

# Robot Localization Simulator

Prateek Garg, Kumar Ashwani, Apurv Verma

April 8, 2012

# Acknowledgement

We would like to acknowledge Assistant Professor Apurva Mudgal for helping us understand the difficult Robot Localization Algorithm. His supervision made it appear all very easy. We are also thankful to the ever enthusiastic CGAL community for helping us with the various CGAL issues.

# Introduction

A robot is placed at an unknown point inside a simple polygon  $P$ . The robot has a map of  $P$  and can compute visibility polygon from its current location. The robot must determine its correct location inside the polygon  $P$  at a minimum cost of travel distance.

# Robot Localization Algorithm I

## Input:

Map polygon  $P$ , the visibility polygon  $V$ .

## Output:

The robot localizes to its actual position  $h \in H$

- 1: Compute the set of hypotheses  $H$ .
- 2: **while**  $|H| > 1$  **do**
- 3: Compute the majority-rule map  $P_{maj}$
- 4: Compute the polygons  $G_{ij}$  for each pair of hypotheses,  $h_i$  and  $h_j$
- 5: Compute the majority rule map  $K_i$  of  $G_{ij}$ 's
- 6: Find the edges on the boundary of  $K_i$  which are not on the boundary of  $P_{maj}$
- 7: Draw grids and compute the set of coordinates  $Q_H$  on these edges.
- 8: Make instance  $I_{P,H}$  of  $\frac{1}{2}$ -Group Steiner Problem

# Robot Localization Algorithm II

- 9: Solve  $I_{P,H}$  to compute a half computing path  $C \subset P_{maj}$
- 10: Half-Localize by tracing  $C$  and making observations at coordinates  $Q_H$
- 11: Move back to the starting location.
- 12: **end while**

Visibility polygon is an indispensable component in the hypothesis generation step of the algorithm. Since CGAL had no inbuilt support for computing visibility polygons we implemented the following two routines for our purposes.

- ▶ Visibility Polygon of a point inside a polygon
- ▶ Visibility Polygon of an edge of the polygon.

# Visibility Polygon of a Point Inside a Polygon I

## Definition

**Visibility Polygon of Point:**  $p$  is the bounded polygonal region of all points of the polygon visible from  $p$ .

## Algorithm

1. Collect all the vertices of the polygon which are visible from the point  $P$ .
2. Iterate over the list of visible vertices and for each reflex vertex, compute the spurious vertex introduced in the visibility polygon.
3. Finally sort all the vertices in an order so that they form a simple polygon.



# Examples I

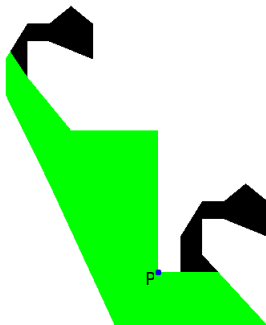


Figure: Visibility Polygon of Point

# Examples II

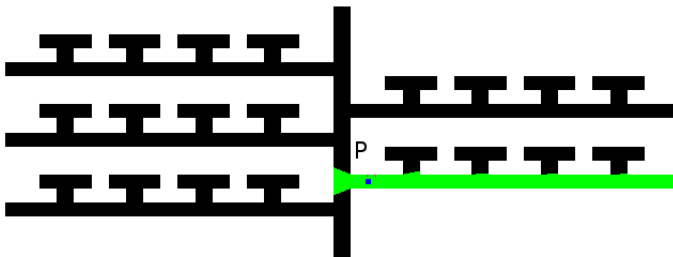


Figure: Visibility Polygon of Point

# Visibility Polygon of an edge of the polygon

## Definition

**Visibility Polygon of Edge:**  $e$  is the bounded polygonal region of all points of the polygon visible from any point on the edge  $e$ .

The algorithm for the visibility polygon of an edge has been taken from [Guibas()].

