CS 8803: AI for Robotics

Final Project - Summer 2015

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 10 (ten) 60-second video clips of a robot moving in a wooden box. Your goal will be to predict the position of the robot as accurately as possible for the following 2 seconds after the end of each video. Your predictions will be compared to the actual video data after the end of the video we provide you. In other words, the full videos are actually 62 seconds long, we will give you only the first 60 seconds of each, and compare your predictions to the last 2 seconds which you will not be given.

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, 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 video lectures may be necessary.

Although the project involves predictions based on video data, it will **not** be necessary to use Computer Vision techniques. In addition to each of the 10 test videos, we will provide you with a text file that contains extracted centroid coordinates for that video that you should use as input to your algorithm. If you choose to do additional video processing, you will be able to submit your own version of processed input data for grading.

You will also be provided a ~20 minute “training” video, and corresponding centroid data, which you can use to make your program more accurate. This training data will not be used to score your project’s accuracy. Note: “training” and “testing” videos will be shot using the same conditions, e.g. the same robot moving in the same box, shot with the same camera from the same angle and distance.

In order to have a standard deployment platform you will be provided with a Virtual Machine image. This image will include most of the libraries you are likely to need as well as all the files you need to get started with the project.

Specifications: Write a Python program that reads a list of data points representing the coordinates of the centroid (center of mass) of the robot in each frame of the video file. Your program should return a prediction of the coordinates of the centroid corresponding to the robot's position for the next 2 seconds (or 60 frames) after the end of input.

The prediction should be in the form of 60 pairs of integers, each pair separated by a newline, with the two elements of the pair separated by a comma. The prediction should be output in a text file called `prediction.txt`.

Sample output:

```
641,345
635,337
...
235,39
```

Each pair of integers represents one frame of the video. There will be 60 remaining frames in each "testing" video that are not shown; therefore you will need to give 60 pairs of integers as output. 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.

The main executable of your program should be named `finalproject.py` and invoked as follows:

```
$ python finalproject.py input_file
```

Where `input_file` is the name of one of the 10 provided test files containing a list of centroid positions. The result will be a list of 60 coordinate pairs in the same format output to a file called `prediction.txt`.

The program should take **no longer than 1 minute** to output all ten predictions, i.e. 5-6 seconds per input file.

Your code will be tested on the provided Virtual Machine, so you must make sure it executes correctly in that environment.

In addition to writing the code you will need to include a Project Report that describes your approach. See **Grading** section below for details on what information to include in your Project Report.

Submission: One person per team should submit their team's project. That team member should put a zip archive containing the requisite files into their Drop Box on T-Square:

Include the following in your submission:

- File called `members.txt` which contains the list of all your group members and their @gatech.edu email addresses. One name per line.
- Source code, including `finalproject.py` as well as any additional source files, all of which should be in Python. Make sure `finalproject.py` is at the top level, but feel free to place additional files in subdirectories if you prefer.
- If your code requires additional libraries not available on the provided Virtual Machine image put instructions on how to install them into a file called `readme.txt`
- Include your Project Report in PDF format in a file called `report.pdf`
- Folder called `inputs/` which contains the 10 provided inputs, or your own version of the inputs if you choose to process the video file yourself and generate different centroid data from those provided.
- All the above files should be placed into a folder called `finalproject`, which is in turn zipped up into a single archive called `finalproject.zip`

An example submission would be structured as follows:

```
finalproject.zip
  |------finalproject/
           |------inputs/
                    |------input1.txt
                    |------input2.txt
                    |------...
           |------finalproject.py
           |------members.txt
           |------readme.txt
           |------report.pdf
```

You can submit any number of times before the deadline. Only your final submission will count.

Grading: The project will be scored out of 40 points.

The breakdown for grading is as follows:

- 0-5 points: Meeting specifications.
    - Full credit will be given if your submission meets all requirements in this document.
    - Points in this category will be deducted for:
        - Missing information about installing prerequisites in the required `readme.txt` file
        - Not outputting a list of 60 pairs of integers (outputting floating-point numbers will result in a deduction)
        - Program that takes longer than 1 minute to execute all 10 test cases
    - In general you will lose one point for each manual fix that graders have to do to your submission, or its output, in order to run and evaluate it.

- 0-15 points: Project Report. The following criteria will be used as a guideline for assigning points:
    - There should be a thorough overview of how your algorithm works in your project report PDF. Your report should demonstrate that the implementation of your algorithm is at a reasonable level of sophistication for a Master's level Computer Science course.
    - The report should contain a justification for why you chose the algorithm. For example, if you tried multiple approaches and found one to perform the best in testing you should give a description of all the algorithms you tried and compare and contrast the results of the tests. A picture is worth 1000 words so feel free to use visual aids liberally.
    - There should be concise but thorough comments describing your program's functionality in your source code. If we find your Project Report lacking in complexity we will look in the code. If your code does not have comments explaining what it is doing, and why, you will lose points.
    - For the graders' sanity, your report should be *no longer than* 15 pages long. There is no minimum length, but make sure to fully document your approach.

- 0-10 points: Functionality. The following criteria will be used as a guideline for assigning points:
    - The program should *correctly* implement at least one filtering, tracking, or path-planning technique taught in the course *or* beyond the scope of the course. Note that errors in implementation will result in deductions. Also note that you are free to use external libraries (within reason), as long as you give proper attribution.

    - The technique you implement should be correctly applied to the problem. Simply implementing an algorithm but not applying it to the problem in a sensible way will result in deductions. Make sure you justify this in your report.
    - The program's functionality should match the description given in the Project Report.

- 0-10 points: Accuracy. We will judge your project's accuracy using the method described below. (Note that up to 3 extra credit points are possible here.)
    - The team with the most accurate prediction will receive 13 points.
    - The team with the second most accurate prediction will receive 12 points.
    - The team with the third most accurate prediction will receive 11 points.
    - All other teams in the top 25% will receive 10 points.
    - Teams in the second 25% will receive 9 points.
    - Teams in the third 25% will receive 8 points.
    - Teams in the bottom 25% will receive 7 points.
    - Teams that do not submit a project will receive 0 points.

Your program's accuracy will be judged against the centroid positions in the remaining 60 frames of each of the ten test videos. 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:

$$\sqrt{dist([0,0],[0,0])^2 + dist([10,0],[5,5])^2 + dist([10,15],[10,10])^2 + dist([25,25],[20,20])^2} =$$
$$\sqrt{((0-0)^2+(0-0)^2)+((10-5)^2+(0-5)^2)+((10-10)^2+(15-10)^2)+((25-20)^2+(25-20)^2)} =$$
$$\sqrt{0+0+25+25+0+25+25+25} = \sqrt{125} \approx 11.18$$

The average of your project's L^2 errors over all test cases, after dropping the worst and best scores, will be used to judge your submission's accuracy.