CS74 Final Report
SketchyPass: Freeform Graphical Passwords for Android

Will Geoghegan and Phil Royer

March 8, 2013

1 Introduction

SketchyPass is an Android authentication system that allows a user to enter a freeform drawing on the touch screen to provide personal identification. An improvement on Android's Pattern Unlock feature, this method allows passwords to take any shape imagined by the user, which should drive improvements in both security and usability. In order to recognize sketches that are slightly different from one another we explore methods to perform binary classification of image distances between any two sketches. These methods include:

- Dimensionality reduction using edge filtering and stochastic point selection
- Cost measurement of point-to-point matching using shape context
- Affine and thin plate spline transformation cost estimates
- Soft margin support vector machine classification
- A learned linear transform weighting to emphasize the most indicative feature

With a combination of these methods, we were able to effectively classify images as correct or incorrect passwords with error rates under 5%.

2 Prior Work

The intended usage of SketchyPass is a replacement of the methods currently used to wake the phone up from an idle setting. The Android Jelly Bean OS provides four password options: Face Unlock, Pattern, PIN, and full keyboard password. We are most interested in improving on the Pattern Unlock feature, with which users enter a password by connecting two to nine nodes aligned on a 3x3 grid. This popular method follows research in the field suggesting that humans have more powerful memory when targeting abstract visual recollections than the sequences of letters, numbers, and special characters that form traditional passwords. The haptic process of inputing a graphical password could also play a role in improved memorization.[1]

We seek to further improve the usability and security of graphical passwords by relaxing the grid restriction of Pattern Unlock, allowing users to draw completely freeform images on a canvas. Biddle et. al. (2012) found an increase in security when a competitor to Pattern Unlock allowed for a 5x5 grid of nodes, so we contend that a further increase in the password space and freedom of expression by the user will see even better results. We also think that relaxing the grid makes password creation more creative and enjoyable, persuading users to draw more complex passwords and change them more often.s

3 Data Collection

To train and test our methods we built a simple Android application for the collection of sample passwords from volunteers, allowing us to gather a clean data set that was created by users with password sketches in mind. The app consists of a large drawing canvas and two small buttons: one to save the current drawing as a .png image on the phone and one to clear the drawing space. The canvas is identical to our proposed password entry system so that our training and testing data is as realistic as possible.

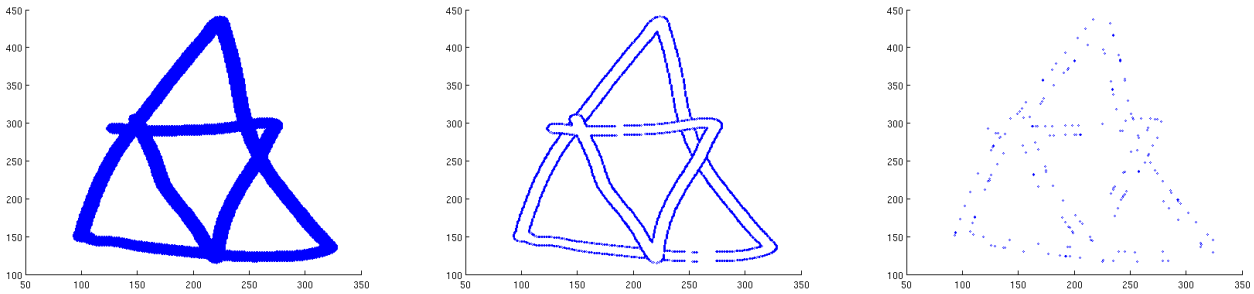
We asked each volunteer to draw a simple password sketch 10 times, collecting from about 30 volunteers (although only about 2/3 of these provided usable data due to difficulties following instructions). We created our data set by, for each volunteer, computing transformation costs for 9 different pairings of the same sketch and 9 randomly selected different sketches. We partitioned our data to create our training and tests sets, randomly selected 70% of the transformation pairings for the training set, and the remaining 30% as the test set.

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

4 Methods

4.1 Dimensionality Reduction

Initially we were faced with a set of high-dimensional images that were very computationally expensive for us to process, so we used an edge detection filter to find the points that were most important to our understanding of the given shapes. Since our drawing app blended the edges of brush strokes automatically, edge detection was simply a matter of keeping each pixel if its color was not equal to the background color or the stroke color. This left us with a dramatically smaller set of points, but each sketch had a different number of edge points, and all of them still had far more than we could compute on in a reasonable time scale. Our solution was to randomly select 200 points from each set of edge points, because that number allowed us to run our entire data set in about 30 minutes with a lab computer.