# Visibility Polygon of an edge of the polygon

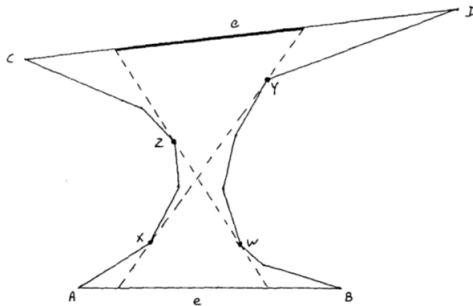


Figure: Visibility Polygon of Edge, Illustration taken from:[Guibas()]

# Algorithm 1

1. Compute the shortest path  $P_{AC}$ , from A to C and the shortest path  $P_{BD}$ , from B to D. Call this pair 1.
2. Similarly compute the shortest path  $P_{AD}$ , from A to D and the shortest path  $P_{BC}$ , from B to C. Call this pair 2.
3. Find out which of these pairs is outward convex. An outward convex pair implies an hourglass shape is formed by the two paths.
4. If none of the pairs is outward convex this means that no portion of edge  $CD$  is visible from any point on edge  $AB$  and we can completely ignore such an edge.
5. If one of the pairs is outward convex then without loss of generality, let pair 1 be the outward convex pair. Now compute the shortest paths  $P_{AD}$  and  $P_{BC}$ .

# Algorithm II

6. Let  $X$  be the point where path  $P_{AD}$  and  $P_{AC}$  split and let  $W$  be the point where path  $P_{BD}$  and  $P_{BC}$  split. Let  $Y$  be the next point on the path  $P_{AD}$  and  $Z$  be the next point on the path  $P_{BC}$ . Extending  $XY$  we get one extreme point of the portion of  $CD$  visible from  $AB$ . We repeat this on other side to get the other extreme point.

# Shortest Path Calculation I

For the calculation of shortest path between any two vertices of the polygon the following property was exploited.

- ▶ The shortest path must turn only at vertices of the polygon.
- ▶ It is possible to move from one vertex to the another only if they are visible to each other.

# Visibility Graph I

## Definition

**Visibility Graph** The visibility graph of a polygon can be formed as follows. Draw a vertex corresponding to each vertex in the polygon. Draw an edge between two vertices if the line joining the corresponding vertices in the polygon lies completely inside the polygon.



# Visibility Graph II

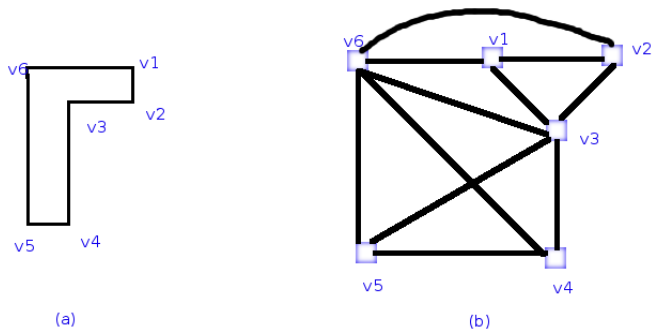


Figure: Visibility Graph

# Examples

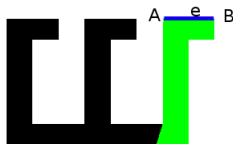


Figure: Visibility Polygon of Edge

# Examples

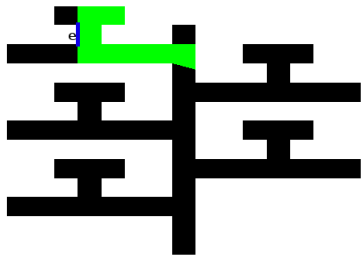


Figure: Visibility Polygon of Edge

# Hypothesis Generation

## Theorem

*A point,  $P$  inside a simple polygon sees atleast one edge of the polygon completely.*

## Definition

**Spurious Edge:** In the visibility polygon of a point, an edge is called a spurious edge if it is obtained by extending the line joining the point  $P$  and a reflex vertex till it meets the polygon.

## Theorem

*The visibility polygon of a point  $P$  has atleast one edge which completely overlaps with an edge of the original polygon.*

# Algorithm

1. Iterate over the edges of the polygon and the edges of the map. and find an edge in the map which has the same length and orientation as an edge in the polygon.
2. Translate the visibility polygon such that the matching edge of the map polygon and the visibility polygon coincide.
3. For each of the remaining edges of the visibility polygon, check whether a complete match exists or not. If all the remaining edges match, the point where the origin was translated is added to the set of hypotheses.

