CS 8803-O01: Artificial Intelligence for Robotics
Project 1: Spring 2017
(Last updated: January 14, 2017)

Project Specifications

Your task is to complete Parts 1–4 (and Part 5 for extra credit, if you are so inclined) in the Runaway Robot section on Udacity. While you may use the Udacity IDE to test your code against some test cases, **you must submit your code to T-Square for grading using the Assignments tab on the sidebar.**

We will run your code against several other test cases to grade your work. There will be seven test cases for Parts 1–2 and eight test cases for Parts 3–4. Each test case is worth one point, so there is a maximum of 30 points for Parts 1–4. There will be five test cases for Part 5, so there are five points of extra credit available as well. We will run your code once per test case; passing the test case is worth 1 point, and failing the test case is worth 0 points.

The parameters for the test cases will satisfy the following constraints:

- `-20.0 <= target_starting_position_x <= 20.0`
- `-20.0 <= target_starting_position_y <= 20.0`
- `0 <= target_starting_heading < 2*math.pi`
- `10 <= abs(target_period) <= 50`
- `1.0 <= target_speed <= 5.0`
- `-20.0 <= hunter_starting_position_x <= 20.0`
- `-20.0 <= hunter_starting_position_y <= 20.0`
- `0 <= hunter_starting_heading < 2*math.pi`
- A positive value for `target_period` means that the robot will move counterclockwise; a negative value means the robot will move clockwise.
- The standard deviation of the Gaussian applied to the target's measurements will be 0 in Part 1, 0.05 times the target's speed in Parts 2–4, and twice the target's speed in Part 5.
- The hunter bot (if present) will have a max speed twice that of the target in Part 3 and 0.99 that of the target in Parts 4–5.

The requirements for passing a test case are:

- Part 1: Identifies the position of the target within a distance of `0.02*target_speed` using at most 10 calls to `estimate_next_pos`
- Part 2: Identifies the position of the target within a distance of `0.02*target_speed` using at most 1000 calls to `estimate_next_pos`
- Parts 3–5: Puts the hunter within a distance of `0.02*target_speed` of the target using at most 1000 calls to `next_move`
- All Parts: The code takes a maximum of 5 seconds per test case; taking longer than 5 seconds will result in the failure of the test case
- All Parts: No exceptions are raised; any exception raised during the execution of a test case will result in the failure of the test case

We will only use the `studentMain.py` file from your submission; any changes you make to `robot.py` or `matrix.py` will not be seen. If you want to modify those classes for your submission, include them in the `studentMain.py` file and don't import `robot` or `matrix` in your script. You're also welcome to use any other libraries that are accepted in the Udacity interface (e.g. `numpy`).

The submission must consist of Python files labeled `studentMain1.py`, `studentMain2.py`, `studentMain3.py`, and `studentMain4.py` (and `studentMain5.py` if you attempt the extra credit). Do not submit an archive. **Files that do not follow this format will receive no credit.** Additionally, you may use any packages that are accepted in the Udacity interface, but **code that does not run in the Udacity interface will receive no credit.** I am remarkably inflexible about these requirements. Failing to follow them imposes an extra burden on the grader to track down the sources of your errors and fix them to be able to grade your code.

Testing Your Code

On T-Square under the Resources tab, I will post a testing suite similar to the one we'll be using for grading the Runaway Robot project. To use the testing suite, put `testing_suite.py` into the same directory as the `studentMain<part-number>.py` files (and `robot.py` and `matrix.py` if you are using them), then run `testing_suite.py`.

In the testing suite, I've provided some premade test cases based on the ones in the Udacity template code and the Udacity autograder. Other, secret test cases will be used to score each part of your project. 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.

Academic Integrity

You must work on 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 might use comments to denote a snippet obtained from StackOverflow).

We are aware there are public repositories for this project available on the Internet. 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 public repository; this could also result in an Honor Code violation. (Consider using the GT Github repository or a repo such as Bitbucket that doesn't default to public sharing.)