Lab 13 Write Up

2

* Assuming that a path exists, it is possible to find a path in any environment given that the reactionary behavior can be used to get the robot to eventually move in any direction or into any orientation.

3

* The output of my A\* algorithm gives a path which consists of a list of {x, y} coordinates existing within the map. First you would use the particle filter to localize your robot within the environment by moving across the environment until the iCreate2 as an accurate estimate of its location. This would work by resampling more frequently across relatively short distances. Next run A \* from the robot’s current location to find the shortest path to the goal. Use odometry in conjunction with the PID controller to move toward each coordinate, utilizing the sonar to search for any obstacles that may appear in the path. If an obstacle is found, the robot would then use A\* to find a path around the obstacle to the goal.
* A\* would have to take into account the DOF of the robot; a standard car cannot rotate in place, nor can it move sideways. This means that path would have to be allow for the restricted motion.
* Given a start and a goal, you would restrict the space to a certain dimension from each side of the start and goal. The environment would be shaped into a square, or rectangle for easier computation.

A large robot requires a wider path to traverse, which could become problematic for in certain environments.

4.1

* The robot moves from the center to the left taking a different path each time. There is a 50% chance that it will go either left or right. The subsequent path it takes for each direction varies slightly with each run. The robot does not seem to have a fixed travel orientation (there is not a designated front and back); instead, it seems to be holonomic, travelling in any direction at any orientation. The seems to rotate to a different direction with each turn, and to a specific orientation while it is moving forward in order to correctly match the final orientation.
* Robot Types
  + Rigid Body Planning (2D)
    - PRM – The robot moves with the same Degrees of freedom and is only constrained to the ground.

The Rigid Body (2D) seems to offer the smallest number of constraints for the planner as it can move in any direction and orientation.

* + Dynamic Car moves with 3 DOF similar to a real car
  + Kinetic Car moves similar to the dynamic, however, physics of motion effect the robot. This means that the car cannot abruptly stop and change directions
  + The Kinetic car needs enough space to slow to a stop to fully switch directions, however, the Dynamic car was unable to find a path to the goal. Both robots constrained the available planners to RRT.
* PRM – two different phases for the algorithm: randomly generating sample points between the start and the goal within the free space, and finding the shortest path between the points. Each point is connected to its neighbor(s) if the neighbors are within a set distance < k creating a roadmap graph. Connections are added until the roadmap is dense enough\*.

RRT – This is a single query planner; random samples are drawn from free space in an outwardly growing manner toward the unknown free space. Once the goal has been added to the tree, a path can be constructed by connecting the points.

* The cost threshold determines which samples are selected within the free space.
* The red dots are the random samples that were taken uniformly within the expanding search of the free space. These are possible points that the road map could have included.

4.2

* The StateSampler for the iCreate2 would use sonar and odometry to sample the free space within its environment from the start to the goal. The odometry would determine whether or not the iCreate2 has reached the goal coordinates, while the sonar would detect if there were any unknown obstacles within the state space (the flat surface where the iCreate2 could traverse). The state would be a position within the environment of the create that it could travel to. The stateValidityChecker would check to see if there are any obstacles within the given state using the sonar, or if the state is reachable.
* MotionValidator would confirm that the car could traverse the path with a valid motion (forward, backward, turning with angle theta). For example, it would not allow the car to move to a state that is directly parallel to the current state because the car cannot move sideways across the y-axis

.