Assignment 2: Gesture Recognizer

Imagine working for a company that would like to use the phone as an input device to an interactive video game system like the XBox or Playstation--for example, using the phone as a paddle in tennis or as a "ball" in bowling. In this assignment, you will build your own 3D gesture recognizer to automatically recognize these gestures.

While in the "real world" you would ultimately need to create a real-time gesture recognizer, for this assignment you will make an *offline* version in Jupyter Notebook.

Specifically, you will make two recognizers:

(i) a *shape-matching* recognizer such as via a Euclidean distance metric or Dynamic Time Warping, and

(ii) a *feature-based* (or *model-based*) recognizer using a support-vector machine (SVM) (recommended), hidden-markov model (HMM), or an alternative supervised learning approach of your choosing.

Within Jupyter Notebook, we will use Python 3 and these amazing libraries numpy, scipy, matplotlib, and scikit-learn. Numpy and scipy provide numeric array handling and signal processing, matplotlib provides visualization, and scikit-learn is the de facto machine learning library in Python. You are welcome to use other libraries as well (e.g., this DTW library).

For your deliverables, you will turn in your Jupyter Notebook, your recorded gestures, and a brief (1-page) report on your algorithmic approaches and performance results.

Table of Contents:

Note: This assignment has been adapted from this assignment created by Jon Froehlich from the University of Washington.

Learning Goals

- Introduce and learn machine learning approaches popular for sensing/gesture recognition, including shape matching and feature-based classification
- Practice basic machine learning pipeline: data collection, signal processing, model training, and model testing using k-fold cross validation
- Introduce and learn popular data analytics tools and toolkits (*e.g.*, Jupyter Notebook, scipy). I hope you'll enjoy learning and using Jupyter Notebook as much as we do!

We expect that you will seek out external sources to help you learn and complete this assignment. An important skill to develop is being able to independently find relevant resources online (Google is your friend!), try out possible solutions, and push through challenges and setbacks. Please properly attribute your sources in your report (and in your code), but feel free to use what you find!

For instance, Jeff found some good resources for what features might be useful for the classification problem by Google'ing "svm gesture recognition accelerometer". You might imagine just searching for "gesture recognition", but that is likely to give you a

bunch of results that aren't relevant to your task. I added "svm" to target results that speak about using SVM (a really popular classification algorithm), which makes it more likely that the results will also talk about the features used. I found that a lot of the results were for computer vision-based recognition of gestures, which isn't what we're doing. I first added "phone" to the keywords, but that was too generic, and didn't improve results much. Adding "accelerometer" to the search terms helped a lot because that's a more specific, somewhat technical term. In combination with the other search terms, it seems to have worked pretty well.

Gestures

Your Python-based gesture recognizer needs to support the following gestures:

1. Backhand Tennis
2. Forehand Tennis
3. Underhand Bowling
4. Baseball Throw
5. At Rest (i.e., no motion)
6. Midair Clockwise 'O'
7. Midair Counter-clockwise 'O'
8. Midair Zorro 'Z'
9. Midair 'S'
10. Shake
11. A gesture of your own creation

For this assignment, you will analyze two gesture sets:

(i) the gesture set provided here, and,

(ii) a gesture set that you create yourself -- of the above gestures--using the Android-based gesture recorder provided for you: GestureLogger. You should record five samples per gesture (just like my example set). By default, the gesture logger stores files in the Downloads folder of external storage.

The GestureLogger is designed to be used on an Android device (phone, tablet, etc). While we know that not everyone will have an Android phone of their own, one important Human-AI skill to develop is scrappiness. We bet that you can somehow find

one! If you're really stuck, we'll have one or two available during office hours -- if you decide to go this route, come to office hours with the pckg ready to install.

Instructions to compile and run the Gesture Logger on your phone

You will need:

1. A Windows, Mac or Linux machine.

2. An Android phone running 6.0 (Marshmallow) or above.

3. A USB cable

Download source code

Download the Gesture Logger source code.

