Visibility Graph – Shortest Path in Polygonal Arena, Motion Planning
Amrendar Kumar, Vandana Swarnkar

ABSTRACT
This thesis is concentrating on the algorithm analysis and explanation of shortest path finding algorithm and most commonly known algorithm for shortest route is Dijkstra's algorithm. The shortest path from the source and the destination with inclusion of the vertices and set of vertices are considered in a polygonal path.

KeyWords
Robot; free space, polygonal area, path identifying, path visibility, shortest path, Dijkstra's Algorithm.
INTRODUCTION

O’Rourke (1987) strongly believed that the unsolved fundamental problems are involving in visibility of computer geometry. The visibility graphs have many of the unsolved problems in the area of robotics. The unsolved problems can only be approached when there is a pure and clear view in computational geometry and by having clear view it can be said as visibility graph of the problem. The visibility of graph varies from levels of implementations and situational considerations. [1]

Mark De Berg (2008) explains that Computational Geometry is mostly implemented in robots. Robots are the automated gadgets those are pre programmed for self decisions and dependency and are also programmed for performing automatic tasks like finding paths and also deciding the shortest paths based upon the requirements. There is always a set of rules programmed into the robot for performing the tasks, those are algorithmic dependent and are different from task to task. [3] Jérémie Chalopin discusses the movement and sensor capabilities that simple robots require to create a path/ map for a known or an unknown region. It is explained as the robot creates an unknown polygon which can be assumed as size $n$. The robots move from one vertex to another vertex by following a particular map and path which might be predefined or sometimes have to back track the movements and find the path and distance vertices. It is explained that during this course the robot has chances of missing the visibility. [2]

The explanation is starting with the working and theoretical elaboration of Dijkstra’s Algorithm in polygonal arena which extends to other shortest route finding algorithms which also considers the least quadratic time taken in the worst case. In very special cases when the robots are left in free space their linear timing is counted. Apart from working explanation this report will also include the comparative studies of the all most every path finding algorithm with inclusion of setbacks in them and in few cases the proposed solutions will also be provided.

The construction of a robot is one challenge and then making it move as per the user requirement another challenge. To plan a motion path for a robot there are few algorithms which allow the movement of robot. The robot will find its path if any existing with the help of algorithm but here the main problem is to find an algorithm that allows motion robot to find the shortest path from all the available paths. In practical it is about finding a good path and not just path.
A good path is always depending on the robot and the task given to it, such as a robot in any industry can travel in straight direction and when it has to take any diversions or turns it will first turn in the same place and then again turn in the direction it has to travel. The turning behavior of the robot for every diversion is time taking process. In order to cover this time taking process a shortest path finding algorithm is much necessary to achieve the required goals in considerably less time. The best recommended algorithm for any shortest path finding is Dijkstra’s Algorithm which is proven for its best time complexity compared to other algorithms. [1,2,3]

LOCAL PATH PLANNING
Robot path planning is generalized in two main sections local path planning and global path planning. Local path planning is mainly dependent on updated local environment information from sensors to access a collusion free path which is independent and optimal. Global is path is difficult to make with the help of local path. The practical approach is in support to local path planning when compared to global path planning because there is frequent update in the local path planning with the embedding of artificial intelligence algorithms in them. [5]

Local path planning consists of few methods which include path planning algorithms based on reinforcement learning, behavioral path planning, rolling window planning, intelligent algorithms and few traditional algorithms. There are few shortest path finding algorithms like Dijkstra’s Algorithm which is explained in brief. [4]

DIJKSTRA’S ALGORITHM
Dijkstra’s algorithm gradually explores to the outward from the starting point to the ending by choosing the shortest path. In the process of algorithm implementation every node is assigned a particular value. These values indicate the weights of the weights of the path which is considered to be short or long based on the weights. These weights are counted from the starting point of the node which can be denoted as P which can
also be mentioned as the upper bound of shortest path, the weight is considered from the starting point; a T label denoted , this label is been modified by the Dijkstra’s Algorithm. The label is gradually modified by the algorithm to satisfy the conditions into P. in every new step a new P is generated and if there are N nodes in the path the shortest path from start point to end point N number P labels will be generated throughout the path. The algorithm helps in finding the number of paths when there are N numbers of nodes the path will be found in N-1 steps. [4,5]

Algorithm steps:

- First step, mark start point with P label: (0, S). 0 indicates that the distance from start point to itself is 0, while S indicates that the serial number of the previous node.

