**Final AI Robot Project Notes**

Github repo:

    WarpEngineer/CS8803Final2014 created by WarpEngineer  
    <https://github.com/WarpEngineer/CS8803Final2014>

**Team meeting Notes**

**Meeting 2:**

Jesse found examples of multivariable kalman filters – cannon ball trajectory

put in special cases

seems to be veering to the right

Divide up into separate sections.

**Subproblem:**

1. Find the walls – by taking max and min of the data points

2. figure out direction and speed based on a few points

3. figure out the drift (which seems to be to the right) – (we have to figure out the noise and drift for the Kalman filter?)

uncertainty perhaps because we don’t which part of our robot is parallel to a wall. we don’t know how close our front is to the centroid coordinate.

whenever you hit the walls you can reset the parameters.

when predicting hitting a wall, simulate the hexbug hitting the wall

when we hit a wall we just have to change the matrices so that it’s pointing down, and then filter that way.

best way is to assume that the hexbug gets stuck against the corner.

for direction and speed just do subtraction from a few data points

flips over at 2:38

Q: how big is the hexbug? (don’t worry about it for now)

AG – can start looking at the Kalman filter

Jesse – will try to write out pseudo code for these subproblems

We can have a few people meet up and do some group programming

(Aside: Course High Performance comp. architecture was too hard and had to drop it … had 6 q sample midterm and 25 q actual midterm

Intro to Health Informatics -

)

Meet next Wednesday 6:40pm

**MEETING 1 NOTES**

**Team meeting times:**

Available after 6pm weeknights

Sundays 11-2pm

The last 120 frames of the video might be enough to predict the next 60

**Video is 24 fps, therefore we only need to predict 2.5 seconds**

Kalman filter might be best. – copy from PSet 2

**PSet 2 – kalman filter was with prediction**

**Jessie is comfortable with python**

**A.G- good with programming**

The issue is getting the algorithim right.

Pycharm is the best IDE

Take a few days to look over course material to see if a Kalman filter works.

Do some research to come up with a game plan.

Look at the video to see how we’ll do things.

**Questions:**

**Notes:**

There's a small teaser of the project on Office Hours 2 :)

**Instructions Key Points**

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.

**My Ideas:**

Seeing how we only need to predict 2.5 seconds (60fps/24 frames = 2.5 seconds),

should we set up an algorithm that waits for the bug to be running alongside one of the long horizontal walls and then predicts the more or less straight line motion?

Or at least when it’s starting to heading in a straight line across the box?

The problem is the overview reads:

“Using that training data, you will construct an algorithm that then takes as

input a “testing” video of a shorter length and outputs a prediction of the position of the

robot as accurately as possible for a period of time **after the end of the testing video**.”

So we’re forced to predict based on whatever comes at the end of the testing video.

That’s what I wrote at the end, the only way I could see is that if we do see a straight line starting at the end of the video, to implement that prediction, otherwise we would have to implement another type of prediction

**Important Posts relevant to our project:**

you can definitely use external libraries for math. Just let us know in the documentation what we should have installed before we run your project. – Chris

The mp4 provided is much longer than the testing data that we'll be inputting: each set of testing data will be about 1 minute long, so you can limit yourself to a minute's worth of video if you like. However, you're certainly welcome to find a more efficient method of determining the centroid from the video. – Chris

The testing videos should be roughly 30 seconds to 1 minute long  - Chris

If you want to save the output to a text file, that's fine.

Q: “The instructions say input is either mp4 or data file. Does our solution need to support both input formats or can we pick one and indicate that in the documentation?”

[**Chris Pryby**](https://piazza.com/class/hz1obk155sg3r0?cid=165) [14 hours ago](https://piazza.com/class/hz1obk155sg3r0?cid=165)

You can pick whichever one you want, and then just let us know in the documentation which to use.

 Yes, assume that there might be some missing data points in the testing data if you decide to use the centroid file as input instead of the video file. If you use the video file as input, then you can design your algorithm to track it with better reliability than my (admittedly hacky) tracking algorithm.

The data for grading purposes will not have any bad data points in it.

Your program won't be able to "see" how correct its predictions are during executions. All 60 predictions need to be made after seeing the testing data.

