CS74 Milestone Report
SketchyPass: Freeform Graphical Passwords for Android

Will Geoghegan and Phil Royer

February 19, 2013

1 Introduction

Our project is SketchyPass, a freeform graphical password system for use on Android smartphones. This kind of password offers several key advantages over traditional pin- or grid-based password systems. First and foremost, the possible input space is vastly larger than that of a pattern on a 3x3 grid or a string of a few characters. Drawings are also easier for humans to remember, but more difficult for computers to recognize. Recognizing a password drawn on a screen as being similar to one stored in a phone is a natural choice for the application of machine learning techniques, specifically binary classification.

The phases of our project are described briefly as follows:

- Write an Android app to collect and store password sketches to be used as the data set
- Pre-process these images, reducing their dimensionality and transforming them into a usable form
- Learn a distance metric for binary classification of sketches

2 Accomplishments

2.1 Android App

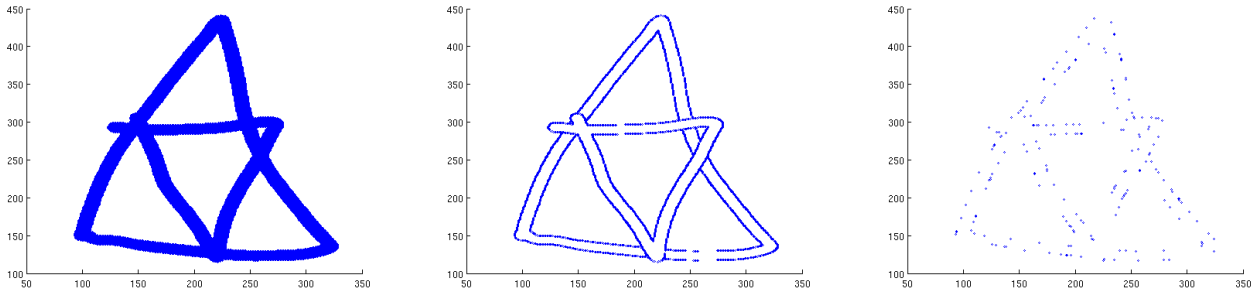
We have created our Android data collection app and used it to collect about 300 sketches from 30 different users, asking each user to draw a simple password sketch ten times in succession. Each sketch was stored as a .png file on the phone. Although this process took longer than we had anticipated, it provided us with data which was very clean and usable.

Figure 1: Our Android data collection app allows a user to draw and save a graphical password.

2.2 Preprocessing

Our sketch processing technique consisted of several steps, the first of which was receiving a .png file from the phone and compressing it into a simple grayscale pixel matrix. We used our Android app to blur the edges of the image so that the edges were the only points for which the grayscale value was neither 0 nor 255. This allowed us to apply a simple edge-detection algorithm to generate an outline of the image's contours.

We then selected 200 random points from the sketch to store, as we believed that 200 points would be enough to give us a uniform distribution over the entire image while still remaining computationally tractable.



(a) A full pixel matrix (b) Edge detection on the image (c) 200 random edge points

Figure 2: Transforming the full pixel matrix of a sketch into a random selection of edge points.

By the end of this process, we had a relatively tractable dataset (636KB) that still maintained the integrity of the password sketches, allowing us to perform shape context transformations on our images.

2.3 Shape Context

We found and modified code from Serge Belongie to perform shape context matching on our drawings.[1][2] This method maps each of the points from one image to a corresponding point in another image by computing the shape context of each point, and minimizing the differences in shape contexts.

Shape context is a feature descriptor that describes a point in an image by finding the relative position of other nearby points from the same image. To compute shape context we drew a number of bins around each point that are uniform in log-polar space, and counted how many other points were in each bin. Then we matched pairs of point from two different images while minimizing the difference in shape context of the matched points. The sum of the minimized differences gave us the shape context cost for two images. Then, according to Belongie's algorithm, we transformed one of the images to conform more closely to the other using both an affine and a thin plate spline transformation. We computed the costs of each of these transformations and added them to the shape context cost to get a total image distance for each pair of images. The results in the graph below show the differences between the mean image distances of sketches from the same user and sketches from different users.

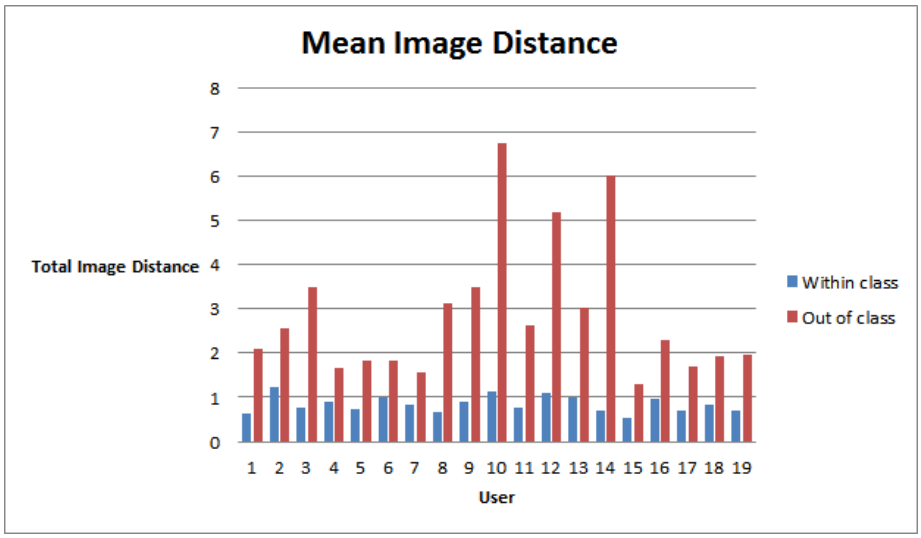


Figure 3: Mean image distance within class and without.

3 Remaining Work

We still need to learn a distance metric and binary classifier for these images. We will explore both Large Margin Component Analysis (LMCA) and the Support Vector Decomposition Machine (SVDM). Both of these methods learn a distance metric and binary classifier simultaneously rather than sequentially, which enhances their accuracy.[3] While LMCA is general to problems containing any number of classes and SVDM is specialized to binary classification problems, we believe the large margin used in LMCA to separate data points belonging to different classes will help avoid false positives in our results, which is our biggest concern.

4 Progress Assessment

The major roadblock to our progress thus far has been the Android app, which took longer than we expected. This delayed us from even starting coding up our preprocessing and learning algorithms until the app was finished and we had collected sufficient data. We had hoped to have our distance metric learning and classification at least rudimentarily completed, which we have not done yet. The preprocessing and shape context matching both went relatively smoothly, however, and we are confident that we can learn distance metrics and classifiers of sufficient strength to make this project successful before the final deadline.

Notes

[1]G. Mori, S. Belongie, and J. Malik, "Efficient Shape Matching Using Shape Contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27:1832-1837, 2005.

[2]S. Belongie, J. Malik, and J. Puzicha, "Matching with Shape Contexts," http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/sc_digits.htm 2001.

[3]L. Torresani and K. Lee, "Large Margin Component Analysis," *Advances in Neural Information Processing Systems (NIPS) 18*, 2006.