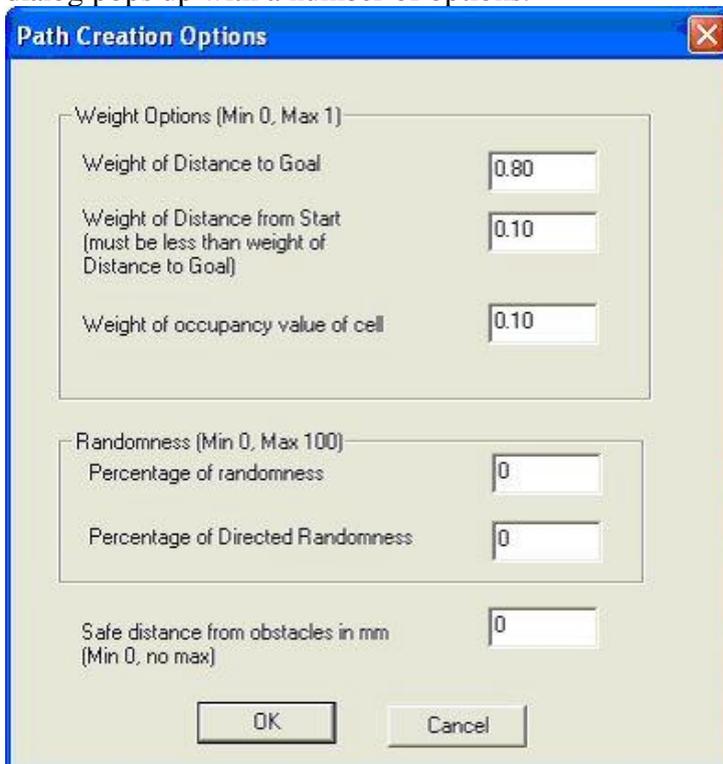


## MapViewer Tutorial 2 – Using Paths And Voronoi Diagrams

### Path Planning with MapViewer

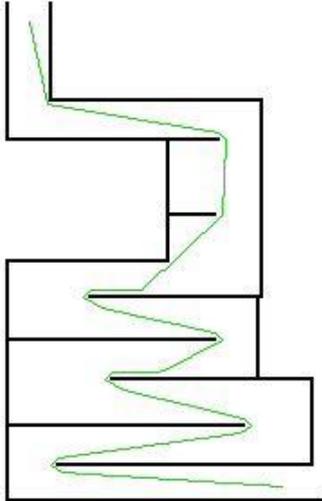
MapViewer can generate paths between two points using a modified version of the A-Star path planning algorithm. To do this, first load a map, either grid map or vector map, and click on the Create Path  button. Now click any two points on the map. If you've clicked on an occupied part of the map for either the start point or the end point, the path creation will fail. Otherwise, a path is created between the two points.

It is possible to change how the path is created by altering how it weights different factors in its search algorithm. To do this, click on the Path Options button . A dialog pops up with a number of options.

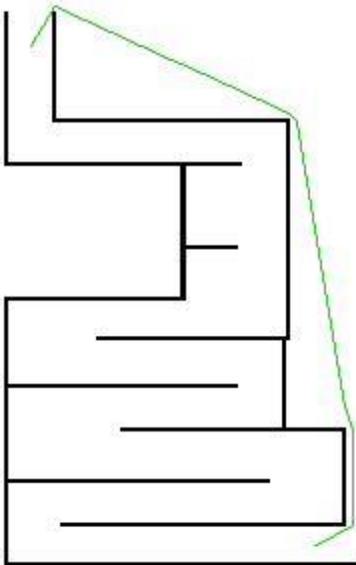


The top three options specify the three weights that the search algorithm uses to decide in what direction to search. The higher the weight, the more that factor affects the final path.

The first weight, Weight of Distance to Goal, says that one direction is better than another if it is closer to the desired end of the path. This is a good metric to use in relatively simplistic environments, as it will take you more or less straight to the goal point. However, it has a tendency to get stuck in local minima. For example, in the picture below, you can see a path created in a map, going from the top to the bottom, with “Weight of Distance to Goal” set to 1, and the other two weights set to 0.



It would have been better to go north first, then south, but going north would move away from the goal, so that direction is never searched. However, by increasing the weight of the “Weight of Distance from Start”, which decreases the fitness of a path if moves farther from the start point without getting closer to the goal, the path created becomes more optimal, as you can see below.



However, the higher the “Weight of Distance from Start” weight, the more processing required to generate the path, since more directions are searched, and in simplistic maps, this is not needed.

The third weight, “Weight of Occupancy Value of Cell”, specifies how much the path planning algorithm will try to stay away from cells with higher occupancy values in order to avoid collisions with obstacles.