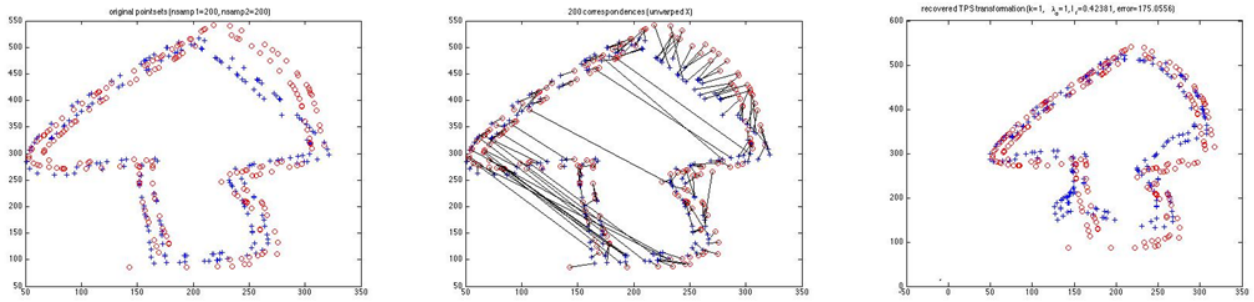
(a) A full pixel matrix (b) Edge detection on the image (c) 200 random edge points

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.

4.2 Image Distance

Once we had a manageable data set, we computed the image distance between images from the same user (sketches of the same password) and images from different users. The best algorithm we found for this task was a combination of shape context and affine transformation. Shape context allowed us to create a mapping from each point in one image to the closest matching point in another image, whis is intrinsically robust to translation. The matching of two points from different images was computed by minimizing a cost matrix encoding an estimation of the relative position of each point among its surrounding points.



(a) Original pointsets (b) Point-to-point mappings (c) Thin plate spline transformations

Figure 3: Performing shape context on two different sketches.

The cost matrix is found by drawing concentric bins around each point that are uniform in log-polar space, and counting out a histogram of how many neighboring points are in each of those bins. So if the bins for two points are represented by g(k) and h(k), the chi-squared difference cost of the points is[2]

$$C_S = \frac{1}{2}\sum_{k=1}^{K}\frac{[g(k)-h(k)]^2}{g(k)+h(k)}$$

We minimized this cost with point-to-point matchings using the Hungarian algorithm such that:

$$H(\pi) = \sum_i C\left(p_i, q_{\pi(i)}\right)$$

Since our point selection was random we also used dummy nodes here to account for outliers by limiting the highest cost of any single point. Once the cost matrix was minimized we computed the left over chi-squared matching cost, which we called the shape context cost between the two images. We now had three image distance metrics to treat as features for learning an acceptable password sketch.

4.3 Support Vector Machine

Our final task was to learn a decision boundary with which images could be classified using our three distance metric costs. Since these costs gave us data very close to being linearly separable, we constructed support vector machines to minimize classification error on the training set at the boundary. Across all three dimensions this was not very successful, but we recognized that the shape context and affine transformation costs were far more indicative than the thin plate spline transformation cost. We then minimized support vector machine error across a range of linear transformations. The best performing transformations weighted shape context 10 or more times heavier than affine transformation cost, and completely marginalized thin plate spline transformation cost. As a result we completely removed thin plate spline transformation cost from our SVM, only using affine and shape context cost.

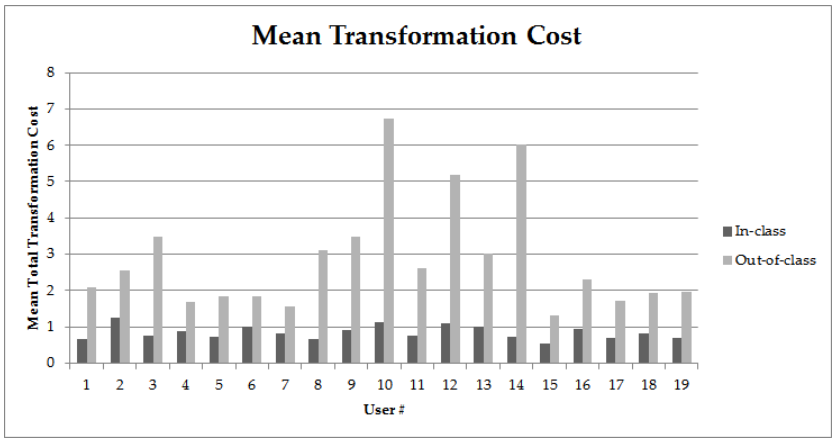


Figure 4: Mean total image transformation cost within class and without.

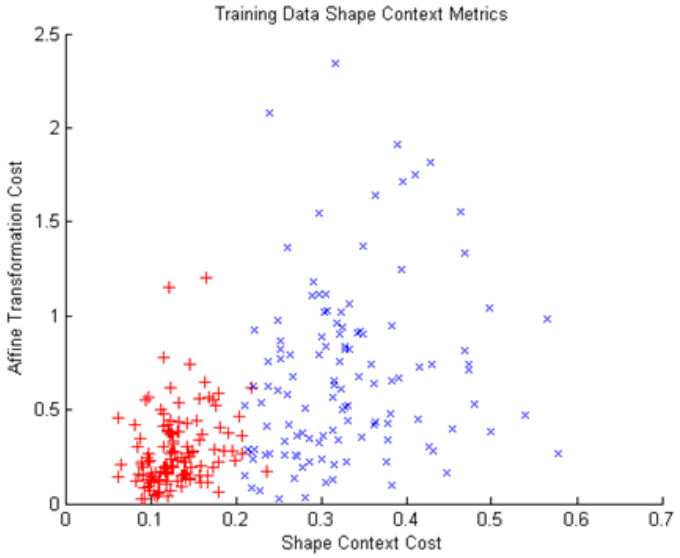


Figure 5: Our untransformed training set. Almost, but not quite linearly separable.

