

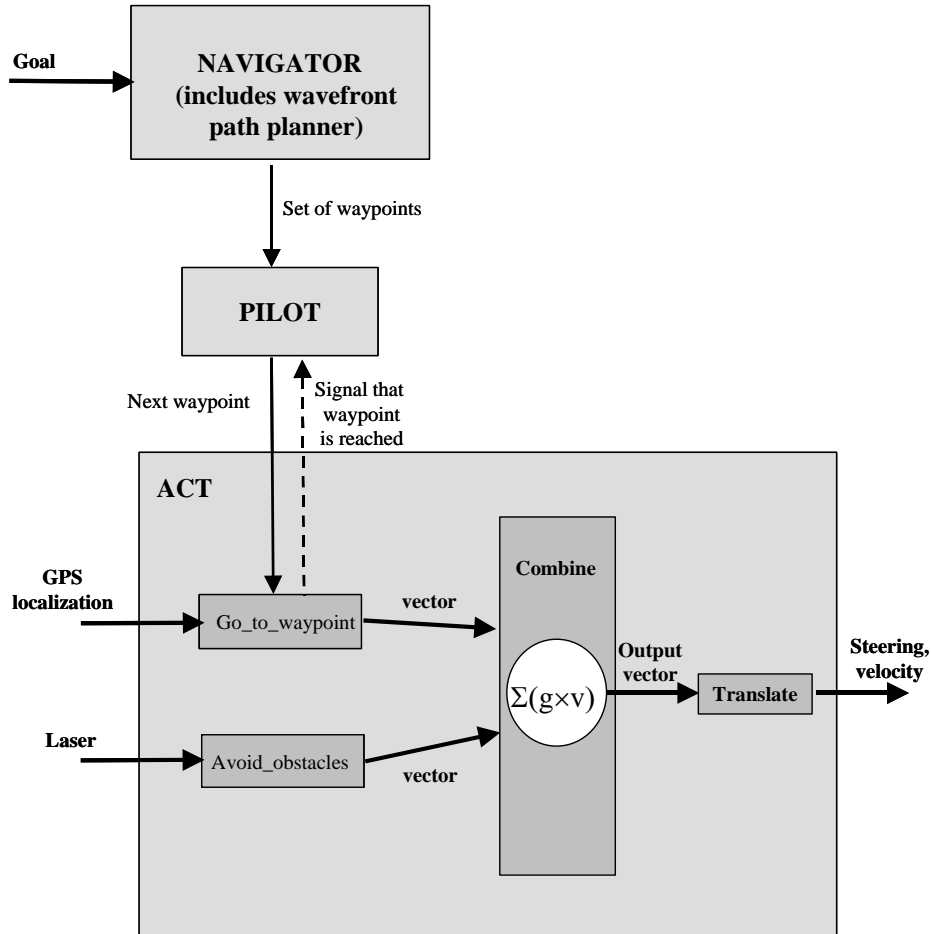
## Homework 4: Path Planning and Execution

**Assigned: Thursday, March 1, 2007**  
**Due: Friday, Mar. 30, 2007 at 23:59:59**

In this assignment, you will write a path planner that makes use of wavefront propagation, and interface the planner with the avoid obstacles and “go to waypoint” behaviors you have already written for previous assignments. The result will be a sort of hybrid architecture, which has a higher level path planner, combined with lower-level motor schemas for avoiding obstacles and going to a waypoint.

(Note: Player includes a Planner proxy. However, the purpose of this assignment is for you to gain experience writing your own planning software. So, you are not allowed to copy or make use of the existing planner proxy software. Instead, you must write your own wavefront planner using the algorithms outlined in class on Feb. 27.)

The structure of your robot software should be as shown in the figure below.



Note that the overall structure of this software is a combination of your Assignment #2 hierarchical approach and your Assignment #3 motor schema approach. This effectively makes this new software organization a *hybrid* architecture.