You can also choose the percentage of randomness in the search algorithm. This specifies how often, rather than searching what seems like the best direction, the algorithm just picks a direction at random. The higher this value, the higher the processing load, but it can be useful in complex environments with lots of local minima (directions that seem to be the best, but there are better alternatives close by).

The percentage of directed randomness is a measure of how often, once a random direction is chosen, that direction is continually followed, rather than choosing another random point. This can lead to more effective random behaviour.

The final setting, “Safe Distance from Obstacles”, specifies how close the robot can get to obstacles. This is usually a little more than the radius of the robot, just to be safe.

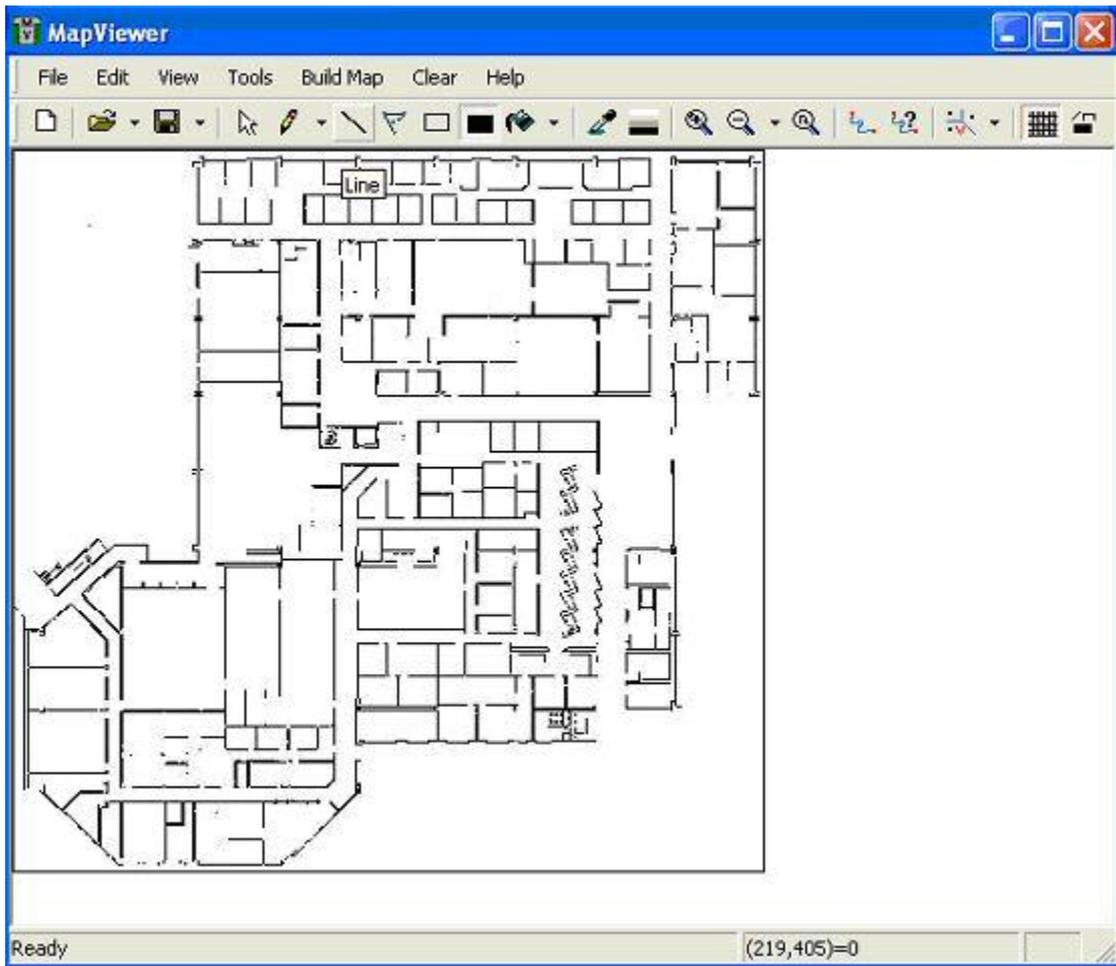
### **Creating Voronoi Diagrams With MapViewer**