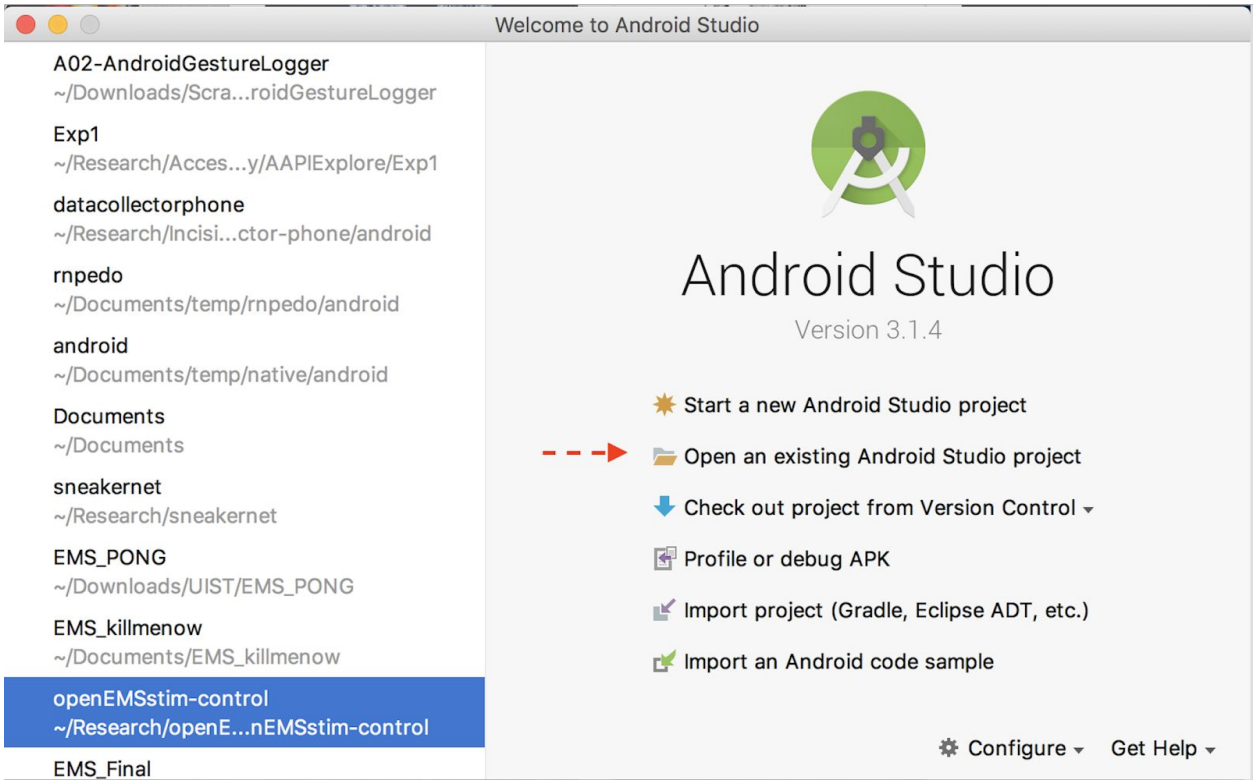
The relevant part of the code is located in `A02-GestureRecognizer/Code/A02-AndroidGestureLogger`.

Download and install Android Studio

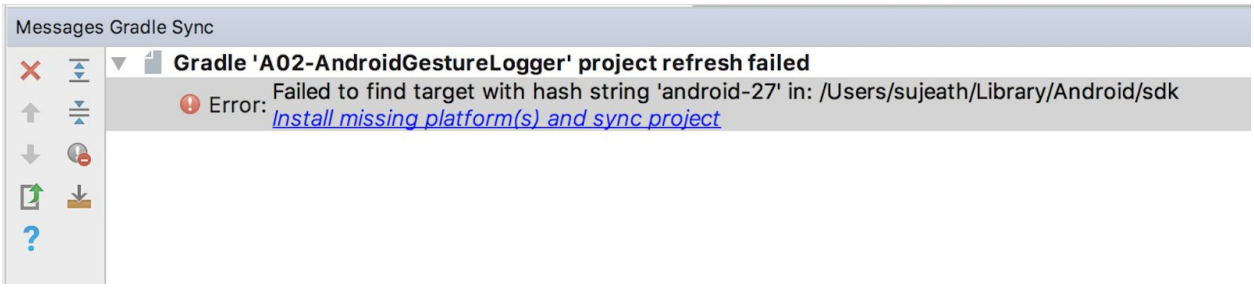
Android Studio is a free development environment from Google for Android apps. While apps can be tested either on an emulator or a physical device, you will need a physical device to collect data from sensors. Install it by following the instructions.

Import the code

1. In the Android Studio startup screen, click on `Open an existing Android Studio project` and select the folder containing the source code. This should be `A02-GestureRecognizer/Code/A02-AndroidGestureLogger`.



2. Android Studio will raise an error containing `'Failed to find target with hash string 'android-27'`. Just click on the helpful `Install missing platforms and missing projects` link and it will fix itself.



