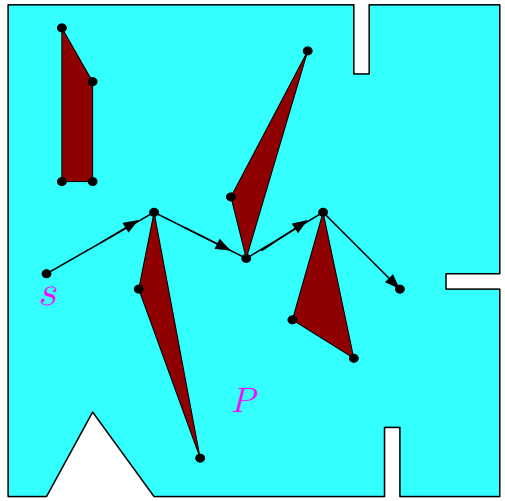
Target searching using link paths



Another problem for searching t in an unknown street P is find a path such that the number of links (or, turns) in the path is as small as possible.

1. 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.

Exploring unknown polygons: continuous visibility



Starting from a point s inside P, the exploration problem is to design an online algorithm which a point robot can use for moving inside P such that every point of P becomes visible from some point on the exploration path of the robot

However, if P contains holes, the exploration problem does not admit competitive strategy.

1. X. Deng, T. Kameda and C. Papadimitriou, *How to learn an unknown environment*, Proceedings of the 32nd Annual IEEE Symposium on Foundation of Computer Science, PP. 298-303, 1991.

Exploring unknown polygons: discrete visibility

In the remaining part of the lecture, we present exploration algorithms and their competitive ratios from the following papers.

1. S. K. Ghosh, J. W. Burdick, A. Bhattacharya and S. Sarkar, *On-line algorithms with discrete visibility: Exploring unknown polygonal environments*, Special issue on Computational Geometry approaches in Path Planning, IEEE Robotics and Automation Magazine, vol. 15, no. 2, pp. 67-76, 2008.
2. S. K. Ghosh and J. W. Burdick, *An on-line algorithm for exploring an unknown polygonal environment by a point robot*, Proceedings of the 9th Canadian Conference on Computational Geometry, pp. 100-105, 1997.
3. A. Bhattacharya, S. K. Ghosh and S. Sarkar, *Exploring an Unknown Polygonal Environment with Bounded Visibility*,Proceedings of the International Conference on Computational Science, Lecture Notes in Computer Science, No. 2073, pp. 640-648, Springer Verlag, 2001.

Motivation for discrete visibility

Many 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.
2. At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming.
3. The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot's trajectory.
4. For good visibility, the robot's camera will typically be mounted on a mast and such devices vibrate during the robot's movement.
5. Hence for good precision the camera must be stationary while computing visibility polygons.

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

Exploration cost

Is the cost associated with a robot's physical movement dominate all other associated costs?

The essential components that contribute to the total cost required for a robotic exploration can be analyzed as follows. 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 where the robot has no apriori knowledge of the environment's geometry, 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.
3. 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 required to complete the exploration of P. Hence, the total cost of an exploration is equated to the total time T required to explore P: $T(P) = t_M\ N_M + t_S\ N_S + r\ D$.

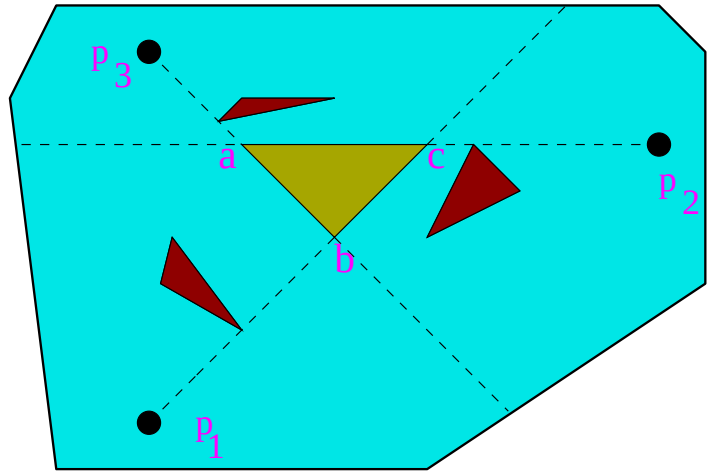
Now, $(t_M\ N_M + t_S\ N_S)$ can be viewed as the time required for computing and maintaining visibility polygons by computer vision algorithms, which is indeed a significant fraction of $T(P)$ because computer vision algorithms consume significant time on modest computers in a relatively cluttered environment.