In this software solution, you should re-use your general hierarchical organization from Assignment #2, but with some changes, as follows:

- You accept the goal position from the user using standard input. This should be done in your main function, with the goal position passed to the navigator function. Your user interface should make it clear how the user should enter the goal (x,y) position. The exact format is up to you. You should do some modest error checking to make sure the goal position entered is numeric, and within the bounds of the robot's environment.
- You extend your "Navigator" module to include the new wavefront path planning method. (You can either include this path planning as part of the Navigator function, or make it a separate method called from within the Navigator function.) The output of the Navigator function is the same as before – a series of waypoints the robot should visit to reach the goal position.
- You extend your "Act" module to include the motor schema combination of vectors from the "avoid obstacles" and "go\_to\_waypoint" behaviors, along the lines of your Assignment #3 solution. You should be able to reuse your behaviors, vector combination, and translate functions from Assignment #3. (However, you may find you have to dampen the avoid obstacle output vector, to allow your robot to go through doors. See a further discussion of this below.)

**Inputting the map.** Obviously, for the path planner to function correctly, it needs to load the map of the environment into memory. For this assignment, we'll use the "hospital\_section" map. At present, we have a bitmap version of this map in "png" format. However this format makes use of data compression, and isn't the easiest for you to read in directly into memory for the purposes of this assignment. So, we'll convert the bitmap to another format. We have several to pick from; we'll choose the "pnm" format. Posted on the class website (under Homework Assignments) will be this map in pnm format, as well as a function you can use to read in the map into a 2-dimensional binary occupancy grid array, in which 0's represent free space and 1's represent obstacles. Note that you will still use the "png" format for Stage display purposes; this "pnm" format is just for your code to have access to the map for planning purposes.

**Growing the map for obstacles.** You need to write a function that grows the obstacles in the grid map, so that your path planner can represent the robot as a point. This obstacle growth function must input a parameter representing the amount of growth desired, in units of the number of grid cells. Note that you will find there is a tradeoff between growing the obstacles too much and growing them too little. If you grow them too much, your path planner may not be able to plan a path to a reachable goal. If you grow them too little, the robot may come too close to obstacles.

**Wavefront path planner.** Once you have your map in memory, and have grown the obstacles, you are ready to implement the path planner. Your path planner must make use of the wavefront path planning algorithm that we discussed in class on Feb. 27. You should start your wave propagation from the *goal* location, back to the start. (Later, after you get the single wave propagation working, you are welcome to implement a *dual wavefront* approach, where you propagate two waves from both the start and goal locations. However, this is not required. The point here is that if you want to implement the dual wavefront, that's terrific. But, if you don't, it isn't required; only the single wavefront is required.)

Once you generate a path, you'll want to smooth it using the approach described in class. The grid cells that remain on the robot path become waypoints. The output of your wavefront path planner should be this series of waypoints to be reached by the robot.

