

Feature-based Sketch Recognition Using Gestural and Geometric Features

Pedro Davalos, Brandon Paulson, and Pankaj Rajan

Texas A&M University, Department of Computer Science

College Station, TX 77843-3112

{pedrodavalos, paulson.b, sachuin23} at gmail.com

Abstract

As pen-based interfaces become more popular in today's applications, the need for algorithms to accurately recognize hand-drawn sketches and shapes has increased. In many cases, complex shapes can be constructed hierarchically as a combination of smaller primitive shapes meeting certain geometric constraints. However, in order to construct higher level shapes it is imperative to accurately recognize the lower-level primitives. Two approaches have become widespread in the sketch recognition field for recognizing lower-level primitives: gesture-based recognition and geometric-based recognition. Our goal is to use a hybrid approach that combines features from both traditional gesture-based recognition systems as well as geometric-based recognition systems. We will show that a) we can produce an accurate recognition system that is robust to various users and drawing styles, b) we can determine which of the gestural and geometric features are most significant as well as identify those which are redundant, and c) we can derive relationships between different users and their styles.

Introduction

Hardware supporting pen-based input has become popular in recent years, as Tablet PCs, SmartBoards, and touch screens are becoming common modes of input for many applications. The use of pens and other sketching/writing tools have been around for centuries. Humans throughout time have learned to demonstrate ideas and communicate via writing and sketches. In today's time, sketches are used in a variety of domains to help convey ideas and designs. Computer-aided design (CAD) tools have been created to allow the visualization of abstract ideas; however, traditional CAD applications use less than intuitive interfaces. Sketching is a very natural choice as an alternative to the multi-tool selection paradigm. Many tools have been created which allow sketching to be easily incorporated into user interfaces [2][6]. But in order for sketching to be an effective means of input, we must develop recognizers that can accurately determine the intention of a sketcher. This can be a hard problem because users are unable to draw perfectly, sketches are often ambiguous, and every user has a different sketching style.

Two approaches have become standard to solving the sketch recognition problem. The first (and earliest) approach treats input sketches simply as two dimensional gestures [3][5]. These gesture-based recognition techniques typically focus on how a sketch was drawn rather than on what the final sketch actually looks like. The typical goal of these systems is to take an input stroke (a sampling of 3-D points in the form of x, y, and time value) and classify each one into a set of pre-defined gestures. This approach has the benefit of using mathematically sound classifiers which produce fast classifications, but

has the disadvantage of using feature sets which are user-dependent and require individual training by each user to give good recognition results. Furthermore, many of these gesture-based features produce systems which are very sensitive to changes in scale and rotation.

A newer approach has been to describe shapes geometrically, focusing on what the sketch looks like and less on how it was actually drawn [4][7][9]. Essentially, these geometric-based techniques are meant to take a single stroke as input as classify it as one of the pre-defined geometric primitives. These techniques are geometric in nature, because they compare a stroke to an ideal representation of each primitive using geometric formulas. Primitives can then be combined hierarchically to form more complex shapes using specialized grammars [1]. Since the geometric tests used by these systems focus more on what the sketch looks like, these recognizers are typically more user and style-independent. This also means that no individual (per user) training is necessary; the only training required is that which is necessary to determine numerous geometric thresholds. This leads to the disadvantage of such a system: geometric-based recognizers typically use numerous thresholds and heuristic hierarchies which are not mathematically sound. This makes inferences about generalization hard to determine because classification is not statistical.

Our goal is to find a way to combine the gesture-based and geometric-based approaches in such a way to take advantage of the positive aspects of each (accurate classification, user independent, mathematically sound, etc.) while avoiding many of the disadvantages. As is the case in many previous works [4][7][9], we focus simply on classifying low-level primitive shapes. We propose using a combination of gesture-based features, along with the values of various geometric tests as features into a statistical classifier. By using mainly geometric features, we hope to preserve user independence. Furthermore, we hope that using a statistical, feature-based classifier keeps our approach mathematically sound and will allow us to make inferences about the generalization of our recognizer. We have chosen to classify single strokes into one of nine different shape classes (arc, line, curve, circle, ellipse, helix, spiral, polyline, and complex); the shape set used in a previous geometric-based recognizer [4]. In this paper we will show that a) combining gesture-based and geometric-based features into a statistical classifier yields accurate recognition results, b) we can perform feature subset selection to determine what features are the most significant to recognition, and c) we can determine specific relationships between users and their styles.

