Lost In JingAn Temple

Problem Solution Algorithm

Xiang Chen

Tsinghua University

MP: 13810306735

Email: chenxiangcyr@sina.com

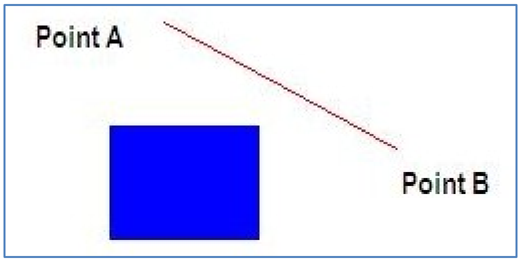
Basic Idea

Concepts

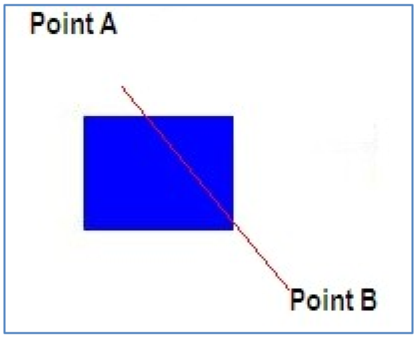
Linked: Point A is linked to Point B if where is a line between these two points which doesn't cross any buildings in the area.

If Point A is linked to Point B, we can also say Point B is linked to Point A.

For example:



In this situation, Point A is linked to Point B.



In this situation, Point A is not linked to Point B.

Transfer Path Problem to Graph Problem

In the area, there are two start locations and one destination location. We use a point to represent each location, so we have three points in the graph now.

For a building, it is a polygon (convex or concave) with some vertexes. We use a point list to represent each building, for example, we use a list of four points to represent a quadrilateral building.

Assume there are N buildings with M vertexes in the area, so we have (3+M) points in the graph now.

Now we need to build a weighted graph, for Point P and Point Q:

(1) If Point P is linked to Point Q, the weight<P, Q> is the Euclidean distance between the two points. Also we know that weight<Q, P> = weight<P, Q>.
(2) If Point P is not linked to Point Q, the weight<P, Q> is a maximum value, which means there is no path directly from Point P to Point Q. Also we know that weight<Q, P> = weight<P, Q>.

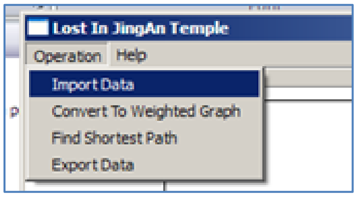
In this weighted graph, we use Floyd algorithm to find shortest path.

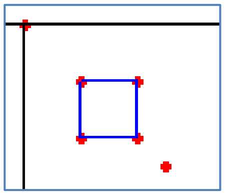
Ref: http://baike.baidu.com/view/14495.htm

Whole Process

In order to show the whole process clearly, I just use an example data set.

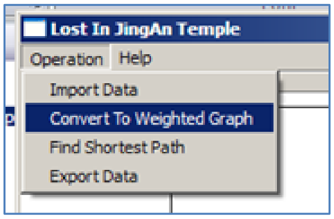
Step 1: Load points of buildings, start point and destination point.

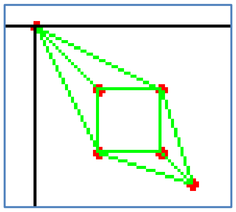




(Points in red and walls of building in blue)

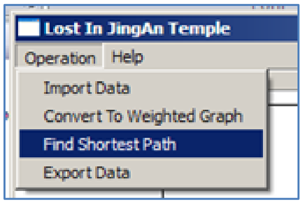
Step 2: Draw weighted graph.

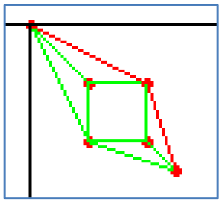




(Points in red and graph edges in green)

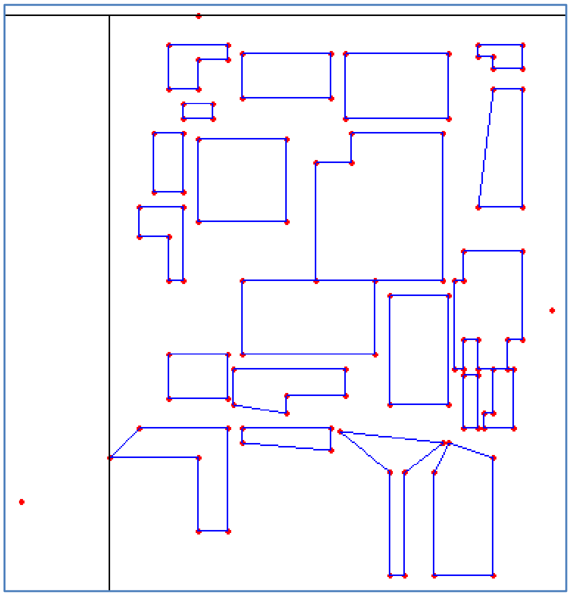
Step 3: Find shortest path.