Therefore, we assume that the overall cost of exploration is proportional to the cost for computing visibility polygons.

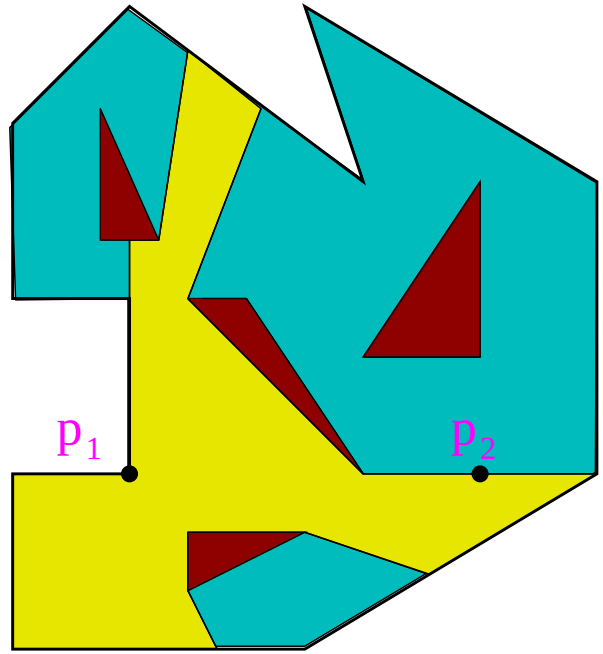
The criteria for minimizing the cost for robotic exploration is to reduce the number of visibility polygons that the on-line algorithms compute.

1. J. Borenstein and H. R. Everett and L. Feng, *Navigating mobile robots: sensors and techniques*, A. K. Peters Ltd., Wellesley, MA, 1995.
2. O. Faugeras, *Three-dimensional computer vision*, MIT Press, Cambridge, 1993.

An exploration algorithm



- We present an exploration algorithm that a point robot can use to explore an unknown polygonal environment P under discrete visibility.
- In order to explore P, the robot starts from a given position, and sees all points of the free space incrementally.
- It may appear that it is enough to see all vertices and edges of P in order to see the entire free-space. However, this is not the case.
- Three views from p_1, p_2 and p_3 are enough to see all vertices and edges of P but not the entire free-space of P.



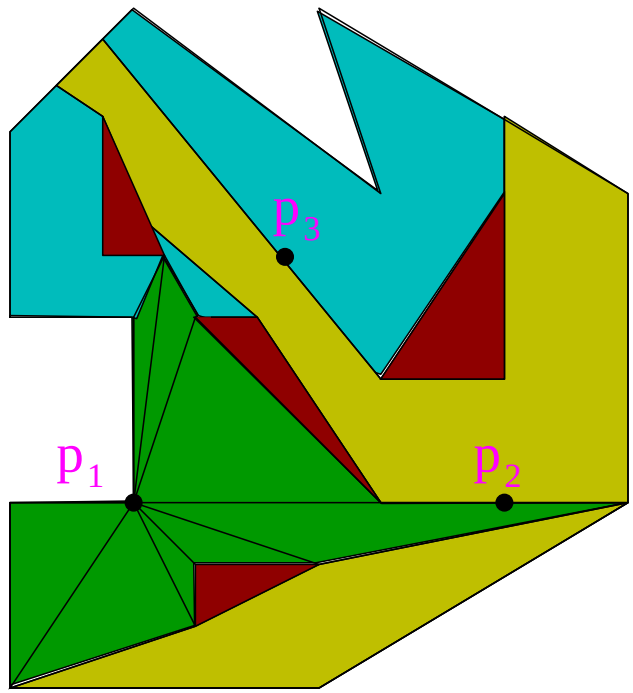
(i) Let S denote the set of viewing points that the algorithm has computed so far. (ii) The triangulation of P is denoted as $T(P)$. (iii) The visibility polygon of P from a point p_i is denoted as $VP(P, p_i)$.