- Second step, mark non-labeled point set as set I and labeled point set as set J. The set of connection sides between two types of points is \{ (v_i, v_j) | v_i \in I, v_j \in J \}. When this set is empty, the algorithm ends.

- Third step, update T label to P label. If node vi is marked with P label: (l_i; k_i), search in set

\{(v_i, v_j) | v_i \in I, v_j \in J \} to find out the side satisfying that l_i + c_{ij} is smallest. c_{ij} is weight of the side.

Mark the end point of the side with P label and return to second step.

Repeat second step and third step until the end point is marked with P label, or the set \{(v_i, v_j) | v_i \in I, v_j \in J \} is empty. Then reverse back from end point, and we can obtain the shortest path seeking for. [3,4,5]

Fig: Motion Path Planning
EXPLANATION
The numbers in the boxes can indicate the present shortest path from p thereto node. Let $d(node)$ denote this distance. First, we have a tendency to initialize $d(s) = 0$ as a result of we have a tendency to travel zero distance to travel from $s$ to $s$, and initialize all different $d(i) = \infty$ as a result of no paths are determined nevertheless. We have a tendency to currently outline a cut (denoted by a dotted line) to be a line dividing the nodes that shortest distances $d(i)$ have already been determined, and we color the particular nodes black. Moreover, we have a tendency to color as grey the node that shortest distance from $s$ we have a tendency are currently crucial.

Fig: Graph

One crucial purpose about Dijkstra's algorithmic program is the one that won't be established here in visibility graph is; Whenever we choose the shortest edge $(a,b)$ across a cut, we say that the shortest distance from $s$ to $b$ has already been found. Moreover, the period of time of Dijkstra's algorithmic program is $O(V^2 + E)$, where $V$ is the range of vertices, and $E$ the quantity of edges. The intuition is that, during a complete graph traveling, after visiting each vertex, there will be consideration of its distance from $V$ all alternative vertices. Therefore, $O(V^2)$. Moreover, all edges are touched once, and there square measure $E$ edges. Therefore $O(V^2 + E)$. Returning to our original notations, where $n$ is the range of vertices, and during a complete graph there square measure $O(n^2)$ edges, Dijkstra's algorithmic program is $O(n^2)$. It is a clear statement of these as facts without proving them. The proofs could be found in textbooks about graph algorithms.

EXAMPLE
The shortest path between $P_{start}$ and $P_{goal}$ among a set $S$ of disjoint polygonal obstacles consists of arcs of the visibility graph
Where: $[1,2,3]$

$$S^* := S \cup \{P_{start}, P_{goal}\}$$
Algorithm SHORTESTPATH(S, P\textsubscript{start}; P\textsubscript{goal})
Input: A set S of disjoint polygonal obstacles, and two points P\textsubscript{start} and P\textsubscript{goal} in the free space.
Output: The shortest collision-free path connecting P\textsubscript{start} and P\textsubscript{goal}.

1. \(G_{vis} \leftarrow \text{VISIBILITYGRAPH}\left(S \cup \{P_{start}, P_{goal}\}\right)\)
2. Assign each arc \((v, w)\) in \(G_{vis}\) a weight, which is the Euclidean length of the segment \(vw\).
3. Use Dijkstra’s algorithm to compute a shortest path between \(P_{start}\) and \(P_{goal}\) in \(G_{vis}\).

Calculating the Time complexity for the shortest path: A shortest path between two points among a set of polygonal obstacles with \(n\) edges in total can be computed in \(O(n^2 \log n)\) time.[1,2,3]

CONCLUSION
The shortest path between two points is found, and also the whole task of automaton motion planning has been solved.

REFERENCES