# Examples I

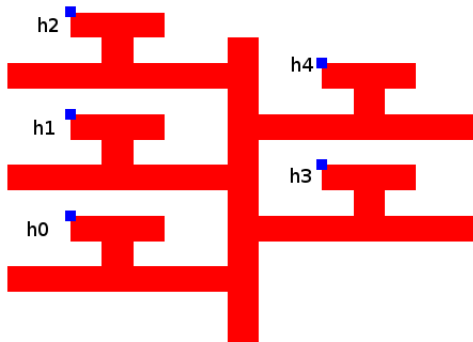


Figure: Hypothesis Generation

# Examples II

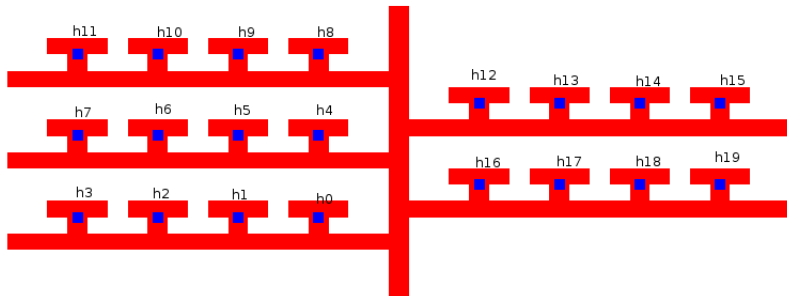


Figure: Hypothesis Generation

# Majority Rule Map

The following example taken from [Apurva Mudgal(2006)] demonstrates the construction of a majority rule map.

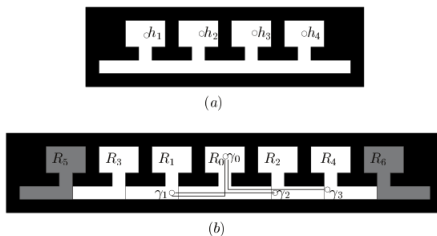


FIG. 2.1. (a) A half-localization problem with grid graph  $G$  and  $H = \{h_1, h_2, h_3, h_4\}$ . (b) The majority-rule map for  $\text{HALF-LOCALIZE}(G, H)$  with two halving paths  $(\gamma_0, \gamma_1, \gamma_2)$  and  $(\gamma_0, \gamma_3)$ .

Figure: Majority Rule Map Construction



# Majority Rule Map

$h_1, h_2, h_3, h_4$  form the set of hypotheses. Arbitrarily we choose  $h_1$  as the origin .

Next we translate all the remaining hypotheses to  $h_1$  to obtain the overlay arrangement. The overlay arrangement contains the following faces  $R_0, R_1, R_2, R_3, R_4, R_5, R_6$ . Recall from the definition of  $Maj(\gamma)$

$$Maj(R_0) = h_1, h_2, h_3, h_4, \quad Maj(R_1) = h_2, h_3, h_4,$$

$$Maj(R_2) = h_1, h_2, h_3, \quad Maj(R_3) = h_3, h_4 \text{ and } Maj(R_4) = h_1, h_2$$

In the majority rule map the region  $R_5$  and  $R_6$  are blocked because less than half the hypothesis said that they were traversable. They have been shown in gray.

1. The overlay arrangement can be easily constructed using CGAL's inbuilt Arrangement class. Obtain the translates of the polygon by choosing one hypothesis as the origin and shifting other hypothesis to it.
2. Insert all these translates in CGAL's inbuilt arrangement to obtain all the faces in the overlay arrangement.
3. Faces which belong to atleast half the hypothesis are marked as part of the majority rule map.

# Examples I

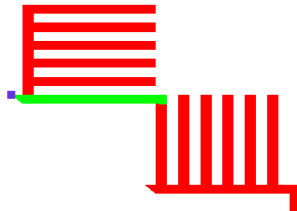


Figure: Map Polygon with robot position and Visibility Polygon

# Examples II

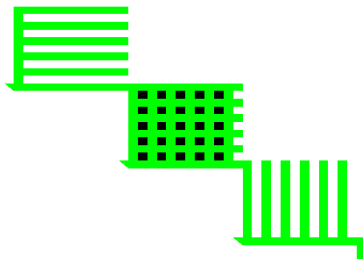


Figure: Majority Rule Map

# Connected Component containing Origin in $P_{maj}$

We need to calculate connected component containing origin in  $P_{maj}$  as the robot can only move in this area .

**Dual graph** of a given planar graph  $G$  is a graph which has a vertex corresponding to each face of  $G$  and an edge joining two neighbouring faces for each edge in  $G$ . We also have a vertex for the unbounded face which is connected to all the faces sharing boundary with unbounded face.