Step 1: $i := 1$; $T(P) := \emptyset$; $S := \emptyset$; Let p_1 denote the starting position of the robot.

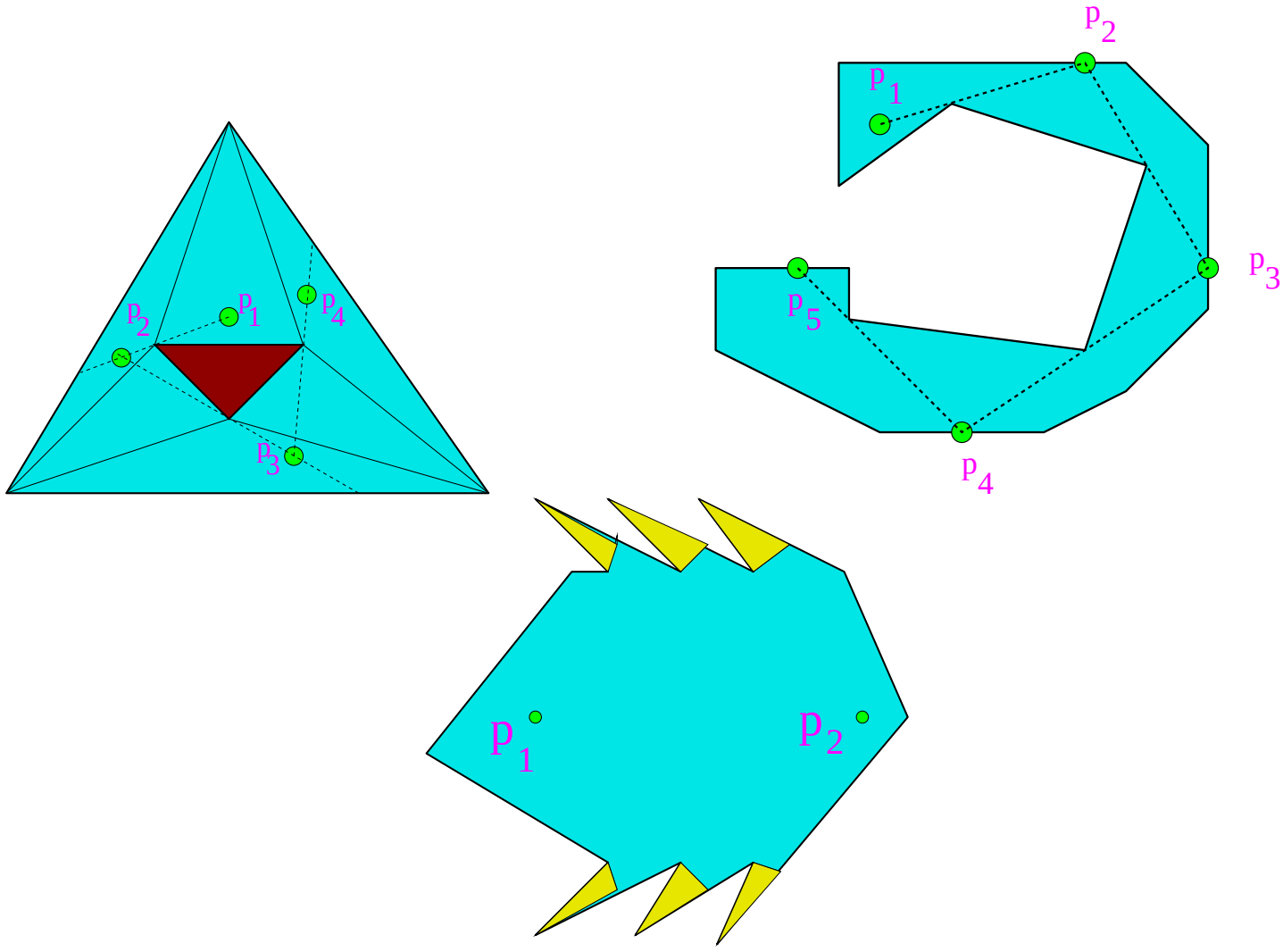
Step 2: Compute $VP(P, p_i)$; Construct the triangulation $T'(P)$ of $VP(P, p_i)$; $T(P) := T(P) \cup T'(P)$; $S = S \cup p_i$;

