CSCI–4190 Introduction to Robotic Algorithms, Spring 2004
Assignment 1: out January 15, due February 2

Motion planning

In this assignment, you will implement a simple motion planner using an approximate cell decomposition approach with a uniform grid representation. There will also be some written questions regarding your implementation and on related motion planning topics.

Your program should do the following:

- Read the problem description from files.
- Display the workspace obstacles and boundary.
- Respond to user commands to step through the motion planning process.

I will supply a sample program to get you started; it will depend on the following libraries:

- the "filereader" library — for reading the problem description from files
- the "CGAL" library — for geometric representations and operations
- the "dolt" library — for graphical display; this library makes use of the OpenGL and GLUT libraries.

These libraries will be installed in the CS department Sun computing environment. If you are working on your own computer, you will have to install them; this should be fairly straightforward on most UNIX systems. See the Assignment 1 web page for links to these libraries.

You will turn in your code electronically. Remember that your program must compile and run on the CS department Suns.

Assumptions

We will make the following assumptions in this assignment:

- The robot cannot rotate, so its configuration may be specified by the two dimensional vector (x, y).
- The robot will be modeled as a convex polygon.
- The world boundary will be convex (but not necessarily rectangular).
- All obstacles will be modeled as convex polygons, and they may overlap.

User interface

Here is how a user should be able to interact with your program:

1. When the program starts, it should display the workspace and the start and goal locations. Typing "w" at any subsequent time should toggle the display of the workspace.
2. Type "c" to compute and the configuration space and add it to the graphical display. Typing "c" at any subsequent time should toggle the display of the configuration space.
3. Type "r" to compute and display the initial decomposition, a single cell. Once the initial decomposition is displayed, the user can do any of the following:
   - Type "r" to compute and display a refinement of the decomposition. FULL, EMPTY, and MIXED cells should all be different colors and should use transparency so that the underlying c-space obstacles can be seen.

     If your motion planner decides that it is finished, it should then (automatically) plan a path in the best E-CHANNEL and display the E-CHANNEL and the path.

- Type "e" to toggle display of the best E-CHANNEL found in the current decomposition (if any).
- Type "m" to toggle display of the best M-CHANNEL found in the current decomposition (if any)

The E-CHANNEL and M-CHANNEL should be displayed using color to highlight the cells in the channel.

4. At any time after the initial decomposition, the user may type "p" to make your program plan a path through the current best E-CHANNEL (if one exists). The E-CHANNEL and path should be displayed.

   After your motion planner has refined the approximate cell decomposition enough, pressing "r" again should have the same function as initially pressing "p" (i.e., causing a path to be planned).

   Subsequently pressing "p" should toggle display of the path.

A few additional notes:

- The user may type "h" at any time to print a help message to `cout` that describes the key bindings (and your color scheme).
- The user may type "q" to quit at any time.
- Your program should print a narrative (to `cout`) that describes what it is doing during the refinement phase.

Creating the configuration space representation

The problem of creating C-space obstacles is simplified by our assumption that the robot and workspace obstacle are convex. We discussed a method that creates a configuration space obstacle in time linear in the number of sides of the workspace obstacle and of the robot. This method, however, is harder to implement. I will award a few bonus points if you implement this linear method.

A much simpler way to create a configuration space obstacle (at least when you have a library of geometric algorithms) is to use the convex hull algorithm. However, the convex hull algorithm is an $O(n \log n)$ algorithm. If we assume that the reference point on the robot is the origin of the robot frame, then we can compute the configuration space obstacle by the following algorithm:

- Let $\mathcal{P}$ be a list of points, initially empty
- For each vertex of the obstacle, $\vec{O}$ with respect to the world frame:
  - For each vertex of the robot, $\vec{R}$ with respect to the robot frame:
    - Add the point $(\vec{O} - \vec{R})$ to $\mathcal{P}$
- Compute the convex hull of the points in $\mathcal{P}$

Note that the workspace boundary is different than an obstacle, so you will need to create its configuration space representation differently.

Uniform grid decomposition motion planning

Your initial decomposition of the configuration space will consist of a single rectangular cell. As you refine the decomposition, you should split each cell into 4 identical subrectangles.

At each iteration, all cells (i.e. rectangles) should be "labeled" as one of:

- EMPTY if it is free of configuration space obstacles,

- FULL if it is contained entirely within a configuration space obstacle, or
- MIXED if it intersects part of a configuration space obstacle.