# Dual Graph

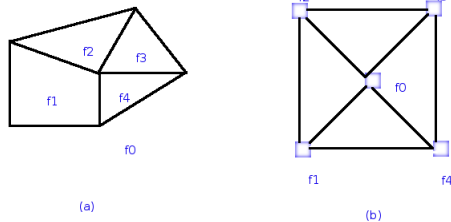


Figure: (a) Graph  $G$  (b) Dual Graph for  $G$

# Algorithm for Majority Map

1. Prepare a dual graph  $G$  of the traversible faces in the  $P_{maj}$ .
2. Find the vertex( $v_0$ ) corresponding to the face containing origin in dual graph  $G$ .
3. Perform Depth First Search `[[BOO()]]` from  $v_0$  to obtain all the connected vertices.
4. Output the union of the faces corresponding to the connected vertices obtained above.

# Examples I

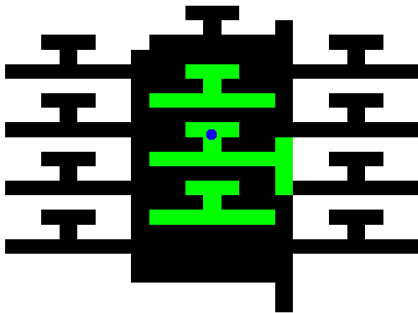


Figure: A Majority Map(blue point represents origin)



# Examples II

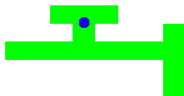


Figure: Connected Component Containing Origin

# Computing the Group Boundaries, $G_{ij}$ I

To compute  $G_{ij}$  we first compute  $F_{ij}$ .  $F_{ij}$  is the face containing origin in the overlay of polygons  $P_i$  and  $P_j$ . The following diagram taken from [Apurva Mudgal(2006)] demonstrates the notation and the construction.

# Computing the Group Boundaries, $G_{ij}$ II

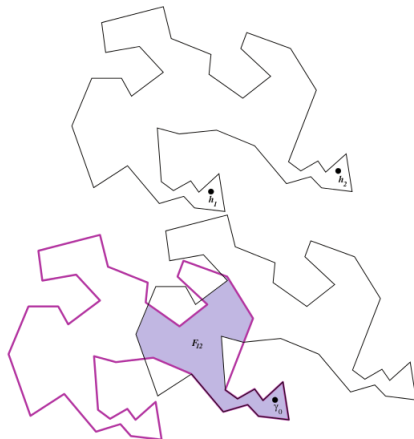


Figure:  $F_{12}$

# Algorithm 1

To obtain  $G_{ij}$  from  $F_{ij}$  we draw visibility polygon of type 1 and type 2 edges and take their lower envelope.

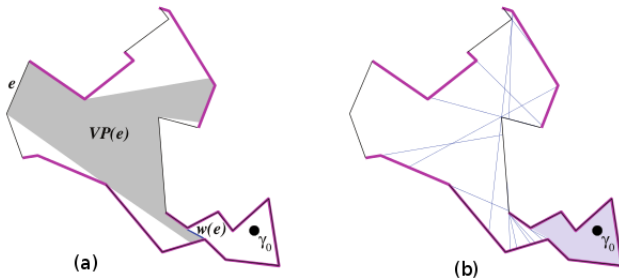


Figure: Lower Envelope

# Algorithm II

Finding the lower envelope is easy using the CGAL's inbuilt Arrangement class. We find the visibility polygon of each edge of type 1 or type 2 and insert it into an arrangement. Later we check the face which contains the point  $\gamma_0$ . This face is nothing else but  $G_{ij}$ . The above algorithm is repeated to obtain  $G_{i1}$ ,  $G_{i2}$ ,  $G_{i3}$ ,  $G_{i4}$ , ....  $G_{ik}$ .

# Examples I

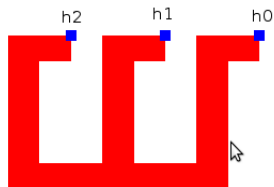


Figure: Lower Envelope

# Examples II

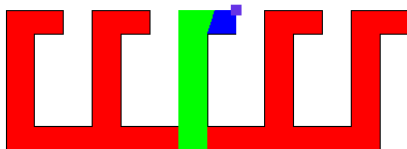


Figure: Lower Envelope

# Examples III



Figure: Lower Envelope



# Computing the Group Boundaries, $K_i$ I

To obtain  $K_i$  we construct the majority rule map of all  $G_{ij}$ 's.  $K_i$  is a region of special interest because of the special following special property. For proof of it, please refer [Apurva Mudgal(2006)]  
A robot initially located at  $h_i$  half localizes if it crosses the boundary of  $K_i$ .

