CS26N
Motion Planning for Robots, Digital Actors and Other Moving Objects
(Winter 2012)

Homework #3
Posted: February 22 – Due: February 29

How to complete this HW: First copy this file; then type your answers in the file immediately below each question; finally print this file and return it to the Wednesday class on the day indicated above. If you need to draw anything, you can do it by hand.

Your name: ……

Your email address: ……

Note: You may discuss this problem set with other students in the class (in fact, you are encouraged to do so).

Grading:

Question#	Max. grade	Your grade
I	20 (7 + 7 + 6)	
II	40 (12 + 20 + 8)	
III	40 (20 (6 + 6+ 8) +20 (12 + 8))	
Total	100	

Problem I. Navigation Function

Consider a robot's configuration space that has been discretized into a uniform *n*-dimensional grid (where *n* is the dimensionality of the configuration space). Let us define the neighbors of a node *q* in this grid to be the 2*n* nodes obtained by moving by ± one increment along a single axis of the grid. Let G_{feas} be the feasible subset of the grid (e.g., the subset where the robot is collision-free).

A "navigation function" NF is a non-negative function defined over G_{feas} that has a single minimum (0) at the Finish configuration (assumed to be a grid node) and such that any node $q \neq$ Finish in G_{feas} has at least one neighbor q' in G_{feas} is such that NV(q) > NV(q').

1. Assume you are given a navigation function NF. Given a Start node in G_{feas}, how can you use this function to construct a path to reach Finish?

2. Assuming that the grid is finite, give a simple algorithm to construct a navigation function.

3. Assume that you want to generate the paths of a large number of robots that start at different Start nodes but must all reach the same Finish node. No two robots can be at the same node at the same time, except at the Finish node. How could you use a navigation function NF to generate the robot paths?

Problem II: Reaching a goal despite uncertainty in control and sensing

A robot represented by a point moves in an infinite uniform two-dimensional grid. The coordinates of a grid node are (*i*,*j*), where *i* and *j* are two integers in **Z** = {..., -2, -1, 0, +1 , +2, ...}. See Figure 1.

The "belief state" of the robot at a given time is the set of all locations (*i*,*j*) where the robot thinks it may be with non-zero probability. We assume that the robot is always in one of the positions listed in its belief state. So, for example, a belief state can be {(-1,0), (0,0), (0,+1)}, meaning that the robot thinks it may be at (-1,0), or at (0,0), or at (0,+1). Its actual location is then any of these three locations.

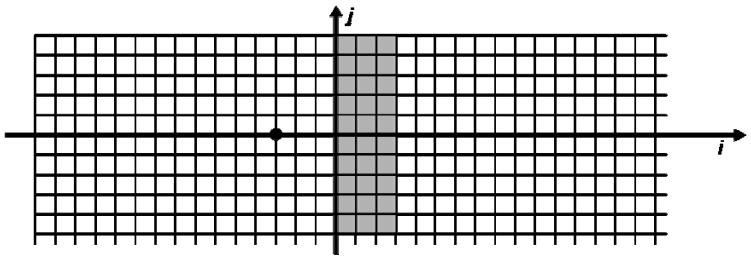


Figure 1: Grid for Problem II. The gray region is the goal region for Questions 2 and 3. The bold dot at (-3,0) is the robot's initial position in Questions 1 and 2.

For this problem, we assume that the robot can execute only motion command, **Right**. The intended effect of this command is to make the robot move right by one increment. But robot control is not perfect, so that there is uncertainty in the outcome of this action.

The uncertainty model (given to the robot) of the action **Right** is the following: by executing **Right** from its current location (i,j), the robot will actually move to $(i+1,j)$, $(i+1,j+1)$, or $(i+1,j-1)$, or stay at (i,j). Each outcome has non-zero probability, but this probability is unknown. The robot can perform the action **Right** as many times in a row as it wants.

The robot has a position sensor, but this sensor is also imperfect. The model of sensing uncertainty (again, given to the robot) is the following: if the robot is at (i,j), then the sensor returns anyone of the 5 positions (i,j), $(i-1,j)$, $(i+1,j)$, $(i,j-1)$, and $(i,j+1)$, each with a non-zero (but unknown) probability.

1. Let the initial belief state of the robot be $\{(-3,0)\}$. So, the robot is actually at $(-3,0)$. Let the robot execute **Right** once and then use its sensor to measure its location. The sensor returns $(-2,-1)$. What is the robot's new belief state?