The objective in motion planning with approximate cell decompositions is to find a *channel* of adjacent cells from the start cell (i.e. the cell containing the start configuration) to the goal cell. Ultimately, we want to find a channel that consists of only EMPTY cells; such a channel is called an E-CHANNEL. However, we are also interested in M-CHANNELs (a channel consisting of MIXED and EMPTY cells) because an M-CHANNEL may become an E-CHANNEL with additional refinement.

For this assignment, use the following basic algorithm:

1. Create an initial decomposition of a single cell.
2. Search the adjacency graph for the shortest E-channel from the start to goal cells.
3. Search the adjacency graph for the shortest M-channel from the start to goal cells.
4. If there is no E-channel or M-channel, return failure. (There is no path from the start to goal locations.)
5. If there is only an E-channel then go to step 8.
6. If there is only an M-channel then go to step 9.
7. If the M-channel is shorter than the E-channel, then decide whether to:
   (a) go to step 9 to investigate the M-channel further, or
   (b) go to step 8 to keep the E-channel found.
8. Generate and return a path in the E-channel from the start to goal location.
9. If the maximum decomposition resolution has been reached, return failure.
10. Otherwise, create a higher resolution decomposition, label each cell, and create the adjacency graph of EMPTY and MIXED cells.
11. Go to step 2

An appropriate maximum resolution for your decomposition is when cells are first smaller than 1/20 the size of the robot. (Sometimes, it's appropriate to use some absolute size.)

Searching the adjacency graph

Use the A* search algorithm to search the adjacency graph to find the shortest path from the start to the goal configuration. I suggest the following formulation:

1. Let OPEN be a list initially containing the start node
2. Let CLOSED be a list, initially empty
3. If OPEN is empty then return failure
4. Remove the node on OPEN with minimum $f()$, let this node be N
5. Add N to the CLOSED list
6. If N is the goal node, the return success
7. For each child (i.e. neighbor) N' of N:
   a. If N' is on the OPEN list, then update its $f()$ value if necessary
   b. If N' is on the CLOSED list, then do nothing
   c. Otherwise, add N' to the OPEN list
8. Go to step 3

A* is a best-first heuristic search, where "best" means minimum estimated cost to reach the goal. This cost is represented by the $f()$ value of a node:

$$f(n) = g(n) + h(n)$$

where $g(n)$ is the cost to reach node n from the start node and where $h(n)$ is an estimate of the distance remaining to reach the goal from n. The above algorithm assumes that the heuristic function $h()$ is *admissible* and *monotonic*. An admissible heuristic will never overestimate the distance to the goal; most admissible heuristics are monotonic. (If the heuristic is not admissible and monotonic, then step 7 must be changed so that if N' is on the CLOSED list, it must be removed put back on the open list if the new $f()$ value is lower than the old $f()$ value.)

You should use the straight line distance from the node to the goal as your heuristic. This heuristic is admissible because the actual robot path cannot be shorter than the straight line distance. Any heuristic (such as straight line (or Euclidean) distance) that obeys the triangle inequality is monotonic.

Generating a path

After you have found an E-channel, you must still generate a path for the robot. There are a number of ways to generate a path, some simpler than others and some that produce a shorter path than others. In general, these approaches produce a path consisting of a sequence of line segments.

You may use any approach you wish to generate a path. Two simple approaches are:

- Connect the centers of consecutive cells. Connect the start and goal configurations to the centers of the start and goal cells.
- Connect successive midpoints on the shared edge between two consecutive cells. Connect the start and goal configurations to the first and last midpoint.

Written questions

1. Describe the significant details of your motion planner implementation, in particular:
   - how you created the configuration space representation of the boundary
   - the function used in your search to determine the cost/distance between two cells
   - how you made the decision in step 7 of the basic algorithm
   - how you generated a path within an E-channel
   - any other relevant details
2. Why aren't approximate cell decompositions optimal?
3. Suppose a FULL cell is sometimes mistakenly labeled MIXED. Is the approximate cell decomposition motion planning still correct? Is it still complete?
4. Suppose a MIXED cell is sometimes mistakenly labeled FULL. Is the approximate cell decomposition motion planning still correct? Is it still complete?
5. Suppose you had to modify your program to plan motions for a robot that can rotate. What would you have to change in your program? Describe and sketch solutions to the main problems you would encounter.