Visibility graphs

BY:

HAARIKA KONERU

Overview

Visibility graphs

Examples

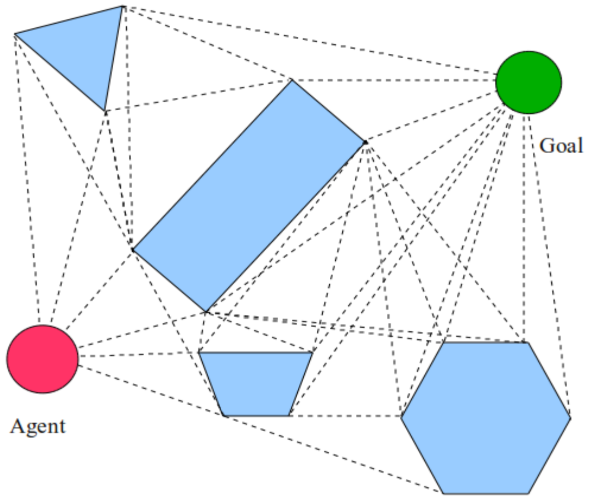
Reduced visibility graph

Applications

Visibility graphs for path planning

Visibility Graphs : Introduction

Visibility: We say that two points p and q are mutually visible if the open line segment joining them does not intersect the interior of any obstacle.

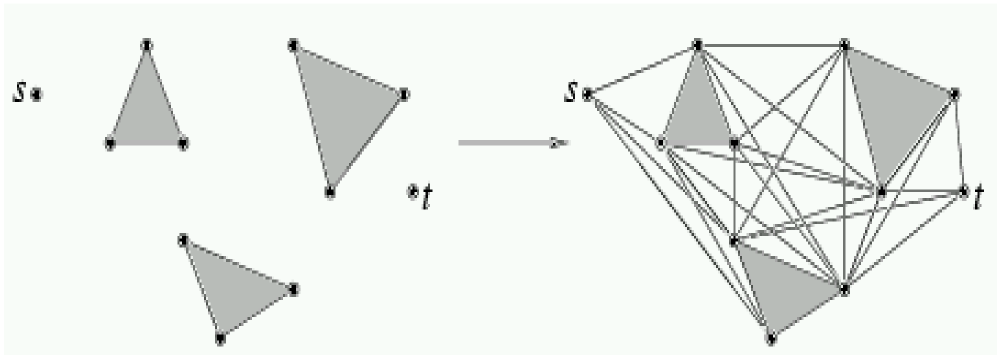


- **Visibility graph** is a graph of inter visible locations, typically for a set of points and obstacles in the Euclidean plane.
- Each node in the graph represents a point location, and each edge represents a visible connection between them
- That is, if the line segment connecting two locations does not pass through any obstacle, an edge is drawn between them in the graph.

Why Visibility Graphs?

Problem - We are given a set of n disjoint polygonal obstacles in the plane, and two points s and t that lie outside of the obstacles. The problem is to determine the shortest path from s to t that avoids the interiors of the obstacles.

Claim - The shortest path between any two points that avoids a set of polygonal obstacles is a polygonal curve, whose vertices are either vertices of the obstacles or the points s and t.



Visibility Graph Construction

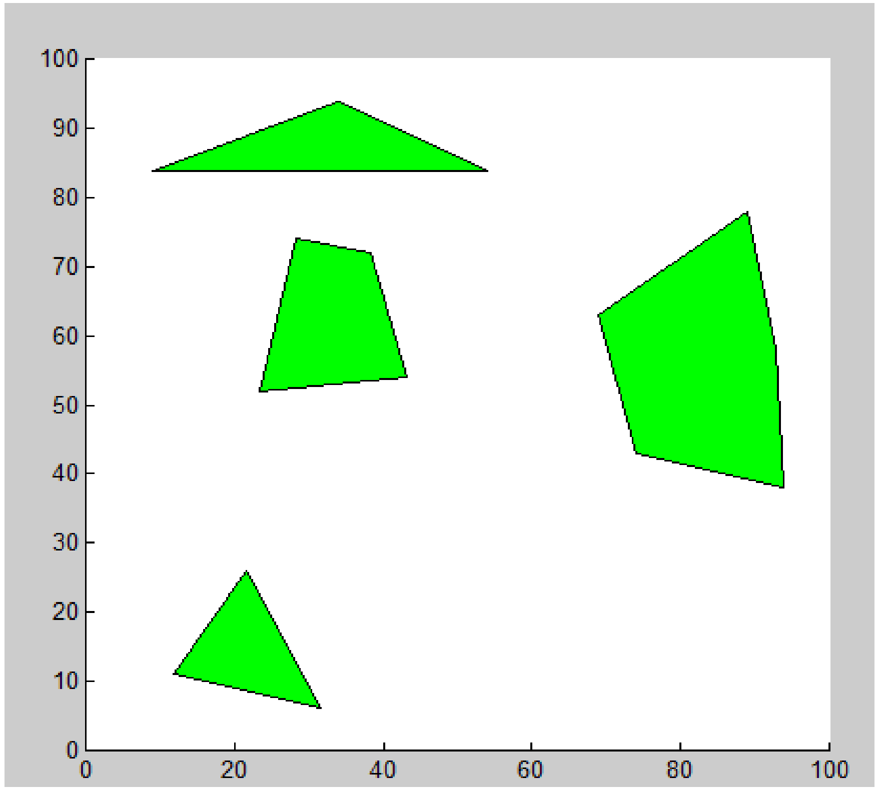
- To construct the visibility graph we must determine which vertices are visible from a vertex v , for every v.
- **Naïve algorithm***:

  Input – a set of vertices n_i , whose edges do not intersect (N in number)

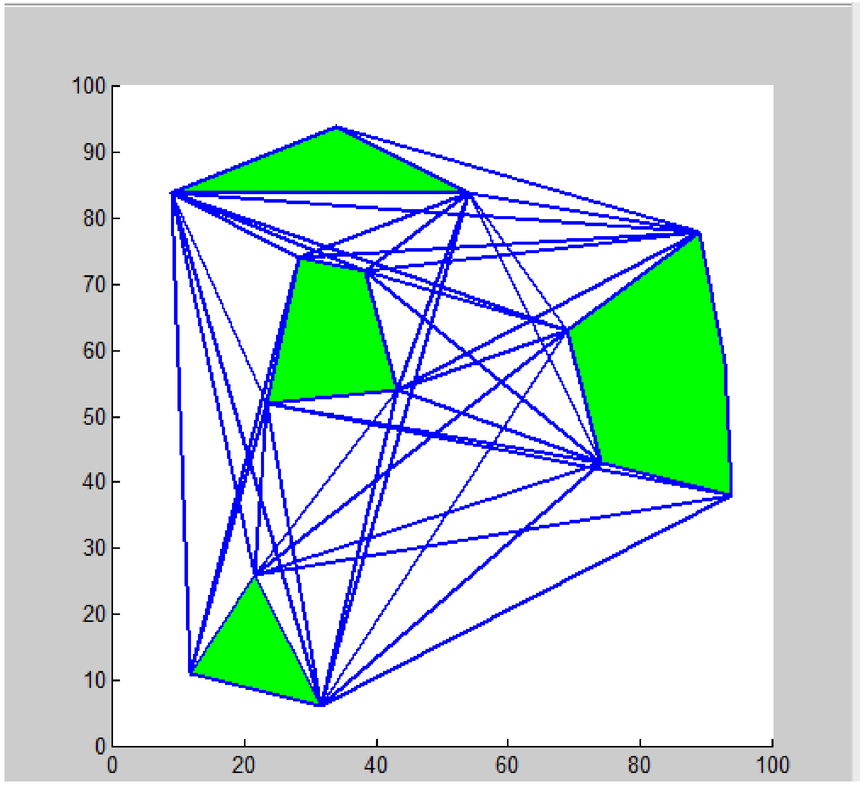
  1. Join the given vertex n_g to all other vertices n_i . (total of N-1)
  2. Check if the **line segment** $n_g n_i$ intersects the given edges(total of N)
  3. Repeat the procedure for every vertex (total of N)