2. Assume again that the initial belief state of the robot is $\{(-3,0)\}$. The robot's goal is now to reach and stop at any grid node (i,j), such that $i \in \{0, 1, 2, 3\}$ and $-5 \leq j \leq +5$ (hence, the goal nodes form a rectangle in the grid, shown in gray in Figure 1). To do this, the robot plans to repeatedly execute **Right** until the data (I,J) returned by the position sensor (after each execution of **Right**) satisfies a certain *termination condition* TC(I,J); then, the robot stops. Note that the robot gets new values of I and J after each execution of **Right** by reading its position sensor. For the robot to succeed, it must stop in the goal. So, TC(I,J) should not stop the robot *before* it has achieved the goal, nor should it let the robot *traverse* the goal without stopping it while it is still in the goal. If the robot traverses the goal and stops after leaving the goal, it has failed to achieve the goal.

   What should TC(I,J) be in order to guarantee that the robot will eventually stop in the goal? [Your answer should be a formal condition on I and J, for example a mathematical expression like $(I > 1) \wedge (|J| < 5)$, where $\wedge$ stands for the Boolean operator "and". The answer is actually a simpler condition than this one.]

   Give a brief proof that your condition is guaranteed to make the robot stop in the goal. [Hint: To prove that the robot will stop in the goal, you must prove that: (1) if TC(I,J) was replaced by the condition *False* (that is , the condition that is always false), then the robot would eventually reach the goal (but would also leave the goal without stopping into it); (2) that TC(I,J) can't make the robot stop before reaching the goal; and (3) that TC(I,J) can't let the robot leave the goal without stopping it before this happens.]

3. Let now the initial belief state of the robot be$\{(-5,0)\}$. Is there a condition TC(I,J) that is guaranteed to make the robot stop in the goal? If yes, give this condition. If not, give a possible sequence of states and sensing data that does not allow the robot to safely stop in the goal, no matter what TC(I,J) it is using.

Problem III: Navigation planning through hotspots

An autonomous wants to go from point *S* to point *F* in a planar environment *P* without obstacle, by using the position information given by its cell phone. However, the robot's telephone company does not provide good coverage. In covered areas, the phone gives the exact position of the robot, but in uncovered areas it gives no information.

More formally, let us model this navigation problem as follows. The robot's position is represented by a point *R* in the horizontal plane *P*. The phone reception areas are represented by *n* discs H_1, ..., H_n in *P* and are called *hotspots* (see Figure 2.*a*). These discs may have different radii and no two discs overlap. Whenever *R* is in a hotspot H_i, the robot exactly knows its current position *R*. Using this information the robot can precisely navigate from any position to any other position within H_i. In addition the robot has a perfect map of the hotspots: it knows the coordinates of their centers and their radii.

However, outside the hotspots, the robot does not control perfectly its motion. It may choose to move in a direction *d*, but the magnitude of its velocity is unknown and only guaranteed to be non-zero. Moreover, its actual motion may make a non-zero angle with the chosen direction *d*. This angle is only guaranteed to be no greater than some given constant angle $\theta < \pi/2$ (see Figure 2.*b*). During the motion the robot's velocity may vary in both magnitude and orientation. Therefore, the robot is only guaranteed to move in an uncertainty cone of half angle $\theta < \pi/2$ centered along *d* and to increase its distance from the apex of the cone over time.

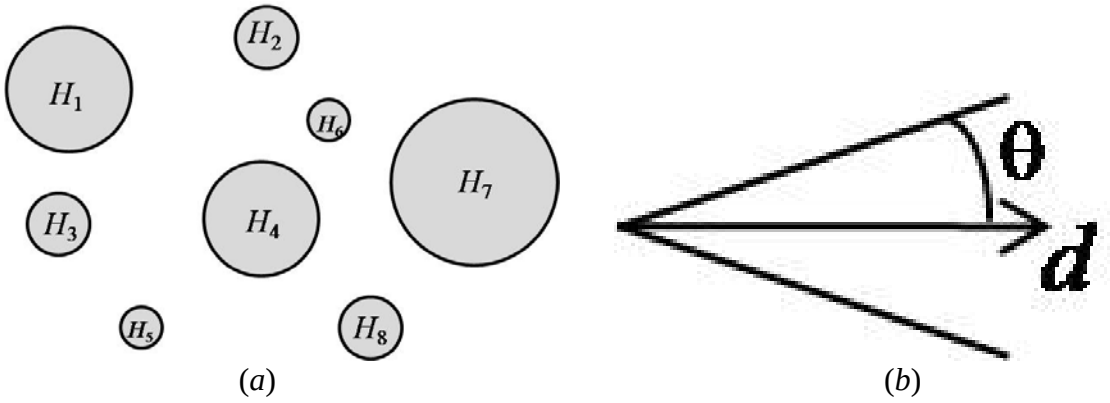


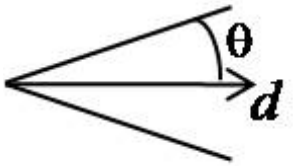
Figure 2: (*a*) A plane *P* with 8 hotspots. (*b*) Cone of possible velocities outside the hotspots when the chosen direction of motion is *d*.

