6.835 Project 1: Early Sketch Understanding – Stroke Segmentation

Due: 5:00PM Friday, February 23^{rd}, 2018

This exercise is intended to give you some hands-on experience in dealing with pen-based computing. We will supply real data (pen strokes) and some basic Python code as a foundation. The exercise then involves implementing the corner finding approach in the attached paper by Stahovich [1].

As the paper is at times unclear on the approach it actually uses, we have outlined it for you. Nevertheless, you should read the paper carefully, and may discover that it is useful to refer to it as you work on this project, in order to understand some of the details.

You will no doubt discover that it is not always easy to figure out what the system described in the paper actually did; sadly this is typical of papers in the research literature. As a result it's useful to learn how to dig out the details: If you want to do research, you need to be able to at least reproduce other people's results. Even if you simply want to use an idea contained in a paper, you have to be able to decipher it in detail. Sometimes this is more work than it should be, but knowing how to do this is an important skill.

Things to note about this project:

- It will likely take you longer than you expect, so start soon. Don't put it off, or you will not get it done in time. Please take advantage of Piazza to get your questions answered. You may also email Billy (wcaruso@mit.edu) with any questions.
- We expect you to implement what's outlined here, not to reproduce everything described in the paper. If you get inspired you are of course welcome to go beyond what we outline, but make sure you do at least what is listed below.
- This exercise is about getting down to the details of dealing with real pen data. You will confront a collection of issues and details that must be dealt with if you want to make sense of pen input, and get to see how this actually works.

1 Before You Start

This project requires coding with Python 3.6. If you are not familiar with Python, please first refer to numerous tutorials available online and ask for help. The course web page has a listing of tutorials in the Materials section.

2 Getting Started

Download the project file package (`project1.zip`) from the course web page. The file contains some Python support code and the dataset `strokes_data.py`. After extracting the file, open them in your favorite text editor.

Using the command prompt, create a virtual machine to hold the dependencies of the mini project. On a Mac, in Terminal you can type:

```
cd MiniProject1/
virtualenv venv -p python3
source venv/bin/activate
```

Instructions for Windows are slightly different[1]:

```
cd MiniProject1/
mkvirtualenv venv -p python3
```

Now the virtual machine should be up and running (you should see the (venv) at the front of the terminal line). Note: A virtual environment is not necessary, but is a good habit for developers to follow.

Next, install the dependencies using Pip.

```
pip install -r requirements.txt
```

The `Strokes` object contains arrays x,y,and $time$, representing the position and timestamp of each sampled datapoint in the stroke. You can visualize the strokes by running `python eval_all_strokes.py` from your terminal. This script will create Stroke objects, send them to be segmented, and then graph the raw stroke and segments. Right now, stroke segmentation is unimplemented, so you should see all the strokes as they appear in Figure 1.

3 Segmenting Strokes

The main part of the assignment is to identify corners and classify the segments between corners as lines or arcs. There are a number of parameters that may have to be adjusted to improve performance, so be sure to develop code that

[1] See this tutorial for additional info: http://timmyreilly.azurewebsites.net/python-pip-virtualenv-installation-on-windows/

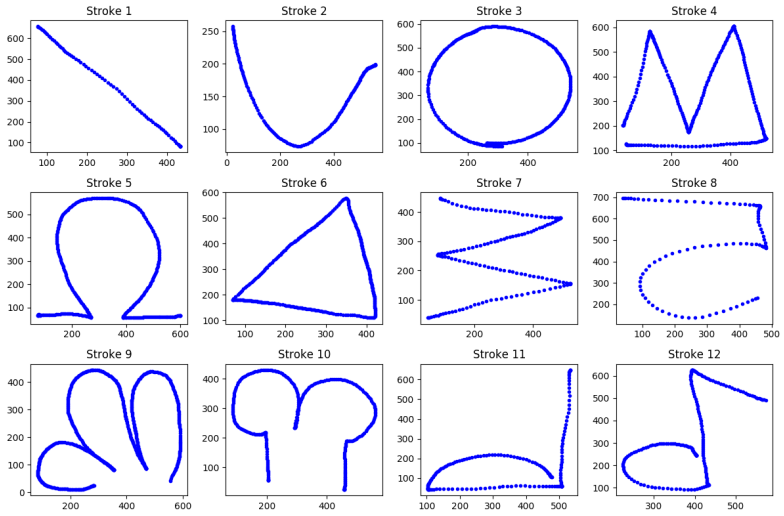

Figure 1: The 12 strokes you will segment

makes this easy. You're free to structure your code however you want, but you must provide this function, found in stroke_segmentation.py:

```
def segment_stroke(stroke):
    ...
    returns segpoints, segtypes
```

- `stroke` is a stroke data structure represented by the Stroke class.
- `segpoints` is a length N array of the indices of points at which the `stroke` is segmented, including the start/endpoints of the stroke.
- `segtypes` is a length N-1 binary array indicating the type of each segment: 0 for a line, and 1 for an arc.

Your stroke segmentation should follow the steps outlined in the paper:

1. Construct an array of the cumulative arc lengths between each pair of consecutive sampled points.
2. Construct an array of smoothed pen speeds at each point. The size of the window over which smoothing occurs is one of the parameters you will need to adjust.
3. Construct an array of tangents. To do this, fit a linear regression line to a window of points surrounding each point, and note the slopes. The size of this window is one of the parameters you will need to adjust.[2]
4. Define curvature, which is the change in orientation (angle) over change in position along the arc length.
   (a) Convert the slopes to angles using the arc tangent function (`atan`).