  Altogether the algorithm is $O(n^3)$

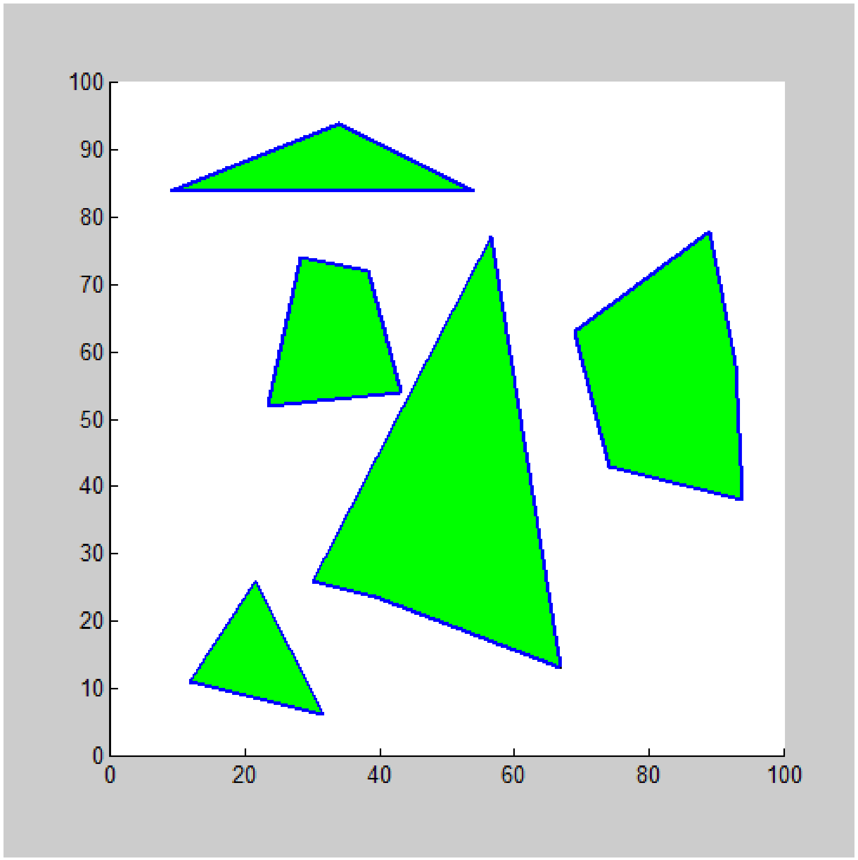
Visibility Graph: Some Examples



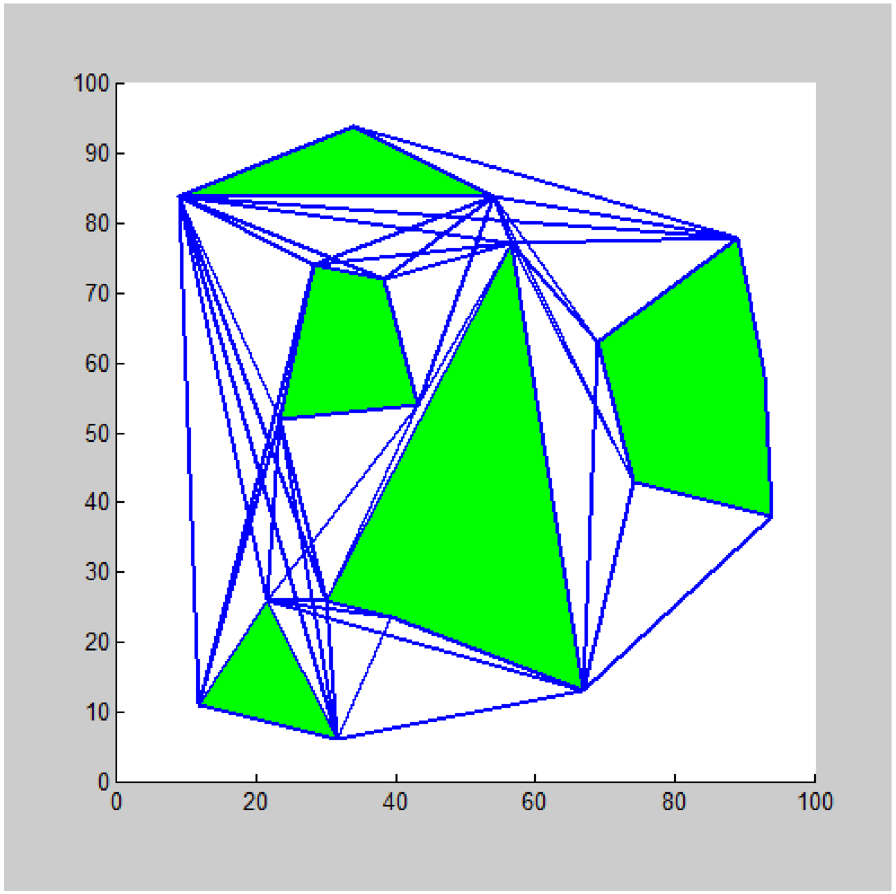
Visibility Graph: Some Examples



Visibility Graph: Some Examples



Visibility Graph: Some Examples



Reduced Visibility Graph

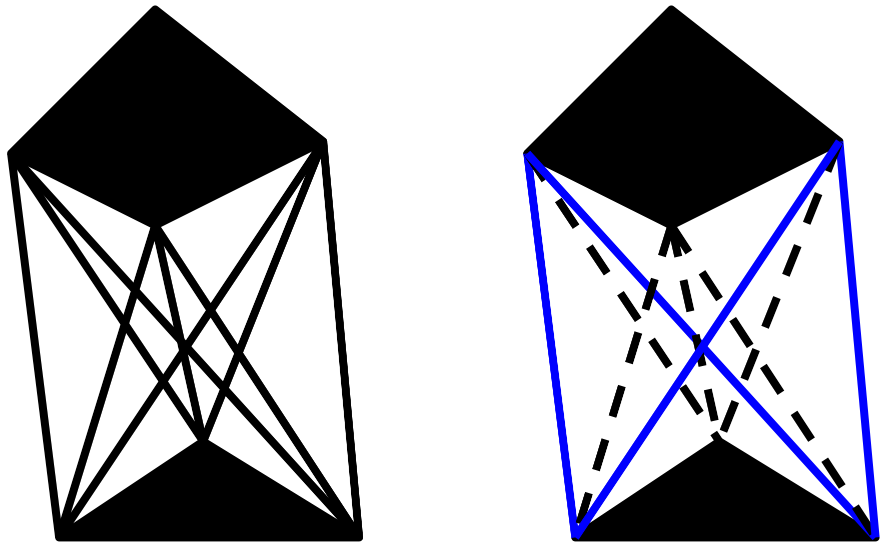