Previous Work

Gesture-based Recognition

One of the first works to introduce us to the concept of pen-based input devices was Sketchpad, developed in 1964 by Ivan Sutherland [8]. This seminal work proved to be beyond its time as pen-based interfaces would not catch on until the 1990s. In 1991, Dean Rubine introduced the first pen-based input gesture recognition system, GRANDMA [5]. This toolkit allowed users to specify single stroke gestures that could

be trained and learned through a simple linear classifier. Rubine proposed thirteen features which could be used to classify simple gestures with an accuracy of 98% on a fifteen-class gesture set when trained with at least fifteen examples. He also proposed two different techniques that could be used to reject ambiguous or non-gestures. Rubine's work would later be extended in 2000 [3]. In this paper, Long et al. added nine new features to Rubine's existing set. They performed multi-dimensional scaling to identify correlated features and ultimately found an optimal subset that contained eleven of Rubine's original thirteen features along with six of their own. Both of these works proved to be helpful in recognizing two-dimensional gestures, but when applied to sketch recognition problems the accuracy of these approaches is not optimal. The nature of the feature sets used by these recognizers requires that gestures be drawn the same way and to the same scale every time they are drawn. For example, a clockwise gesture would not be the same as a counter-clockwise gesture. When treating sketched shapes as gestures these approaches do not perform well because it puts constraints on how users must draw. We want to be able to create recognition systems which are user-independent and allow users to draw as they naturally would without having to worry about issues like where to start a stroke or which direction to draw certain shapes.

Geometric-based Recognition

Because of the drawing constraints imposed by gesture-based classifiers, the most recent shift in sketch recognition has been towards a geometric-approach, which puts virtually no drawing constraints on the user. Essentially, shape grammars such as LADDER can be used to define higher level shapes as a combination of lower level primitive shapes meeting certain geometric constraints (such as intersects, coincident, parallel, etc.) [1]. In order for these higher-level recognition schemes to be effective, it is important that the low level primitive shapes are accurately recognized. There have been many geometric-based recognizers that have been developed to recognize low-level primitive shapes [4][7][9]. Unlike gesture-based techniques, these recognizers do not use statistical classifiers. Instead, they focus on determining the error between a sketched shape and its ideal version using a series of geometric tests and formulas. Some recognizers focus on developing a universal error metric such as the feature area error metric proposed in [9]. However, universal error metrics can be hard to compute and describe for more complex primitive shapes such as spiral and helix. In order to support more primitives, some recognizers use different error metrics for each primitive [4]. These recognizers then rely upon heuristic hierarchies and numerous thresholds to order shape interpretations. Such an approach makes it hard to generalize and prove a recognizers success for future sketches.

Methods

Features

As mentioned before, our goal is to combine gesture-based and geometric-based approaches in such a way as to take advantage of the benefits of both techniques. We will use a statistical classifier (namely a quadratic classifier) that uses a feature set containing both gestural and geometric features. This approach will be more

mathematically sound than using heuristic hierarchies, will maintain user independence because of the addition of geometric features, and will still enable us to return multiple, ranked interpretations - a major benefit as described in [4]. Table 1 gives a list of the feature set we used. Our feature set initially contained 44 total features. Later, we will describe an optimal subset of these features which we obtained through feature subset selection (shown in bold on Table 1). The first 31 features come from geometric tests described in [4]. The remaining 13 features are the classical gesture-based features used by Rubine [5]. The descriptions of each of these features are beyond the scope of this paper, so we will refer one to these papers for the implementation details of each feature.

1. Endpoint to stroke length ratio*	12. Curve least squares error*	23. Spiral fit: avg. radius/bounding box radius ratio	34. Length of bounding box diagonal
2. NDDE*	13. Polyline fit: number of sub-strokes*	24. Spiral fit: center closeness error	35. Angle of the bounding box diagonal
3. DCR*	14. Polyline fit: % of sub-strokes pass line test	25. Spiral fit: max distance b/t consecutive centers	36. Distance between endpoints
4. Slope of the direction graph*	15. Polyline feature area error	26. Spiral fit: average radius estimate	37. Cosine of angle between endpoints
5. Maximum curvature	16. Polyline feature least squares error	27. Spiral fit: radius test passed (1.0 or 0.0)	38. Sine of angle between endpoints
6. Average curvature [†]	17. Ellipse fit: major axis length estimate	28. Complex fit: number of sub-fits	39. Total stroke length
7. Number of corners	18. Ellipse fit: minor axis length estimate	29. Complex fit: # of non-polyline primitives	40. Total rotation
8. Line least squares error	19. Ellipse feature area error	30. Complex fit: percent of sub-fits that are lines*	41. Absolute rotation
9. Line feature area error	20. Circle fit: radius estimate*	31. Complex score / rank*	42. Rotation squared
10. Arc fit: radius estimate	21. Circle fit: major axis to minor axis ratio*	32. Cosine of the starting angle	43. Maximum speed
11. Arc feature area error	22. Circle feature area error	33. Sine of the starting angle	44. Total time

Table 1: Features used by our recognizer. Implementation details for features 1-31 can be found in [4]. Details for features 32-44 can be found in [5]. Bold features are those chosen as the optimal subset through feature subset selection.

