CS 8803: Final Project

Overview: For this project, we will be moving out of the realm of simulated robot data and into a real-world situation. You will be provided with roughly 57.5 seconds of video data of a robot moving in a wooden box. Your goal will be to predict the position of the robot as accurately as possible for (approximately) 2.5 seconds after the end of the video. Your prediction will be compared to the actual video data after the end of the video we provide you.

The pedagogical purpose of this exercise is for you to apply the knowledge and techniques you have learned in this class to real-world data. Since real-world data is usually messier and harder to work with than simulated data, this will give you experience with the practical side (as opposed to the theoretical side) of robotics.

Note that in order to make a more accurate prediction, outside research into more sophisticated techniques than those covered in the online lectures may be necessary.

It is not required to use computer vision techniques in order to extract positional data from the video; we will provide you with a Python script that will extract positional data. (Of course, you're welcome to write your own script to extract more data than just the centroid coordinates we give you!)

Specifications: Write a Python program that returns a prediction of coordinates of the centroid (center of mass) of the robot's blue region for the ~2.5 seconds beyond the end of the video data. The prediction should be in the form of a list of 63 pairs of integers. For example:

```
prediction = [[0,0],
              [10,0],
              [10,15],
              ...
              [100,100]]
```

Each pair of integers represents one frame of the video. There are 63 remaining frames in the video that are not shown; therefore you will need to give 63 pairs of integers.

The first element of each pair is the x-coordinate of the centroid, given by the number of pixels from the left side of the video window. The second element of each pair is the y-coordinate of the centroid, given by the number of pixels from the top of the video window. (Note that the video's dimensions are 854 × 480 pixels in each frame.)

Your prediction's accuracy will be judged against the output of the hexbug_tracker.py file you will be provided for the remaining 63 frames of video. We will compute the L^2 error between your prediction and the actual data. The lower your error, the better your accuracy. See below for an example with 4 frames of data:

```
prediction = [[0,0],[10,0],[10,15],[25,25]]
actual = [[0,0],[5,5],[10,10],[20,20]]
```

In this case, the L^2 error would be:

$$\sqrt{dist([0,0],[0,0])^2 + dist([10,0],[5,5])^2 \; + \; dist([10,15],[10,10])^2 + dist([25,25],[20,20])^2} =$$

$$\sqrt{((0-0)^2+(0-0)^2)+((10-5)^2+(0-5)^2)+((10-10)^2+(15-10)^2)+((25-20)^2+(25-20)^2)} =$$

$$\sqrt{0+0+25+25+0+25+25+25} = \sqrt{125} \approx 11.18$$

Grading: The project will be scored out of 50 points. The breakdown for grading is as follows:

- 0-4 points: A Python program is submitted that generates a prediction of the required form.
    - This criterion does not look for correctness of your prediction. Rather, it checks that the program you submit fulfills the specifications given.
- 0-10 points: The program is well-documented. Make sure to give thorough and concise comments describing your program's functionality.
- 0-24 points: The program's functionality matches its documentation, and the functionality is at a reasonable level of sophistication for a master's level computer science course. The following criteria will be used as a guideline for assigning points:
    - 0 points: The program is not submitted.
    - 6 points: The program incorrectly implements the major parts of the functionality described in its documentation, or the program's functionality is overly simplistic.
    - 12 points: The program correctly implements some of the major parts of the functionality described in its documentation.
    - 18 points: The program correctly implements all the major parts of the functionality described in its documentation.
    - 24 points: The program correctly implements all the major parts of the functionality described in its documentation, and the functionality implemented is particularly novel or sophisticated.
- 0-12 points: The program generates an accurate prediction.
    - Accuracy will be judged by the method described above.
    - The team with the most accurate prediction will receive 12 points.
    - The team with the second most accurate prediction will receive 11 points.

    - The team with the third most accurate prediction will receive 10 points.
    - All other teams in the top 25% will receive 9 points.
    - Teams in the second 25% will receive 8 points.
    - Teams in the bottom half will receive 6 points.
    - Teams that do not submit a project will receive 0 points.

Submission: One person per team should submit their team's project. That team member should put the requisite files into their Drop Box on T-Square (using an archive file may be necessary). Include the following in your project:

- A list of all of your team members
- All files necessary for the execution of your program
- All source code (which should be in Python)
- A readme file explaining how to execute your program

Your program should use either the video file testing_video.mp4 or the positional data file testing_video-centroid_data from T-Square as an input. More specifically, the program should be reasonably able (perhaps with minor changes to filenames/strings) to take another video/positional data file as input and return an output relevant to the new input. (If this requirement is not satisfied, there will be a deduction from the "functionality" portion of the grading scheme.)

The program may use other video files/data sets during its execution (e.g. for training purposes), but try not to let it take longer than 3 minutes total in its execution. (We won't be running it on the Udacity IDE, but we still want to be able to grade it reasonably quickly! :)