The standard visibility graph has many needless edges which are not going to be used in any case.

These are the ones that are going towards to the obstacles

The lines that are useful for path planning are the ones going around the obstacles, or passing through the vertices that are extremes around the obstacles.

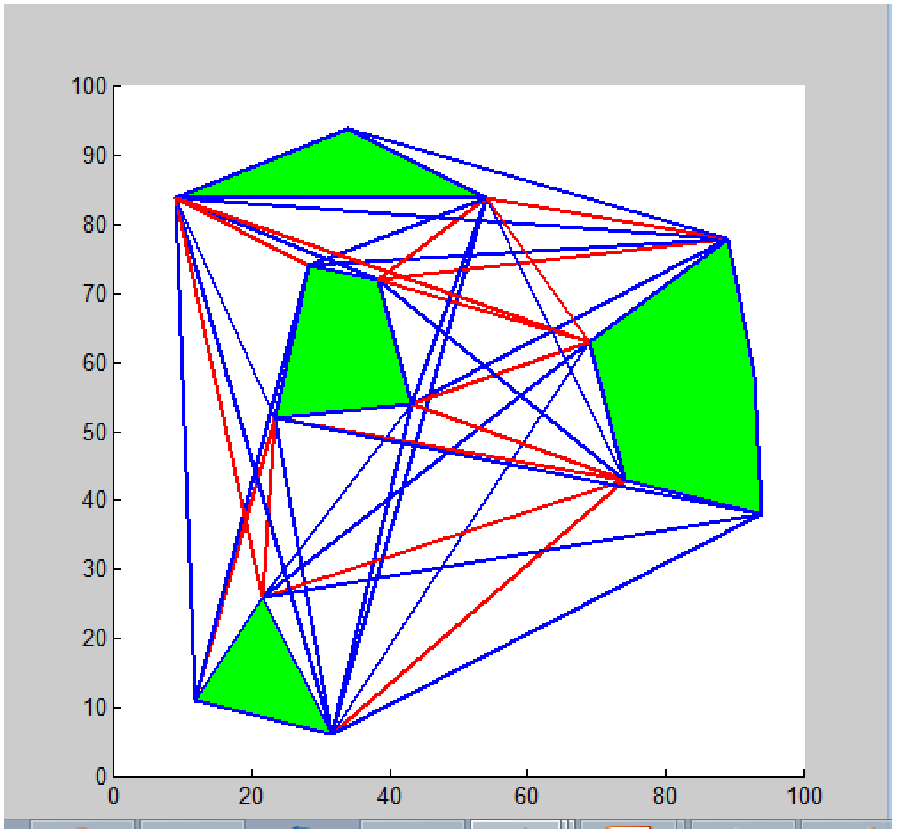
Eliminate concave obstacle vertices.