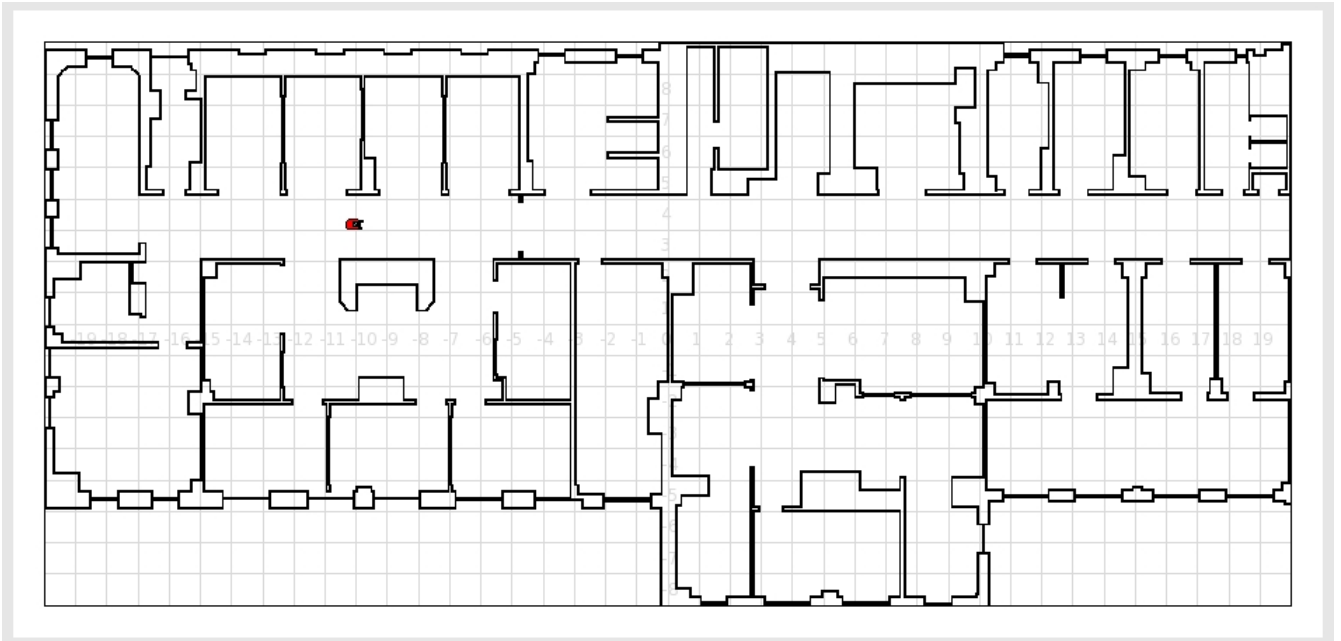
Note that if your path planner is given an unreachable goal position, it should state this fact and gracefully exit. A goal position could be unreachable because it is inside an obstacle, or it could be behind a wall, with no access. Your code should be smart enough to recognize this, and let the user know.

**A note on doorways.** A very tricky aspect of robot navigation in indoor environments is enabling the robot to move through doorways. You don't have a lot of room for error, and if your obstacle avoider is too sensitive, the robot may not be able to go through doorways, even though the robot should easily fit through the passage. This will be a tricky aspect of tuning your code so that the robot can follow the path generated by the path planner. Your code should work for any passage that is at least twice the width of the robot. Smaller passages may not work; you won't be penalized for that.

**Setting up everything.** In this exercise, you must use the bitmap called "hospital\_section.png". Define your window as follows:

```
window(  
  size [950 500 ]  
  center [0 0]  
  scale 0.045  
)
```

Your robot should be set up as it was in Assignment #2, with the same starting pose: [-10.071 3.186 -722.333]. If you have everything set up properly, your environment should look like the following:



**Testing your code:** Your path planner should work for any goal position given; in the case of unreachable goal positions, as previously stated, your planner should recognize this and gracefully exit. Test your code for lots of different goal positions. Then, create screendumps of your robot's path to the following specific goal positions (i.e., after the robot follows the path generated by your wavefront planner):

- (7.5, 5.5)
- (8.5, -4.0)
- (-18.5, 7)
- (-9, -4)

**WRITE UP THE FOLLOWING (written up in a single pdf file called *yourlastname-HW4.pdf*):**

- a) A discussion of issues you ran into in getting your robot's behavior to work properly. Particularly, I'm interested in your experience with the robot moving through doorways, and how you tuned your code to make it work. Another issue you may run into is the obstacle avoidance behavior knocking the robot off its path, and then the robot having difficulty picking back up the path. If so, please discuss how you handled this issue.
- d) 4 screenshots of your robot moving along the planned paths to the 4 goal points mentioned above. Be sure to turn on the robot trace.

**SUBMITTING YOUR HOMEWORK:**

Place all your files in a single directory. These files should include:

- Your written answers to the questions above, plus screen dumps as requested (i.e., the file "yourlastname-HW4.pdf")
- Your configuration file, called "HW4.cfg"
- Your world file, called "HW4.world"
- Your makefile, called "makefile" or "Makefile"
- Your robot control code, called "yourlastname-HW4.cc" .
- Any additional include files you created (called whatever you want them to be called).

Remove all other unnecessary files.

Use the submit script **594sir\_submit** to submit your files. (These will be emailed to Dr. Parker.)

[When I unpack your files, I should be able to say "make yourlastname-HW4", and then "player HW4.cfg", and then "./yourlastname" to run your code. Be sure you have everything set up properly so that this works as expected.]