CS 7638: Artificial Intelligence for Robotics

Indiana Drones Project

Fall 2021 - Deadline: Monday Dec 6th, Midnight AOE

Project Description

The goal of this project is to give you practice implementing a SLAM module and an obstacle avoidance system to guide a drone through a forest. In this world, the drone aims to navigate around obstacles (trees) while searching for treasure.

Part A (worth 40%) asks you to build a SLAM system that can keep track of where your drone is located after a series of movements (using measurements to landmarks). Complete the SLAM class in the *indiana_drones.py* file. The movements and measurements will have noise.

Your drone will be dropped into an unknown forest by a team seeking to extract treasure in the area. The drone has special sensors that are able to detect the trees in the region. The measurements from these sensors can be used to help the drone navigate. The sensors will provide the bearing and distance to the trees' center (relative to the drone's location). Both of these measurements contain noise. Since this noise value is not provided, you will need to come up with a way to estimate the noise.

Your drone should make a map of the environment, using its starting location as the origin (0,0) by taking advantage of the measurements to the trees to maintain a good estimate of its own position relative to the initial (0,0) drop-off location.

In part A, your drone will receive a set of measurements as it follows a pre-scripted set of movements, and your SLAM module will need to calculate and report your drone's position after each measurement and movement (relative to the arbitrary (0,0) staring point). [You do not guide the drone in part A and do not extract any treasure, this part is only to test your SLAM module.]

Note that your drone will never have access to the map of where the randomly scattered trees are located.

Part B (worth 60%) asks you to navigate your drone around the environment to extract a treasure marked by * while avoiding the trees. Complete the IndianaDronesPlanner class in the *indiana_drones.py* file.

Your commander will provide you with the location of the treasure. You will need to collect the treasure while avoiding the trees in the forest. In this project we assume the drone is a point (even though in the visualization it occupies some area). There is penalty for each tree the drone crashes into (whenever the drone enters within the radius of the tree's center). Remember that the drone moves on a path, so even if the starting and ending points of your movement aren't inside the tree's radius, the path could intersect it, which would result in a penalty.

There is a time penalty for extracting dirt (failing to extract the treasure). This case happens when you attempt to extract in a location that does not contain a treasure of the type you specified (within a maximum extraction distance). Your drone must be within this pre-defined distance (0.25) of the treasure in order to extract it successfully. When you issue your `extract` action you should supply 3 arguments total, including the treasure type (*) and current estimated location (x, y) of the drone as follows: `extract * 1.5 -2.1` [command treasure_type x y].The treasure will only be extracted if it is within the defined radius, if not there will be a time penalty.

Your drone has a maximum turning angle [in radians] and a maximum distance [in meters] that it can move each timestep [both passed using a parameter]. Movement commands that exceed these values will be ignored and cause the drone to not move. You should specify the movement as follows: `move 1 1.57`. [command distance steering] which means the drone will turn counterclock-wise 90 degrees [1.57 radians] first and then move a distance of 1 meter.

Submitting your Assignment

Your submission will consist of the *indiana_drones.py* file (only) which will be uploaded to Gradescope. Do not archive (zip,tar,etc) it. Your code must be valid python code, and you may use external modules such as numpy, scipy, etc.

We encourage you to keep any testing code in a separate file that you do not submit. Your code should also NOT display a GUI or Visualization when we import or call your function under test.

Testing Your Code

We have provided testing suites similar to the one we'll be using for grading the project, which you can use to help ensure your code is working correctly. These testing suites are NOT complete, and you will need to develop other, more complicated, test cases to fully validate your code. We encourage you to share your test cases (only) with other students on Piazza.

You should ensure that your code consistently succeeds on each of the given test cases as well as on a wide range of other test cases of your own design, as we will only run your code once per graded test case. For each test case, your code must complete execution within the proscribed time limit (5 seconds) or it will receive no credit. Note that the grading machine is relatively low powered, so you may want to set your local time limit to 2.5 seconds to ensure that you don't go past the CPU limit. Note that if VERBOSE is on in the *testing_suite_indiana_drones.py* file, printing will take a lot of time and slow down your execution. So please feel free to increase the time limit while debugging with the VERBOSE on, but when you submit your code, it should run within the 5 second time limit on Gradescope.

A visualization file has been provided to aid in debugging. The visualization will plot 6 pieces of data: the real location of drone, the estimated location of drone, the real location of trees, the estimated location of trees, the types of the trees ('A', 'B', ... etc) and the location of treasure present in environment. The real location of the drone will be a drone with 4 rotors. The estimated location of the drone will be a small blue dot. The real location of trees will be represented by green circles of varying radii. The estimated location of a tree will be a small black dot. The type of tree/treasure will be next to the real location. The treasure is represented by a red triangle.

The estimated points to plot need to be returned from `next_move` as a 2nd (optional) value in the form of a dictionary. The keys should be the landmark id and the values be x,y coordinates. The key representing the drone's estimated location will be 'self'. `{'self': (.2, 1.5), '123': (.4,1.9)}`

Usage: `python visualize.py [-h] [--part {A,B}] [--case {1,2,3,4,5}]`

Example to run the visualization: `python visualize.py --part B --case 3`

We are using the Gradescope autograder system which allows you to upload and grade your assignment with a remote / online autograder. We **may also choose to use the last grade you receive via the remote autograder as your final grade** at our discretion. (See the "Online Grading" section of the Syllabus.)

Academic Integrity

You must write the code for this project alone. While you may make limited usage of outside resources, keep in mind that you must cite any such resources you use in your work (for example, you should use comments to denote a snippet of code obtained from StackOverflow, lecture videos, etc).

You must not use anybody else's code for this project in your work. We will use code-similarity detection software to identify suspicious code, and we will refer any potential incidents to the Office of Student Integrity for investigation. Moreover, you must not post your work on a publicly accessible repository; this could also result in an Honor Code violation [if another student turns in your code]. (Consider using the GT provided Github repository or a repo such as Bitbucket that doesn't default to public sharing.)

Frequently Asked Questions (F.A.Q.)

Q: How can I uniquely identify trees in the environment? *A:* Each tree will have a unique id. Although there may be more than one of the same type of tree in the area, each will have a different unique id.

Q: What are the (x,y) return values from process_measurement() and process_movement() in part A relative to? And how are they different? *A:* Both of them return your best guess for the position of the drone relative to its start location (0,0). process_measurement() gives an estimation of the drone's position after a measurement, and process_movement() provides the drone's position estimate after the drone has moved.

Q: Which way is the drone facing when it lands? *A*: Although slightly unrealistic, your drone will always have a bearing of zero degrees when it lands. (You are welcome.)

Things to think about

1. GraphSLAM estimates the X and Y locations of the drone, but the orientation accumulates noise too. Do you need to handle that? If so, think about how to handle that.
2. The noise in movements and measurements are for the distance and angle to a point. What does this mean for the variance of the noise in the X and Y position? Is it constant? If not, how does it scale? Do you need to handle that? If so, think about how to handle that.
3. How can the drone detect a potential crash of its path with a tree? And when it detects a potential crash with a tree, how can the drone avoid the crash? Could it figure out what options it has and choose one way it could go? Or does it need to find an exact path to avoid the crash?