# Examples I

For The below map the blue region shows  $G_{ij}$  and the combined region of blue and green represents  $F_{ij}$  in the overlay arrangement of the polygons  $P_i$  and  $P_j$ .

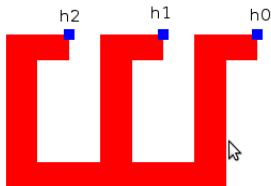


Figure: Map Polygon with Hypotheses

# Examples II

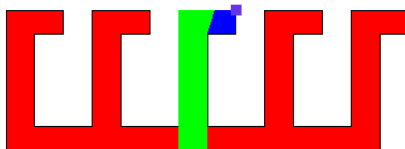


Figure:  $G_{02}$  and  $F_{02}$

# Examples III



Figure:  $G_{01}$  and  $F_{01}$



Figure:  $K_1$

# Computing Reference Points ( $Q_H$ )

## Definition

**Reference Points:** are the discrete set of points on the edges of  $\partial K_i \cap \partial G_i$  which are used to determine the half localization path of robot [see [Gregory Dudek(1998)]].

## Definition

**Half Localization Path:** is the path travelling along which the robot can eliminate half of the hypotheses by making observations at the reference points.

# Examples

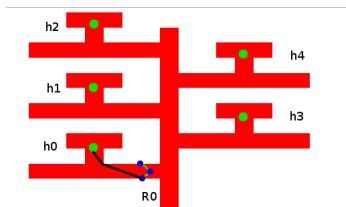


Figure: Reference Points in blue color and hypotheses in green color

The reference points shown in the figure are for the hypothesis  $h_0$ . The robot on following the path from  $h_0$  to  $R_0$  can differentiate between hypotheses  $\{h_0, h_1, h_2\}$  and  $\{h_3, h_4\}$  based on its observation at  $R_0$ . The path shown in black is the half computing path.

# Algorithm 1

1. For every  $i$  find those edges in  $K_i$  which are not part of the boundary of majority map. These are important edges on which we will find out reference points. Let  $L_i$  contains all these edges for a particular  $i$ .
2. Calculate  $r_0$ (geodesic) radius of the smallest geodesic disk centered on  $\gamma_0$  that intersects at least half of  $L'_i$ .
  - 2.1 For all  $i$  calculate the minimum distance between  $\gamma_0$  and any of the line segments of  $L_i$ .
  - 2.2 Take the median of all the distances calculated above as the geodesic radius( $r_0$ ).
3. Let  $k$  be the number of hypotheses and  $R$  be a sequence of radii  $r_0, 2 * r_0, 4 * r_0 \dots, 2^{\lceil \log_2 k \rceil}$ .
4. For every hypothesis  $i$  perform the following steps
  - 4.1 Place each line segment  $\sigma$  in  $L_i$  on an axis aligned square centered at  $\gamma_0$  of side length  $2 * 2^j * r_0$  (where  $j$  from  $(0..\lceil \log_2 k \rceil)$ )

# Algorithm II

- 4.2 Decompose the square into  $k \times k$  grid using  $k - 1$  horizontal and vertical lines.
- 4.3 Calculate the intersection of the line segment with the grid line. These intersection points are the reference points. Also include end points of the line segment (even if they do not lie on grid) in the set of reference points.



# Examples I

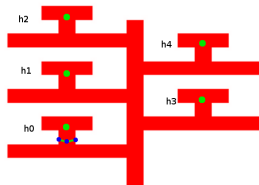


Figure: Reference Points in blue color and hypotheses in green color

# Examples II

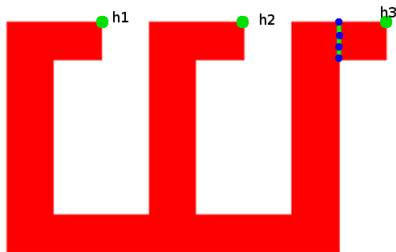


Figure: Reference Points in blue color and hypotheses in green color

# Bibliography I

 <http://www.boost.org/>.

 Joseph S. B. Mitchell Craig Tovey Apurva Mudgal,  
Sven Koenig.

A Near-Tight approximation algorithm for the robot  
localization problem.

*SIAM Journal on Computing*, 2006.

 Sue Whitesides Gregory Dudek, Kathleen Romanik.  
Localizing a Robot with Minimum Travel.  
1998.

 Daniel Leven Micha Sharir Robert E. Tarjan Guibas,  
Hershberger.

Linear time algorithms for visibility and shortest path problems  
inside simple polygons.