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

Step 4: *If* $i = 0$ *then goto* Step 7;
Step 5: *If* $VP(P, p_i) - T(P) \neq \emptyset$ *then* choose a point z on any constructed of $VP(P, p_i)$ lying outside $T(P)$;
Step 6: $i := i + 1$; $p_i := z$; *goto* Step 2;
Step 7: Output S and $T(P)$;
Step 8: Stop.



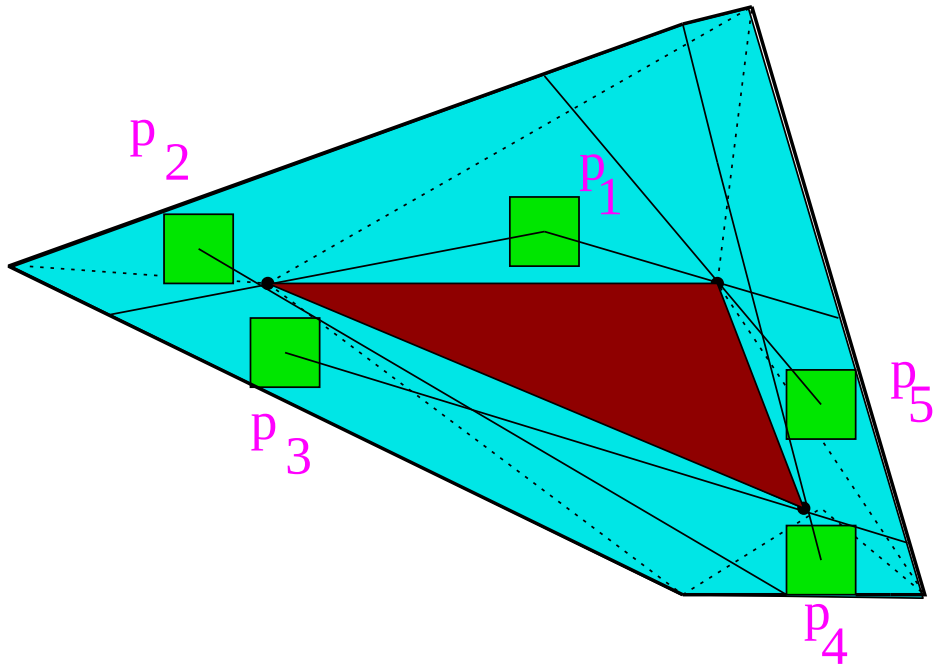
Competitive ratio



The algorithm needs $r + 1$ views. Competitive ratio is $(r + 1)/2$, where r denotes the number of reflex vertices of the polygon.

Open Problem: Can the bound be improved?

Convex robot exploration



We wish to design an algorithm that a convex robot C can use to explore an unknown polygonal environment P (under translation) following the similar strategy of a point robot.

C needs more than $r + 1$ views for exploration.