It (the hexbug’s motion) certainly seems random, but it's definitely possible to extract information about the robot. One of the best solutions last year used a particle filter; other good showings used EKFs. Some other more complex techniques such as neural networks and simulated annealing were also used.

**Student post on project approach ideas:**

**https://piazza.com/class/hz1obk155sg3r0?cid=161**

**“**If I were going to approach this final project now what I would probably do is:

* Determine the box boundaries
* Estimate the robot's typical velocity based on the movement history
* Estimate how the incoming angle of impact on a wall affects the outgoing angle and the robot's speed based on the movement history
* Use this information to predict the next 60 frames using math/trig”

“The video is over ten minutes long, meaning you can use many intervals to train and test, refine and refine. “

Final Project L2 Error

 In the summer semester, there was only one set of testing data (instead of several) with 63 data points instead of 60. The lowest L^2 error attained was a phenomenal 87.73, and the next best errors were 254.1, 257.5, and 276.2. The mean error over all submissions was 1164, and the median error was 1043.

Some other interesting factoids about the testing data from the summer:

 - I ran 10000 trials of a script that generated 63 random pairs of integers (uniformly distributed, with x value in [0,863] and y value in [0,479]) and computed the L^2 error of each trial. The largest error I obtained was 3275, the smallest was 2165, and the average error was 2702.

 - The lowest error I got by just predicting the hexbug remains in one spot for all 63 frames was 1256.

So if you can get an error less than 2100 or so, you're beating random. If you can get an error less than 1200 or so, you're beating the best possible "output just one point" strategy.

**FINAL PROJECT PDF FILE:**

**CS 8803: AI for Robotics**

**Final Project - Fall 2014**

**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 a “training” video of a robot moving in a

wooden box. Using that training data, you will construct an algorithm that then takes as

input a “testing” video of a shorter length and outputs a prediction of the position of the

robot as accurately as possible for a period of time after the end of the testing video.

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 video 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 reads either a .mp4 video file or a list of data

points representing the coordinates of the centroid (center of mass) of a robot in each

frame of the video file. Your program should return a prediction of the coordinates of the

centroid of the robot’s blue region for 60 frames. The prediction should be in the form of 60

pairs of integers, each pair separated by a newline, elements of a pair separated by a

comma. The prediction should be output in a text file called prediction.txt. An

example output file has been provided in the “Resources” tab on T-Square.

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. (Note that the video’s dimensions are 854 × 480 pixels in each frame.)

In writing the program, you will be provided a “training” video which you can use to make

your program more accurate. This training data will not be used to score your project’s

accuracy. The “training” and “testing” video data will be taken from the same conditions:

the “testing” videos will just be shorter videos.

**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! :)

**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 not meeting specifications such

as:

■ Missing information in the required readme file

■ Not outputting a list of 60 pairs of integers

■ ...

● 0-5 points: Documentation. The following criteria will be used as a guideline for

assigning points:

○ There should be a brief overview of how your algorithm works in your readme

file.

○ There should be a justification given for the algorithm you select in your

readme file.

■ For example, if you try multiple algorithms and find one to perform

the best in testing, give a description of the algorithms you tried and a

brief description of the results of the tests.

○ There should be thorough and concise comments describing your program’s

functionality in your source code.

● 0-20 points: Functionality. The following criteria will be used as a guideline for

assigning points:

○ The program should *correctly* implement at least one filtering, localization,

or path-planning technique taught in the course *or* beyond the scope of the

course. (Note that errors in implementation will result in deductions.)

○ The program’s functionality should match the description given in the

readme file and the comments in the source code.

● 0-10 points: Accuracy. Accuracy will be judged by 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 output of the hexbug\_tracker.py file you

will be provided for the remaining 60 frames of video. We will compute the L2 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 L2 error would be:

√*dist*([0, 0], [0, 0])2 + *dist*([10, 0], [5, 5])2 + *dist*([10, 15], [10, 10])2 + *dist*([25, 25], [20, 20])2 =

√((0 − 0)2 + (0 − 0)2) + ((10 − 5)2 + (0 − 5)2) + ((10 − 10)2 + (15 − 10)2) + ((25 − 20)2 + (25 − 20)2) =

√0 + 0 + 25 + 25 + 0 + 25 + 25 + 25 = √125 ≈ 11.18

The average of your project’s L2 errors over all test cases will be used to judge your

submission’s accuracy.