5 Results

Our initial results, while promising, were not quite what we had hoped for. Our soft-margin SVM achieved an error rate of roughly 15%, more than our stated goal of 10%. After some investigation, we were able to determine that a very few out-of-class outliers were throwing off our data enormously. Our solution was to trim the furthest outlying 1% of out-of-class examples from our training set. This had exactly the effect we desired, bringing our error rate to convergence around 3% for transformation factors $t > 10$.

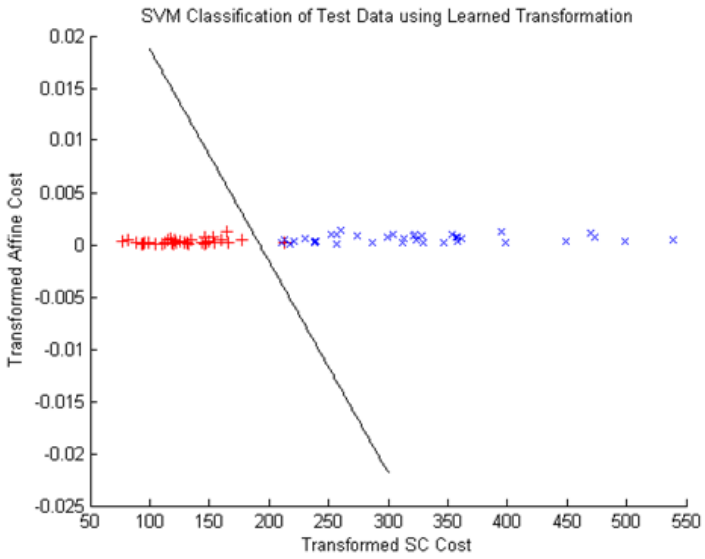


Figure 6: SVM decision boundary on the test set. $e = 0.01$ for this iteration.

When minimizing our training set error rate with respect to the transformation factor we had learned a weighted linear transformation that heavily emphasized the shape context cost over our other two features. Essentially, error rate was minimized when weighting the shape context cost over affine transformation and thin plate spline costs, as depicted below:

One disappointing result was the speed of our method. Each new picture requires 5-6 seconds to compute its transformation cost features when our program is running on a desktop machine. Presumably, this already unacceptable amount of time would increase when running on a smartphone. A user will not wait around for 5-10 seconds waiting for their password sketch to be recognized. A way around this might have been to select fewer data points, or perhaps (drastically) transposing all of our code into a language like C.

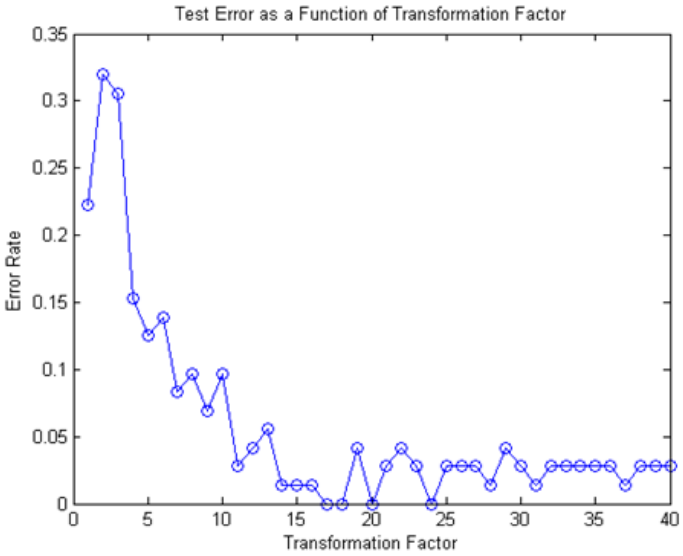


Figure 7: Test set error rate as a function of transformation factor.

6 Conclusions

While, in the end, we did not use any of the methods we had envisioned using at the outset of the project, we surpassed the performance milestone of 90% accuracy we had set for ourselves. This is obviously a major a positive; but, like all projects, ours still has much room for improvement. We did not collect nearly as much data as we had hoped to collect, and this shows in the volatility of our accuracy with respect to the random initialization of our training and test sets.

All in all, we are pleased with results we were able to achieve. We classified highly variable password sketches drawn by different people with a 97% success rate, an accomplishment that we believe could feasibly be extended to a functioning component of the Android, and perhaps eventually iOS, operating systems.

Notes

[1]R. Biddle, S. Chiasson, P.C. van Oorschot. Graphical Passwords: Learning from the First Twelve Years. ACM Computing Surveys 44(4), Article 19:1-41 (August 2012).

[2]S. Belongie and J. Malik (2000). "Matching with Shape Contexts". *IEEE Workshop on Contentbased Access of Image and Video Libraries (CBAIVL-2000).*