CS 7638 - AI for Robotics - Asteroids Project

Fall 2019 - Due Monday Sep 16th, Midnight AOE

Introduction

This projects asks you to track a collection of asteroids and:

- estimate their future location
- pilot a craft around them to a goal location

The asteroid field

The asteroids travel in a square with corners (-1,-1) and (1,1). Asteroids outside of the bounding box are considered "out of play" for the moment.

Time is delimited in discrete steps (t=0,1,2,. . .).

Each asteroid's path is specified by two parametric equations in time:

$$x(t) := a_x(t - t_{start})^2 + b_x(t - t_{start}) + c_x$$

$$y(t) := a_y(t - t_{start})^2 + b_y(t - t_{start}) + c_y$$

Each asteroid has a unique set of parameters $a_x, b_x, c_x, a_y, b_y, c_y, t_{start}$.

Given this specification, each asteroid's motion can be modeled using x, y, dx, dy, ddx, ddy.

After drifting outside of the bounding box, most asteroids never return, but a few (<5%) reemerge onto the field later.

Student submission

Your work on this project is to implement the class `Pilot` in the `pilot.py` file. The `Pilot` class must implement three methods:

- `observe_asteroids`: called once per time step, informing the pilot of the latest asteroid measurements. Measurements will include Gaussian noise. Only asteroids currently in the field are measured.
- `estimate_asteroid_locs`: predict the locations of asteroids in the time step after the latest observation.

- `next_move`: given the craft's current state (and previously obtained asteroid observations), choose the next move for the craft to execute.

Task 1: estimation

Estimation counts for the majority of the credit in this project.

On each time step, the pilot will be asked to estimate the locations of all asteroids on the next time step.

The pilot's estimates from the prior step will be compared with the asteroids' current locations. Estimates within `min_dist` will be considered matches.

The estimation is successful if 90% of the asteroids currently active match the prior step's estimates.

Task 2: navigation

The navigation task is intended to be tackled after completing the estimation task. Your navigation code will likely make use of your estimation code.

The task initializes a craft below the asteroid field and asks you to pilot it through the field and into a goal area above it.

The craft (implemented as `CraftState` in `craft.py`) has the following properties:

- current position, heading, and velocity (`x`, `y`, `h`, `v`)
- performance characteristics (`max_speed`, `speed_increment`, `angle_increment`)

Each move by the craft is specified by:

- angle change: the craft may turn left, right, or go straight. Turns adjust the craft's heading by `angle_increment`.
- speed change: the craft may accelerate, decelerate, or continue at its current velocity. Speed changes adjust the craft's velocity by `speed_increment`, maxing out at `max_speed`.

Testing your code

Two local test scripts are provided with this project:

`test_all.py` runs your code against all available test cases. This script closely mirrors the auto-grader we will use to grade your work. To run:

```
$ python test_all.py
```

`test_one.py` runs your code against a single test case, with additional display options. This script is intended to assist debugging.

```
$ python test_one.py --help
usage: test_one.py [-h] [--case {1,2,3,4,5,6,7}]
                   [--display {turtle,text,none}]
                   {estimate,navigate}

positional arguments:
  {estimate,navigate}   Which method to test

optional arguments:
  -h, --help            show this help message and exit
  --case {1,2,3,4,5,6,7}
                        test case number
  --display {turtle,text,none}
```

Test cases are provided in the `cases` subdirectory. We will use similar but different cases to grade your code.

These testing suites are NOT complete, and you may 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.

Submitting your assignment

Your submission will consist of the pilot.py file (only) which will be uploaded to Canvas. Do not archive (zip,tar,etc) it. Your code must be valid python version 2.7 code, and you may use external modules such as numpy, scipy, etc that are present in the Udacity Runaway Robot auto-grader. [Try to ensure that your code is backwards compatible with numpy version '1.13.3' and scipy version '0.19.1']

Your python file must execute NO code when imported. 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. If we have to manually edit your code to comment out your own testing harness or visualization you will receive a -20 point penalty.

We also use the bonnie auto-grader system which allows you to upload and grade your assignment with a remote / online auto-grader. See the submit.py file posted as part of the assignment on Canvas for details. You are not required to use this online/auto-grader feature, but if you do, it will give you more assurance that your code will work correctly with our auto-grader when we grade the file you submit on Canvas. We **may also choose to use the last grade you receive via the remote auto-grader 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.)