Open problem

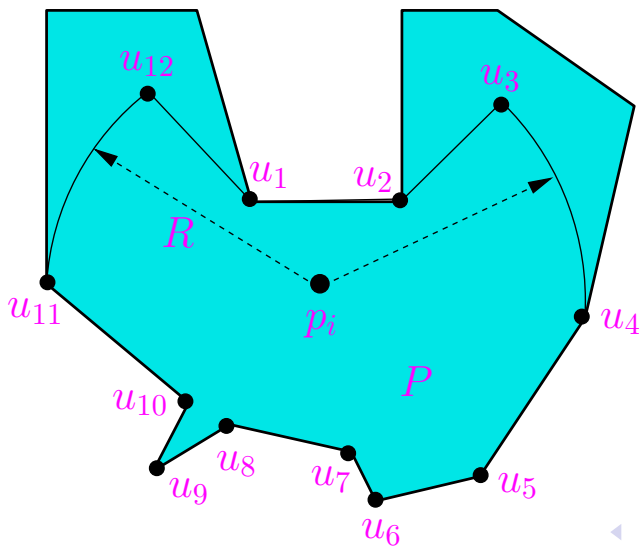
Can one derive an upper bound on the number of views for a convex robot exploration?

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.

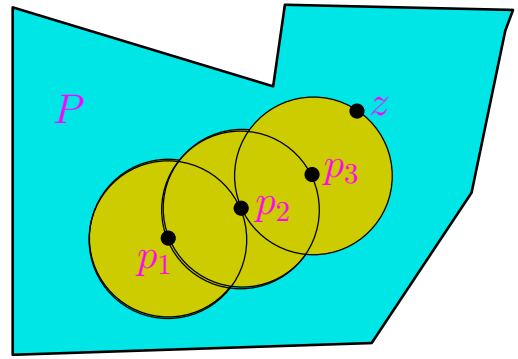
Therefore, the portion of the boundary of a polygonal environment within the range distance R is only considered to be visible from the camera of the robot.

Vertices of restricted visibility polygon from p_i with range R are $u_1, u_2, \ldots, u_{12}$.

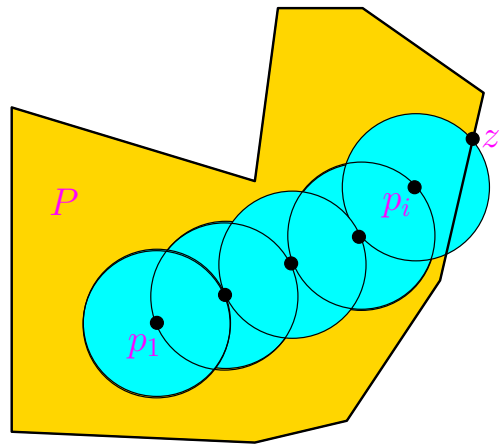


An exploration algorithm using restricted visibility

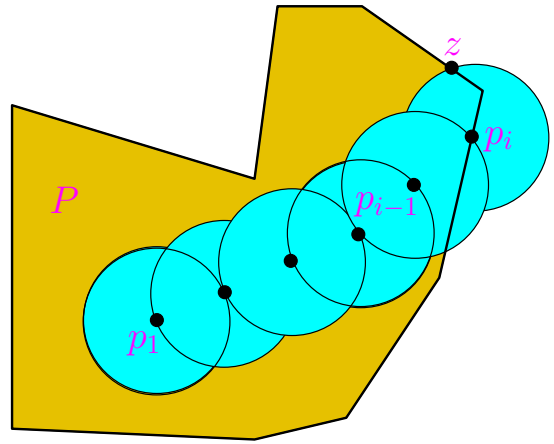
- The algorithm starts by computing the restricted visibility polygon $RVP(P, p_1)$ from the starting position p_1.



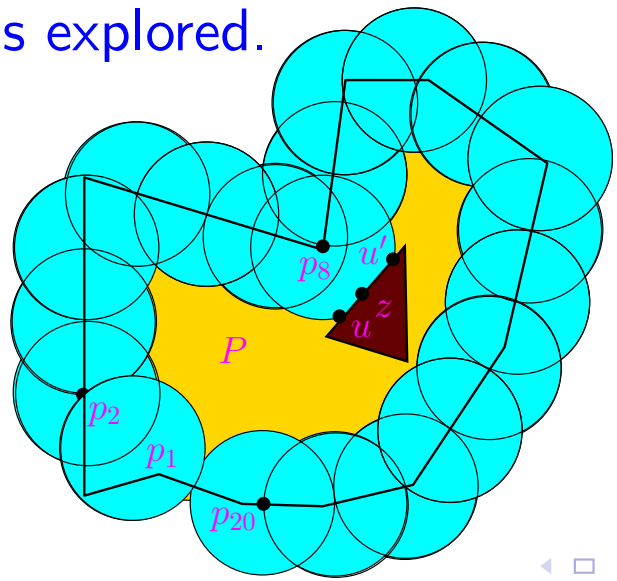
- It chooses the next viewing point p_i on a constructed edge or a circular edge of $RVP(P, p_{i-1})$ for $i \geq 1$ till a boundary point z of P becomes visible.