If we apply this property to the graph shown earlier, we obtain the reduced visibility graph.

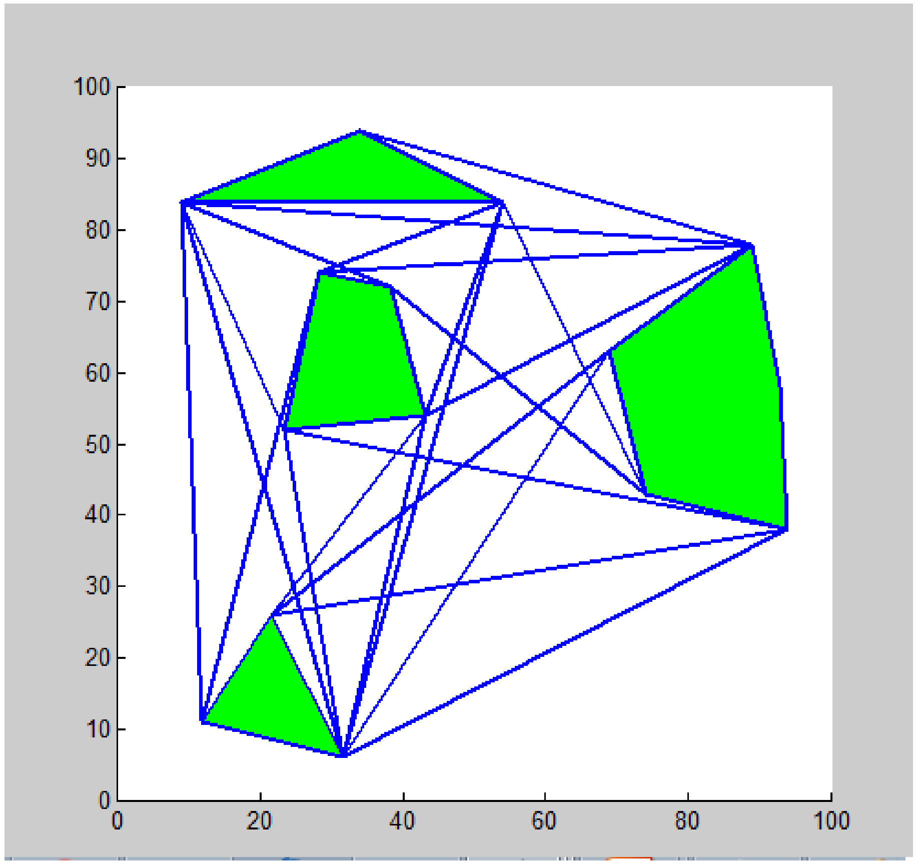


- Although shortest path doesn't change, the time it required to find considerably decreases.
- For instance there are 58 edges in a graph whereas in reduced graph case there are 36 edges which is 38% less than the standard graph.
- In the earlier case the shortest path finding requires 0.074 seconds, but it is 0.044 in reduced graph case. This is 40% faster than standard graph case.

Reduced Visibility Graph: Example



Reduced Visibility Graph: Example



Applications

- Robot path planning
- Placement of radio antennas
- Complex network theory
- Regional planning
- Military mission planning

Visibility graphs for Robot Path Planning

Visibility graphs are used in robot path planning which computes collision free paths of virtually any type moving among stationary obstacles.

Robot path planning steps

- Construction of visibility graph.
- Roadmap approach
- Shortest path between edges.

Dijkstra’s algorithm finds shortest path between two nodes of *n* nodes and *k* arcs graph in $O(n\log n + k)$ time.

```
$void$ ShortestPath ( $\boldsymbol{S}, p_{\text{start}}, p_{\text{goal}}$ ) {
    $G_{\text{vis}} \leftarrow$ VisibilityGraph ( $\boldsymbol{S} \cup \{p_{\text{start}}, p_{\text{goal}}\}$ ) ;
    Assign weight to each $e(u,v) \in G_{\text{vis}}$ ; // Euclidean length $\overline{uv}$
    Find shortest path from $p_{\text{start}}$ to $p_{\text{goal}}$ with Dijkstra's algorithm ;
}
```

References

1. https://en.wikipedia.org/wiki/Visibility_graph

2. http://cs.smith.edu/~streinu/Teaching/Courses/274/Spring98/Projects/Philip/fp/visibility.htm

3. http://www.cs.wustl.edu/~pless/546/lectures/l22.html

4. Choset, Howie M. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

Thank you!