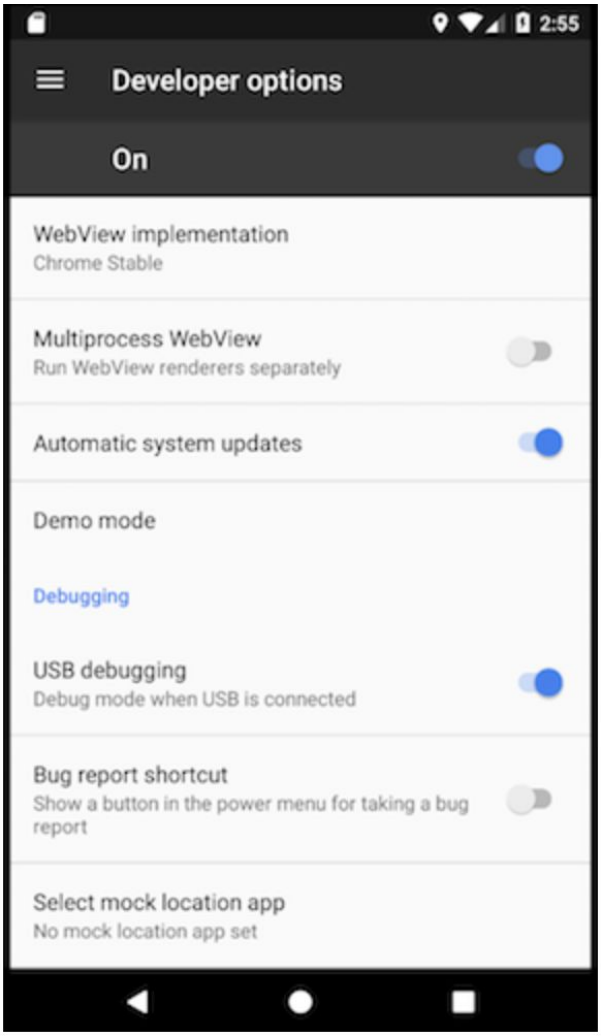
Connect your phone

1. Connect your phone via USB.

2. To let Android Studio talk to your phone over USB, you need to turn on Developer Mode. Follow these instructions.

3. Go to `Settings > Developer options`. If you don't see Developer options in your settings, then something went wrong in the previous step.

4. Turn on *USB debugging*.



Run the app

1. In Android Studio, click on the green run icon or go to `Run > Run 'app'`.

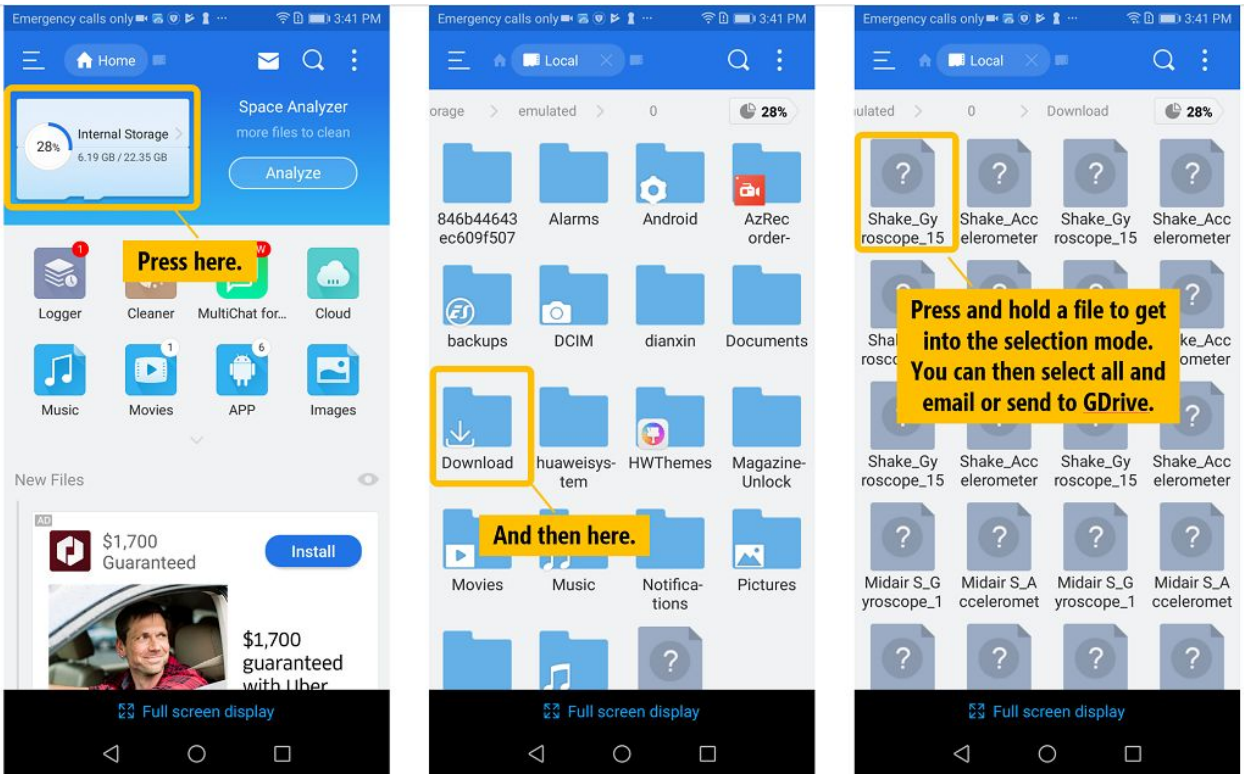
2. It should prompt you to select a device.