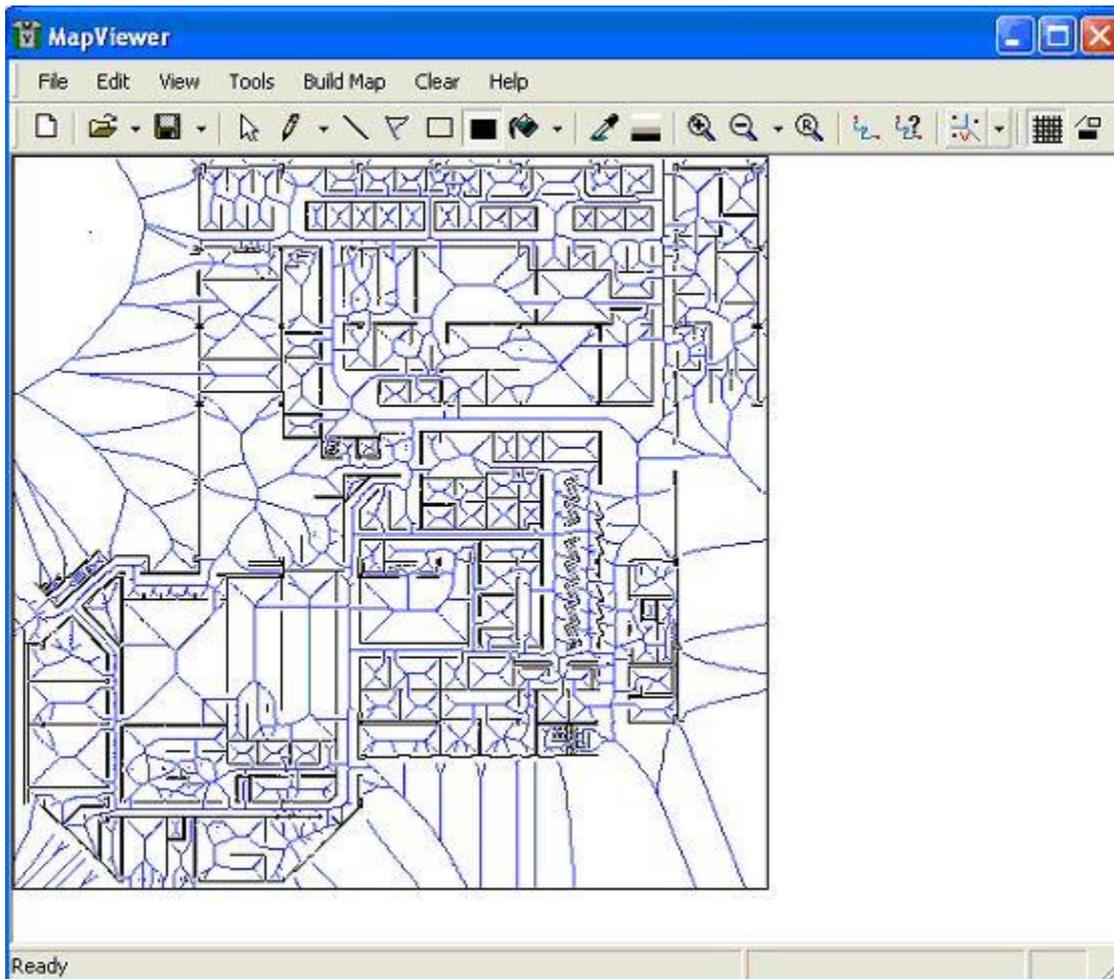
MapViewer can create Voronoi diagrams in grid or vector maps. A Voronoi diagram is a graph where represents all points on the map that are equidistant from their two nearest occupied cells. For lots of detailed information of Voronoi diagrams, go to the website <http://www.voronoi.com>, or read the masters thesis of the author of MapViewer, Shane O’Sullivan, at <http://www.skynet.ie/~sos> .

Voronoi diagrams are used in many different fields, from character recognition, planning mobile telephone network layouts, designing road and rail networks, as well as mobile robotics. MapViewer uses Voronoi diagrams internally for a few different purposes, including two grid map benchmarking techniques and for converting a grid map into a vector map with a line fitting algorithm. However, to simply view the Voronoi diagram of a grid map, follow the following steps.

1. Load the map that comes with this tutorial, *oldlibrary.map*, by clicking on Load  button and choosing the file.
2. Click on the Generate Voronoi  button.
3. A dialog box pops up asking you for the range of cell values that will be considered to be obstacles. Leave it at the default values for now. Click OK. After a few seconds, the screen will change, and you’ll see the Voronoi diagram of the map.

The first picture is MapViewer with just the map, the second picture is the map with the Voronoi diagram generated in it.





All of the blue lines represent all points in free space that are equidistant from their closest occupied cells.

It is possible to alter the Voronoi diagram by changing the minimum distance between two occupied cells. This specifies how far apart two occupied cells must be (in millimetres) in order to have an edge of the diagram placed between them. To change this, click the down arrow  next to the Generate Voronoi button . The dialog box prompts you for the minimum distance between obstacles. You cannot set the value to less than 1.5 times the width of one grid cell, as that would result in a Voronoi edge being placed between every two occupied cells (which would be correct if the map were theoretically a discrete set of points where two adjacent occupied points did have empty space between them. However for grid maps, two adjacent cells are not discrete points, they each represent a square area, and therefore have no unoccupied space between them).