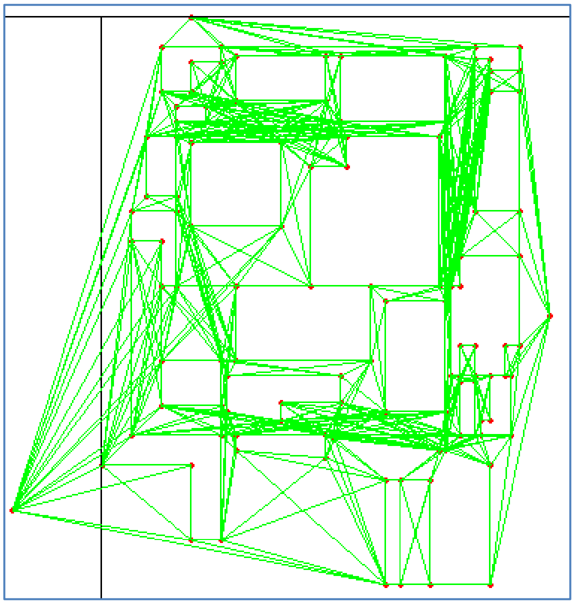


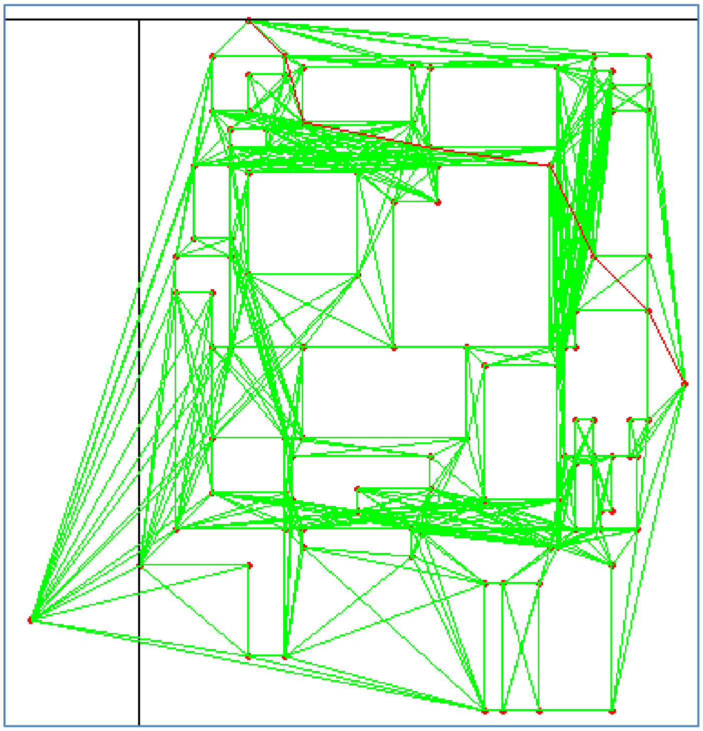


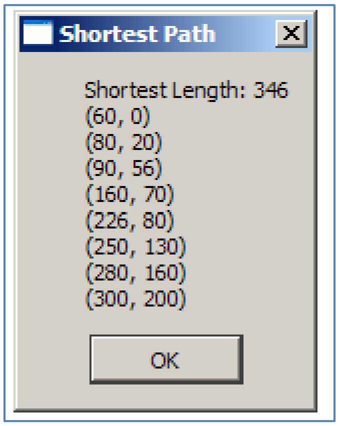
(Points in red and shortest path in red)

Result of “Lost in JingAn Temple” Problem









Shortest Length: 346
(60, 0)
(80, 20)
(90, 56)
(160, 70)
(226, 80)
(250, 130)
(280, 160)
(300, 200)

Shortest path:

(60, 0) -> (80, 20) -> (90, 56) -> (160, 70) -> (226, 80) -> (250, 130) -> (280, 160) -> (300, 200)

Length: 346

Program Structure

I used two main techniques to develop this program.

1. Qt, a foundation class framework to build GUI.
   Ref: http://qt.nokia.com/
2. OpenGL, a C++ library to draw 2-D or 3-D model.
   Ref: http://www.opengl.org/

Github Repository: https://github.com/chenxiangcyr/LostInJingAnTemple

Code comments

In order to help readers understand the code, I list important files and methods below.

Table 1 Main Files

main.cpp	Program rooter, initialize class LostInJingAnTemple
glwidget.h	Define basic data types and declare basic variables
glwidget.cpp	Draw 2D model
lostinjingantemple.h	Define basic GUI
lostinjingantemple.cpp	Implement program core algorithm

Table 2 Main Methods in lostinjingantemple.cpp

// Compute distance between two points float distanceBetweenTwoPoints(Point point1, Point point2);
// Check if the two points is on the edge of the building bool onTheBuildingEdge(Building building, Point point1, Point point2);
// Check if line<point1, point2> is the edge of the building bool isTheBuildingEdge(Building building, Point point1, Point point2);
// Check if the line<point1, point2> crosses the building bool crossTheBuilding(Building building, Point point1, Point point2);
// Check if the line<point1, point2> crosses any building of the building list bool crossTheBuildings(vector<Building> buildings, Point point1, Point point2);
// Check if the line<point1, point2> crosses with line<point3, point4> bool crossLines(Point point1, Point point2, Point point3, Point point4);
// Check if the two points represent the same position bool samePosition(Point point1, Point point2);
// Check if the point is inside the building bool isPointInsideBuilding(Building building, Point point);
// Check if the line<point1, point2> is inside the building bool isLineInsideBuilding(Building building, Point point1, Point point2);
// Check if the line<point1, point2> is fully inside the building bool isLineFullInsideBuilding(Building building, Point point1, Point point2);
// Check if the line<point1, point2> parallels with the line< point3, point4> bool isTwoLinesParalleled(Point point1, Point point2, Point point3, Point point4);
// Check if the point3 is on the line<point1, point2> bool isOnLine(Point point1, Point point2, Point point3);