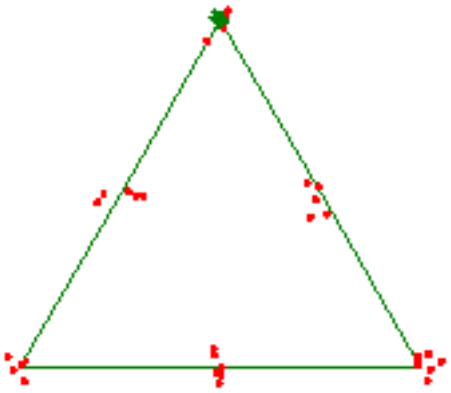
CS 8803 – O01: Artificial Intelligence for Robotics
Project 1 : Polygon Robot
Summer 2018

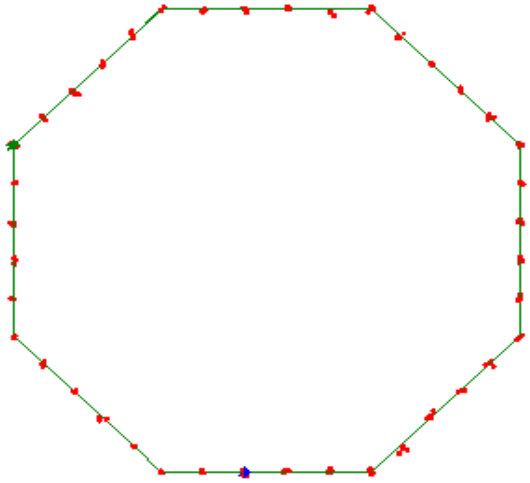
Project Specifications

Hopefully you have successfully completed the Runaway Robot mini-project. Project 1 is basically the same assignment, but with one small change made to the robot's motion model. Instead of always moving in an approximate circle, the robot can now move in the shape of arbitrary convex polygons. This has been implemented by adding one additional robot parameter (*line_length*) which specifies the number of straight forward movements (each of *target_speed*) to make on a single line segment before turning. Note that the robot sends back a location measurement after each move, even if it did not turn as part of that move.

When using the robot class defined in the robot.py file, your code MUST call move_in_polygon instead of move_in_circle. Note that this file has been updated from that used in the mini project!

Of note is that when *line_length* is set to one (1) the Polygon Robot acts like the Runaway Robot. However, if the *line_length* parameter is set to a number larger than 1, (especially if the *target_period* parameter is small) the robot's motion will more closely approximate a polygon than a circle. Below are two examples of robot motion with (period=3, line_length=2) and (period=8, line_length=5). Note that while the robot motion is noise free, the reported locations (measurements) may include Gaussian noise in parts 2, 3, & 4.





You must submit your code to T-Square for grading using the Assignments tab on the t-square sidebar.

We will run your code against 10 test cases per part to grade your work. Each test case is worth 2.5 points, so there is a maximum of 100 points. We will run your code once per test case; passing a test case is worth 2.5 point, and failing a test case is worth 0 points.

The parameter ranges for the test cases ***are slightly different from the Runaway Robot mini-project***, and the will satisfy the following constraints:

- -20.0 <= target_starting_position_x <= 20.0
- -20.0 <=target_starting_position_y <= 20.0
- -math.pi <= target_starting_heading <= math.pi
- 3 <= abs(target_period) <= 12
- 1.0 <= target_speed <= 5.0
- 1 <= line_length <= 16
- -20.0 <= hunter_starting_position_x <= 20.0
- - 20.0 <= hunter_starting_position_y <= 20.0
- -math.pi <= hunter_starting_heading <= math.pi

target_period will be an integer. It represents the number of turns required for the robot to make one complete cycle [e.g. the number of sides of the polygon].
line_length is also an integer, representing the number of segments that make up each "straight" line.
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, and 0.05 times the target's speed in Parts 2–4.
The hunter bot (present in parts 3 & 4 only) will have a max speed twice that of the target in Part 3 and 0.99 that of the target in Part 4.

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 & 4: 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

As you will only upload the studentMain files in 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 the classes in the appropriate studentMainN.py files (and don't import robot or matrix). 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 2.7 files labeled studentMain1.py, studentMain2.py , studentMain3.py, and studentMain4.py only. Do not zip, tar, or archive the files together. Files that do not follow this format will receive a 25% deduction.

Your studentMainN.py files should execute NO code when they are imported. (i.e. comment out any testing code you may have, we only want to test the estimate_next_pos or next_move function.) They should also **NOT** display a GUI or Visualization when we import them 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 25% deduction. Be VERY careful that your submitted code does not contain any imports for modules you may have created for testing or visualization purposes. We recommend testing your final submission in an empty directory, to make sure that it will run correctly without any other files.

Testing Your Code
On T-Square, we have posted a testing suite similar to the one we'll be using for grading the Polygon Robot project. To use the testing suite, put testing_suite_full.py into the same directory as the studentMainN.py files (and robot.py and matrix.py if you are using the default versions), then run

```
python testing_suite_full.py
```

In the testing suite, we've provided 10 premade test cases. We will use 10 other, secret test cases to score 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 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 obtained from StackOverflow).

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.)