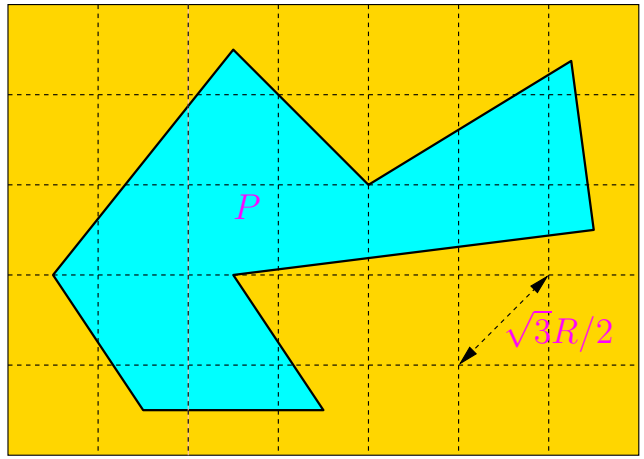
- Taking z as the next viewing point p_i, $RVP(P, p_i)$ is computed. Taking viewing points along the boundary of P in this fashion, restricted visibility polygons are computed till all points of this boundary of P become visible.



- The process of computing restricted visibility polygons ends once the entire P is explored.



Competitive ratio



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

$$\left\lfloor \frac{8 \times Area(P)}{3 \times R^2} \right\rfloor + \left\lfloor \frac{Perimeter(P)}{R} \right\rfloor + r + h + 1.$$

The competitive ratio of the algorithm is

$$\left\lfloor \frac{8\pi}{3} + \frac{\pi R \times Perimeter(P)}{Area(P)} + \frac{(r+h+1) \times \pi R^2}{Area(P)} \right\rfloor.$$

Open problem

Can one improve the competitive ratio of the algorithm?