[2]Numpy provides `np.linalg.lstsq()`, a built-in function for performing least-squares linear regression.

(b) To see a problem that crops up with using arc tangent (and with using angles in general), plot the arctangent of the slopes that you obtained for `strokes[4]` – the "Ω" symbol, and briefly explain what you think is going on. As this phenomenon this will corrupt your estimates of the curvature (change in angle over change in position along the arc length), they will have to be corrected.

(c) Write and test a method called `correct_angle_curve` (in stroke_segmentation.py) to correct the phenomenon you found in (b). You will need to define your own input and output arguments for the function.

Hint: Be careful when assigning curvature to consecutive overlapping points (e.g. points with the same position at consecutive times).

5. Identify corners. The paper is a little unclear on this, but the basic idea is to have two separate criteria, both of which contribute points:

   (a) Using speed alone, select local minima of speed that are lower than a threshold percentage of the average speed, and

   (b) Identify local maxima of curvature, and accept them if the speed at that point is below a threshold (even if not a minimum) and the curvature is above a threshold.

6. Combine the two sets of points identified in step 5, with any nearly coincident points merged. You will need to decide what "nearly coincident" should mean, and decide which of the two points to keep.

7. You should now have identified a set of segmentation points. You will now classify each segment as either a line or a curve. (For this assignment, you do not need to consider elliptical curves, only circular curves).

   (a) For each segment, fit a line using linear regression, and fit a circle using the provided code, `circle_fit.py`.

   (b) Calculate the residual errors for the line and the circle.[3]

   (c) Find the angle subtended by the arc of the circle.

   (d) Based on the residual errors found in (b) and the angle subtended by the arc of the circle found in (c), classify each segment as a line or curve (0 for a line, and 1 for an arc).

8. You can view the output on all given shapes in the dataset using `eval_all_strokes`, or on a single shape using `eval_stroke`, which will both call your `segment_stroke` function and pass your output to `plot_segmentation(stroke, segpoints, segtypes)`. Run these scripts from the command line:
   `python eval_all_strokes.py`
   `python eval_stroke.py <stroke number>`

[3] Although Numpy's `np.linalg.lstsq` function provides the residuals for the line fit, you will need to write your own function to compute the residuals of the circle fit.

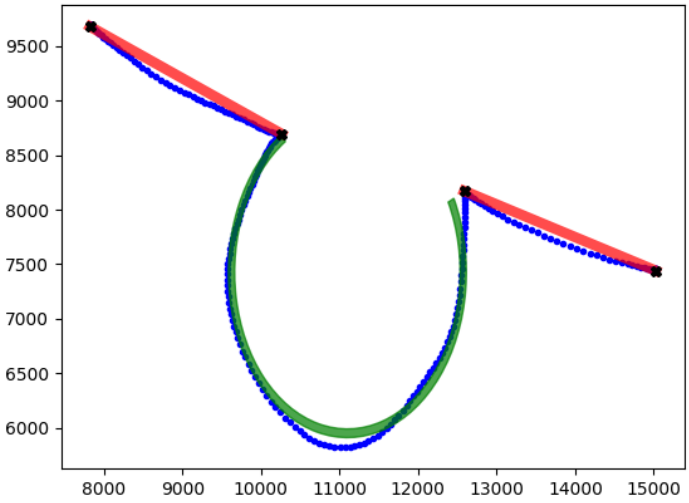

Figure 2: An example system output (not obtained using default parameters)

In this viewer, segments classified as lines are rendered in red, and segments classified as arcs are rendered in green. Corners are indicated with black x's. An example of reasonable system output is shown in Figure 2.

9. In your code, you should have at least seven parameters:

Parameter	**Value used in the paper**
Size of window for smoothing penspeed	5 total (2 points on each side)
Size of window for computing tangent	11
Speed threshold 1	25% of average
Curvature threshold	.75 degree / pixel
Speed threshold 2	80% of average
Minimum allowed distance between two corners	unspecified
Minimum arc angle	36 degrees

Using these values, evaluate your system and print out the results with `eval_all_strokes.py`. Experiment with different values for these parameters, and print out the best results you can obtain, indicating the parameter values that you have chosen.

10. Now, we will take our segmentation one step further and attempt to Merge adjacent segments. Do so according to the following rules:

    (a) If a segment is shorter than 20% of the length of its adjacent segment, or, if adjacent segments are of the same type, try to merge them

    (b) If error of fit of new segment is $< 10\%$ of the sum of fit errors of the original 2 segments, use the new segment

    You are free to change these percentages to make your system optimal.

4 Submission

Please submit the following in a zipped file to the course website in the Homework section:

1. A folder containing your source code. Please submit all the files you created/used in this project.

2. A color image of the results of your system with default parameters.

3. A color image of the results of the best parameter values you could identify, along with the values.

4. A PDF containing the following:

   (a) The answer to the question in step 4, regarding the subtlety of arc tangent, and why this will corrupt the estimates of the curvature.

   (b) Your definition of "nearly coincident", and how you chose the value for the parameter (*i.e.* the minimum allowed distance between two corners).

   (c) Compare the results of your chosen parameter values with those of the default. If yours turn out to be better, explain what you did, and why it works better.

   (d) The paper describes several additional steps to improve this system. Which of these (or what other steps) would be the most effective in improving your system as it now stands?

References

[1] Stahovich, Segmentation of pen strokes using pen speed. In *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural.*