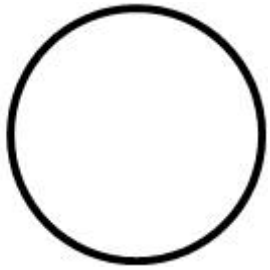
The robot's start position *S* and its desired final position *F* are both located in hotspots, but in different ones. So, to navigate from *S* to *F*, the robot must hop from hotspots to hotspots. Since it can navigate accurately within a hotspot, the robot may select the point in the circular boundary of the hotspot where it will leave the hotspot. From this leave point the robot must choose a direction of motion *d* and keep moving along this direction (with the uncertainty model defined above) until it reaches a hotspot of its choice. The

only directions *d* that are allowed are North, South, East (like in Figure 2.*b*), and West. The robot cannot switch to another direction *d* between two hotspots; it may pick a new direction *d* only when it leaves a hotspot.

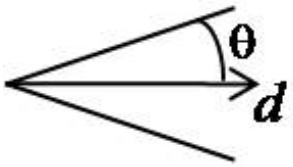
1. We would like to build a backward planner to plan a motion from *S* to *F*. (A backward planner is a planner that plans motion commands from *F* to *S*.) To do this, we define the **preimage** *Pr*(*G*,*d*) of a set *G* of hotspots for $d \in$ {North, East, South, and West} to be the region in the plane *P* such that a motion along *d* from any point in this region is guaranteed, under the uncertainty model defined above, to eventually reach a hotspot in *G* (anyone).

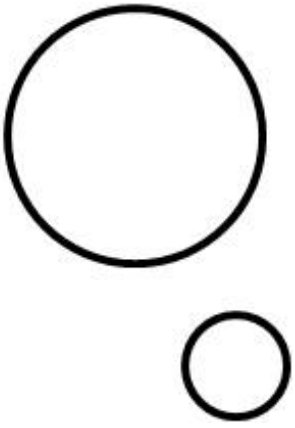
   a. Let *G* consist of the single hotspot shown below. Draw the region *Pr*(*G*,*d*) for *d* = East in the figure below. The angle θ is shown in the figure.



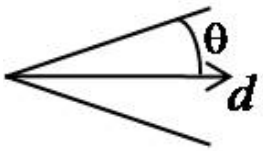


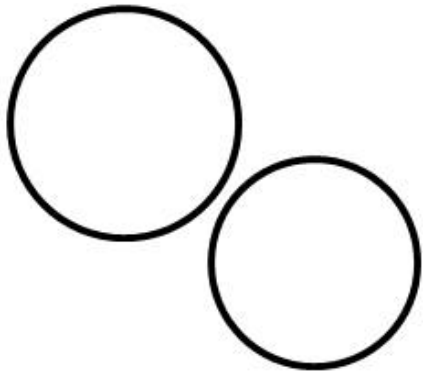
   b. Let *G* consist of the two hotspots shown below. Draw the region *Pr*(*G*,*d*) for *d* = East in the figure below. The angle θ is shown in the figure.





c. Let G consist of the two hotspots shown below. Draw the region $Pr(G,d)$ for d = East in the figure below. The angle θ is shown in the figure. [Hint: The answer is slightly more involved than for question b above.]





2. Assume from now on that an algorithm is available to compute $Pr(G,d)$ for any set of hotspots G and any for $d \in$ {North, East, South, and West}.

   a. Formulate backward planning as a search problem:
   - What would be the nodes of the search tree represent (e.g., hotspots, sets of hotspots, something else)?
   - What would be the start node (root) of the search tree?
   - What would be the finish nodes of this tree?
   - What would be the successors of a node? [Give a precise description in English.]
   To answer these questions, you should first form a good idea of how backward planning will work on this problem.

   b. Assume that the search succeeds. Explain in English how the robot should execute a plan computed by the backward planner.