3. If all went well in the previous section, your device should be visible in the list. Try reconnecting your device or checking your notifications for any security messages if that doesn't work.

4. Select the device and press next. The system will then compile the app and install it.

You only need to do this step once. For subsequent runs, just tap the icon in your applications list as if it were installed through the Play Store.

To get the files off of the phone's storage, I used ES File Explorer and then shared the selected files with one of my Google Drive accounts (you could also email them but sharing to GDrive was more efficient for me). You can download ES File Explorer from the Google Play Store.



Parts

- [1 pt] Record your own gesture set. See above.
- [3 pts] Visualize both gesture sets (my recordings of the gestures and your recordings of the gestures) in Jupyter Notebook. Your visualizations should include anything that helps you analyze the signals and aid your signal processing and classification approaches. At the very least, you should visualize the raw x, y, and z accelerometer and gyroscope signals as line graphs and visualize some sort of frequency analysis (e.g., spectral density plot, spectrogram). Please appropriately label axes, titles, and include legends.
- [4 pts] Design and implement a shape-matching gesture recognition approach (*e.g.*, using DTW). What transformations of the signal are necessary here (*e.g.*, smoothing, detrending, etc.)?
- [4 pts] Design and implement a model-based gesture recognition approach (*e.g.*, using an SVM or another model of your choosing). What features are most discriminable? How did you encode those features in your model?
- [4 pts] Evaluate your two approaches using k-fold cross validation. For each user (my gesture set and your gesture set), randomly split the data into k-folds (k=5). Use four folds for training and one for testing and repeat this five times (with a different fold reserved for testing each time). You do not need to examine cross-user performance (*e.g.*, training on my gesture set and testing on your gesture set); however, see the Bonus section. For performance metrics, your Notebook should print out the following for both the shape-matching and model-based approaches: (i) overall accuracy; (ii) per-gesture accuracy; (iii) and a confusion matrix.
- [+1 pts] You will receive one bonus point if either your shape-matching or your model-based approach perfectly classify the gesture data (for both gesture sets).

Deliverables

- [2 pts] Your Jupyter Notebook + your gesture set (either a github or gitlab link or a zip uploaded to Canvas). Your Notebook should include all the code you wrote to visualize, process, and classify the gestures along with an evaluation framework + performance results. Your Jupyter Notebook should be clear, well-organized, and sufficiently commented (with additional markdown as

necessary). As always, please acknowledge any websites or other sources you used to inform your solutions and code.
- [2 pts] A brief, ~1-page report with: (i) a description of your shape matching approach and its performance including overall accuracy, per-gesture accuracy, and a confusion matrix; (ii) a description of your model-based classification approach and its performance including overall accuracy, per-gesture accuracy, and a confusion matrix; (iii) an enumeration of key challenges; (iv) and a reflection of what you learned. You can include as many images as you want (e.g., copy/pasted from your Notebook). Images are free. You can write the report in Jupyter Notebook but please make this clear in your submission.

Bonus

- [+1 pt] Create a new gesture set that is more challenging by varying the speed of your gestures (e.g., fast, medium, slow) and other aspects (e.g., different orientations of the phone). Analyze this new data. How well do your algorithms perform? What seems to break?
- [+1 pt] For the model-based approach, run an additional analysis that examines performance as a function of the number of training examples. How well does your classification approach work with a small number of samples?
- [+1 pt] Imagine that instead of trying to classify the gestures, you were trying to determine how many unique gestures there were in a set (for this scenario, pretend that you do not actually know). Design and implement an approach that attempts to determine how many unique gestures there are in the set. What kind of problem is this in machine learning? What approach did you use?
- [+2 pts] Write a gesture recorder/logger for iPhone.
- [+2 pts] The gestures we used in this assignment were pre-segmented. Update your analysis pipeline to work with an incoming stream of sensor data and appropriately segment the start and end of gestures before attempting classification.
- [+4 pts] Convert either your shape-matching or your model-based classification approach to a real-time classifier on Android. Note: only try this if you have finished the main assignment. You will need to demo this in class for the points.
- [+4 pts] Adapt your real-time model to work on a smartwatch. Also must demonstrate this in class to receive the points.