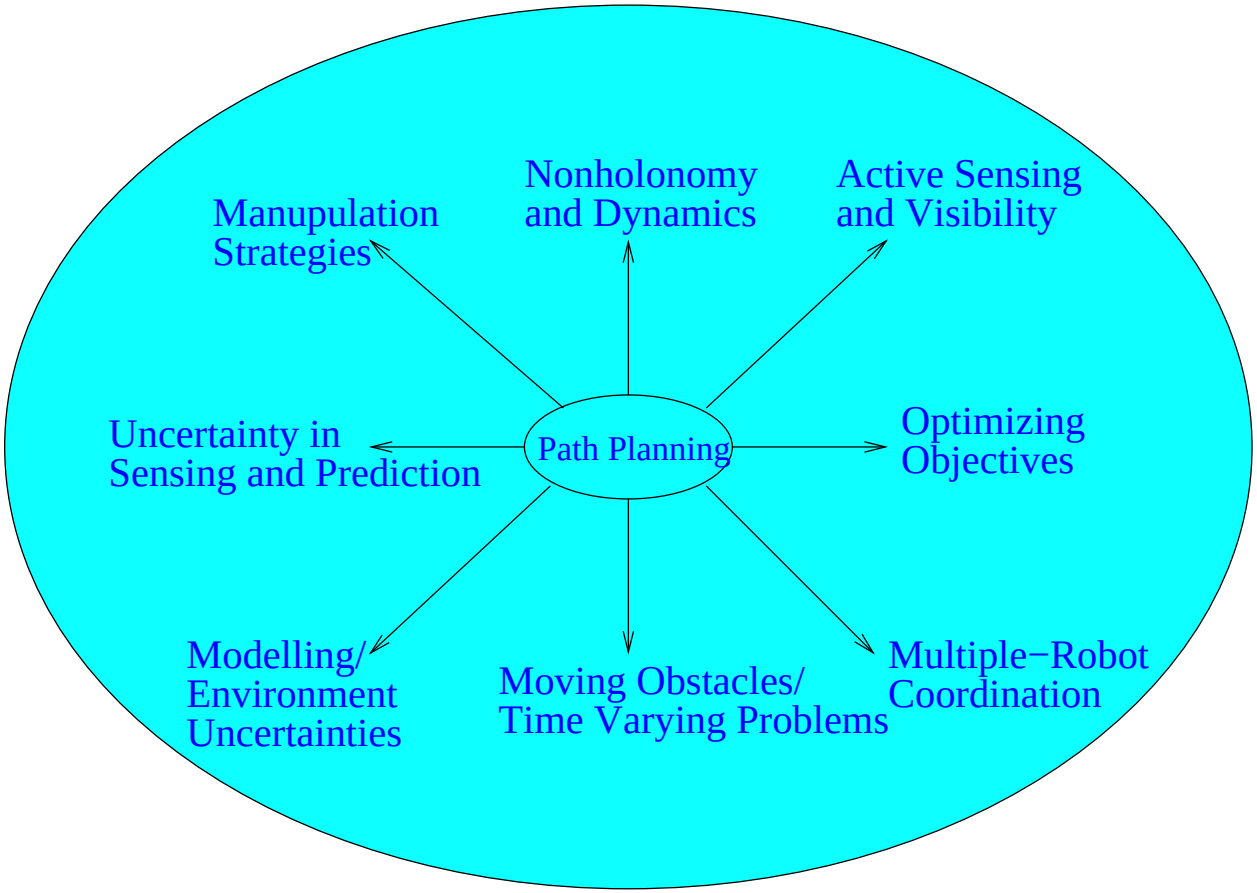
Exploration and Coverage Algorithms

1. A. Bhattacharya, S. K. Ghosh and S. Sarkar, *Exploring an Unknown Polygonal Environment with Bounded Visibility*, Lecture Notes in Computer Science, No. 2073, pp. 640-648, Springer Verlag, 2001.
2. S. K. Ghosh, J. W. Burdick, A. Bhattacharya and S. Sarkar, *On-line algorithms for exploring unknown polygonal environments with discrete visibility*, Special issue on Computational Geometry approaches in Path Planning, IEEE Robotics and Automation Magazine, 2008 vol. 15, no. 2, pp. 67-76, 2008.
3. E. U. Acar and H. Choset, *Sensor-based coverage of unknown environments: Incremental construction of morse decompositions*, The International Journal of Robotics Research, 21 (2002), 345-366.
4. 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.

5. H. Choset, *Coverage for robotics– A survey of recent results*, Annals of Mathematics and Artificial Intelligence, 31 (2001), 113-126.
6. X. Deng, T. Kameda and C. Papadimitriou, *How to learn an unknown environment I: The rectilinear case*, Journal of ACM, 45 (1998), 215-245.
7. F. Hoffmann, C. Icking, R. Klein and K. Kriegel, *The polygon exploration problem*, SIAM Journal on Computing, 31 (2001), 577-600.
8. C.J. Taylor and D.J. Kriegman, *Vison-based motion planning and exploration algorithms for mobile robot*, IEEE Transaction on Robotics and Automation, 14 (1998), 417-426.
9. P. Wang, *View planning with combined view and travel cost*, Ph. D. Thesis, Simon Fraser University, Canada, 2007.
10. S. P. Fekete and C. Schmidt, *Polygon exploration with time-discrete vision*, Computational Geometry: Theory and Applications, 43:148-168, 2010.

Concluding remarks

In this talk, we have presented a few off-line and on-line algorithms for robot path planning in the plane.



Basic path planning problems.

Thank You.