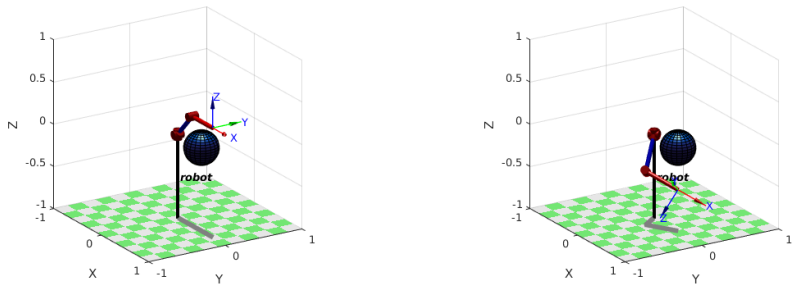
CS4610/CS5335: Homework 2

Out: 2/20/2019, Due: 3/1/2019

Please turn in this homework to Rob Platt via email on the due date. HW PA Q1–5 should be submitted in the form of a set of five files named Q1.m, Q2.m, etc. All this should be zipped up into a single file and emailed to the TAs.

Have a look at the accompanying zip file. Stub files for the questions are provided to you. You should implement each of these. Once implemented, you should be able to run "hw2(X)" in order to run code for question "X". hw2.m is given to you and should not need to be modified. The only thing you need to do is to insert code into the stub functions in Q1.m, Q2.m, etc.



(a) (b)

Figure 1: Illustration of Q2. (a) initial arm configuration. (b) end effector in desired position.

PA Q1: Write a function that draws n random samples in the unit (three dimensional) cube and returns the $n \times n$ all-pairs distance matrix between pairs of samples. Note that this distance matrix should be symmetric and the diagonal should be zero.

PA Q2: The distance matrix calculated in Q1 can be viewed as a fully connected weighted graph. For a given radius and given number of samples, prune edges with weights greater than radius and randomly subsample vertices down to the specified number. Return the corresponding adjacency matrix.

PA Q3: For a given radius and number of samples, calculate whether the r-disk graph on the unit cube *should* be connected or not according to Theorem 7 of the Karaman and Frazolli paper. Compare this result with the actual connectivity statistics you got in Q2. How are they the same or different? Why do you think that is?

PA Q4: Write a function that calculates the sPRM graph for a given set of samples and a given radius. You should prune any input samples that are in collision and output the remaining samples as samplesFree. You should output the weighted adjacency matrix for the sPRM graph. Note that we've

given you two functions to help with collision checking: robotCollision and checkEdge.

PA Q5: Write a function that implements RRT to find a path from qStart to qGoal. Return qMilestones, an $m \times 4$ matrix of vertices (i.e. points) along the path.