Tutorial 4 – Building Maps From Sonar Data With MapViewer

MapViewer has the ability to take raw sonar range data and position data of the robot, and convert it into a grid map. The methods used are detailed in the masters thesis of MapViewer's author, Shane O'Sullivan, which is available from his website at http://www.skynet.ie/~sos . Some of the theory is based upon work done previously by Kurt Konolige, while other parts are base upon algorithms devised by O'Sullivan. See sections *2.8.3* and *4.1.7* specifically.

Currently (Fecbruary 2005) two file formats are supported. The first, MapViewer log format, is specific to MapViewer (it is detailed below). To build a map from a file of this format, click on "*Create Map\Build Map From Sonar Data - Map Viewer File Format*" in the menu bar, and select the file. The second log file type supported is that used by the Player/Stage simulator. To build a map from a log file using this format, click on "*Create Map\Build Map From Sonar Data – Stage File Format*" in the menu bar, and select the file.

Once the file is selected, you will be asked for a filtering value. This determines how inaccurate a sonar reading must seem to be discarded. The higher the value, the more readings will be discarded. The filtering algorithm is also described in the author's thesis, as well as in papers available from his website. Choose your filtering level, click OK, and after a while – this can take some time as it can be very CPU intensive – a grid map will appear on the screen. You can cancel the operation at any time by clicking the Cancel button.

Sonar data can be quite inaccurate, and suffers from a lot of interference and noise for a number of different reasons which won't be explained here. See the thesis referenced above for more information. Because of this inaccuracy, maps generated from sonar data are often quite inaccurate. However, quite a lot of effort has been put into compensating for this noise, so the map generated should be a relatively accurate reconstruction of the environment in which the data was collected.

The only assumption about the file is that each individual record in it (robot position + sonar readings from that position) was recorded in the file after the one that preceded it, i.e. that the file was written in the same sequence in which the readings were received.

MapViewer File Format Explained

The file begins with the word "*robotrun*" to identify it as a file that records a test run of a robot. Following this must be the number of the sonars on the robot in question. This is done using the term "*numsonars*" and the number, for example,

numsonars 7

for seven sonars. Next, the layout of the sonars must be specified, followed by the word "*end*". This means the Cartesian position and angle that each sonar is facing. This is required so that when a range reading is received from, say sonar 2, it is important to know that sonar 2 is at the point (130, 40) from the centre of the robot, and facing at an angle of 15^{o} different to the angle that the robot is facing.

For example, the layout of the sonars on an ActivMedia Pioneer 1 robot are as follows:

```
numsonars 7
start sonar
100 100 90
120 80 30
130 40 15
130 0 0
130 -40 -15
120 -80 -30
100 -100 -90
end
```

On each line, the first two numbers represent the Cartesian coordinates of the sonar in relation to the centre of the robot (for simplicity the robot is assumed to be circular), and the third number is the angle that the sonar is facing in relation to the angle that the robot it facing. The robot is assumed to be aligned with the X axis when looking at these readings, so the first line states that sonar 0 is facing north, or 90^{o} from the X axis in an anti-clockwise direction. Sonar 6 is facing south, and so on.

Finally we come to the meat of the file – the sonar range reading data. To begin this section, the term "*DATA SONAR*" is printed in the file. Following this, each line contains a series of numbers. The first three numbers are the pose of the robot (the X, Y and angle of the robot). The number of figures following the pose must be the same as the number of sonars declared earlier (7 in the example given) with the *numsonars* term. Each number is the range read by the particular sonar when the robot was in the position and facing the angle stated in the first three numbers.

For example, if we only had two sets of readings (not nearly enough to build any kind of map), it would look something line below (with 7 sonars – hence 10 numbers per line, 3 for the pose, then one per sonar).

```
DATA SONAR
0 0 0 870 2349 2500 2500 2500 2500 926
1 1 0 870 2349 2500 2500 2500 2500 926
END
```

It ain't all that hard……

For simplicity, there are a number of example robot run files included with this tutorial. I've also included a C++ class that will both read and write these *Robot Run* files, called RobotRunFileHelper, in case you'd like to incorporate files of this type into your own application. An example usage is also included in driver.cpp.

Player/Stage File Format

See the official Player/Stage file format for an explanation of the log file format used by that simulator.