CS22510 Building occupancy grids

Michal Wojciech Goly [mwg2]

18th March 2016

1 Design and implementation

1.1 Initial approach

I approached this assignment slightly differently than I usually do. Because I have never programmed in object oriented C++ before, instead of jumping straight into writing code, I begun by going through online tutorials to quickly learn the new concepts. I realised that because this assignemnt was relatively small, I could rapidly create a working prototype without using any of the object oriented features.

I took a piece of paper and tried to calculate the coordinates of the first obstacle myself, before hard coding it into my program. I wrote some utility functions, which could be used for all the conversions and maths necessary to calculate the position of the obstacle in metres, along with conversion mechanism to indices. I took into account the 0.1 metres offset in the grid, and after successfully calcualting the first occupied cell, I moved on to reading data from the source files provided. I decided to store both poses and ranges in an appropriate 2D vectors of doubles. I also stored sensor angles in a separate vector, which made it possible to iterate over all the vectors and calculating obstacle positions in the grid. I would ignore any range of 2.5 metres or more, as it refered to infinity reading of a sensor in the assignment specification. Finally, I marked occupied cells as asterisks in the previously created matrix of chars initialised with spaces.

1.2 Object oriented solution

Even though the prototype described above did the job, I wanted to take advantage of the object oriented features of C++. I also thought about making my approach of marking cells a bit smarter by using probability values, instead of binary true or false. Finally, I wanted to make the presentation a little bit nicer to look at, therefore I decided to use the allegro5[1] library to create the GUI.

I identified 3 classes I could split my code into:

- `GridFrame` which would take take of the GUI and initialisation of the other objects in the system.
- `Grid` which would represent the 50x50 grid in the system.

- `Robot` which would hold the information about the poses and ranges readings, along with the sensor angles data.

I also decided to move all the utility functions into a separate namespace `converter`, because I have done some research on the Internet, and apparently there is no point in making a class with only static member functions in C++. Because it is a multi-paradigm language I thought a separate namespace made more sense.

As mentioned before, I tried to use probability values instead of spaces and asterisks to mark potentially occupied cells. I decided that all the cells will start with a probability of 5. This value could be then decreased or increased as required by appropriate functions later in the program, up to the minimal and maximal values of 1 and 9 respectively. After each subsequent reading, I wanted to increase the probability of the cell containing the obstacle, and decrease the probability of all the cells between the robot and the current obstacle. I achieved that functionality using the Bresenham line drawing algorithm[2], which I modified to create a vector of pairs of indices of all the cells between to cells specified.

Finally, I really enjoyed using the allegro5 library to create the GUI for the program. I managed to create a 500x500 pixels window with a title, presented in an approrpiate position on the screen. In order to create an animation, I paint the grid within the window for each row in the input files. Different shades of grey colour are used to indicate the probability of given cell being occupied. White means cell is unlikely to be occupied, whereas black means there is a very high possibility given cell is occupied. I also use an appropriate event queue in order to make sure the user is able to close the window whenever he wants to. Animation stops at the end and waits for the user to exit the window.

References

[1] http://liballeg.org/ [Accessed: 14th March 2016]

[2] http://tech-algorithm.com/articles/drawing-line-using-bresenham-algorithm/ [Accessed: 14th March 2016]