* - one of the top 10 features according to feature subset selection
[†] - not explicitly used in [4] but can be easily calculated

Data

To perform our tests, we used a dataset consisting of 1800 total sketch examples (the same dataset from [4]). Each example consists of a single stroke and is labeled with the intention of the original sketcher. A stroke is defined as the set of points (x coordinate, y coordinate, and timestamp) sampled between pen-down and pen-up events. The data samples came from 20 different users. Each user provided 90 samples – 10 of each shape class. Figure 1 shows some examples from the dataset.

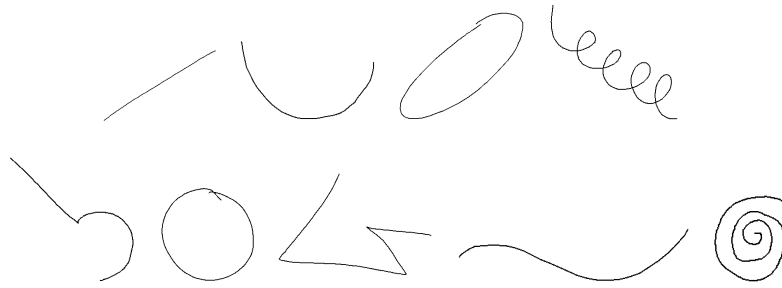


Figure 1: Examples of each shape class from our dataset.

Quadratic Classification

Classification of hand-drawn sketches into primitive shape components can be accomplished if sufficient discriminant features are extracted which characterize the differences between each shape class. We have used an ample number of discriminating features which have been extracted from both a geometric and gestural standpoint. Since our goal is to classify new sketches with an efficient, user independent, near-real-time system, we will utilize a quadratic classifier for recognition of the primitive hand-drawn sketches. This classification method implies multiple assumptions including a unimodal Gaussian distribution of the classes in feature space. Further implications from quadratic classifiers are the inability to adapt in real-time to custom fit user styles, or to learn any new user trends for each of the classes. However, our assumption is that the primitive sketch classes are fixed and invariant to users. This assumption of class invariance and user-independence is primarily due to the flexibility of the multi-layered architecture of geometric-based sketch recognition, where higher level recognizers perform the tasks of reconstructing more elaborate shapes by the composition from basic primitives.

Further aspects of quadratic classifiers include the requirement of sufficient training data that will accurately model the true density of the data and will generalize all new test data in the real world. As previously described, we have used a large enough dataset from a diverse number of users, where the samples were labeled accordingly, with sufficient variance in styles. Consequently, given our requirements, a quadratic classifier appears as an optimum method to perform the automated classification.

Additional considerations for the selection of the automatic classifier include the cost of misclassification. This application does not have any custom misclassification costs for each class, as a misclassified stroke could be re-sketches or cleared if the assigned label is significantly different than as intended. Alternatively, since the classifier can return multiple, ranked interpretations, the correct interpretation could be chosen with simple changes in the pen-based interface (such as clicking the pen button to go to the next interpretation). Furthermore, misclassifications due to ambiguities will always be unavoidable such as a circle and an ellipse with similar axis, and a slight curve with a line.

Our decision to use a quadratic classifier for primitive hand drawn sketches gives us the option to work with the original raw feature space, consisting of the 44 features we have

identified and extracted. We can also project the data to a reduced feature space that can potentially increase our classification performance. These projections to lower dimensional space can be achieved by Linear Discriminant Analysis (LDA) or by Principal Component Analysis (PCA). We evaluated classification performance on both of these reduced dimension spaces as well as classification on the original feature space. LDA projections try to find the basis that will preserve the best class discrimination information, while PCA projections align the data in the basis of greater variance. PCA projection vectors are defined by the eigenvectors with the highest eigenvalues of the covariance matrix. And LDA, through Fisher's Linear Discriminant, finds the linear function that preserves the greatest class separation by analyzing the between-class scatter and the within-class scatter.

Results

Regularization

Before performing any classification experiments, we carefully analyzed the data, as the quadratic classifier requires much computation to invert the covariance matrix of a given feature-space. When the data is not normalized or there is large variance in the ranges of each feature, numerical instability will result from the near singular covariance matrices. However, we can overcome this phenomenon through ridge-regression, which introduces a regularization parameter to remove singularities from the covariance matrix. Preprocessing of our data indicated a non-normalized distribution in the raw feature space, with singularities in the covariance matrices, but as expected, ridge-regression solved the instabilities by removing singularities. For our implementation we used a regularization parameter of 0.01.

After the regression parameter for the covariance of raw feature space was optimized, and the eigenvalues of the data were analyzed to select the optimum eigenvectors for PCA projections, the classification performance was evaluated through cross-validation with random sub-sampling of the dataset (82% train, 18% test). We performed multiple classification experiments using the quadratic classifier on each of the three feature spaces previously described (original raw feature space, LDA space, and PCA space).

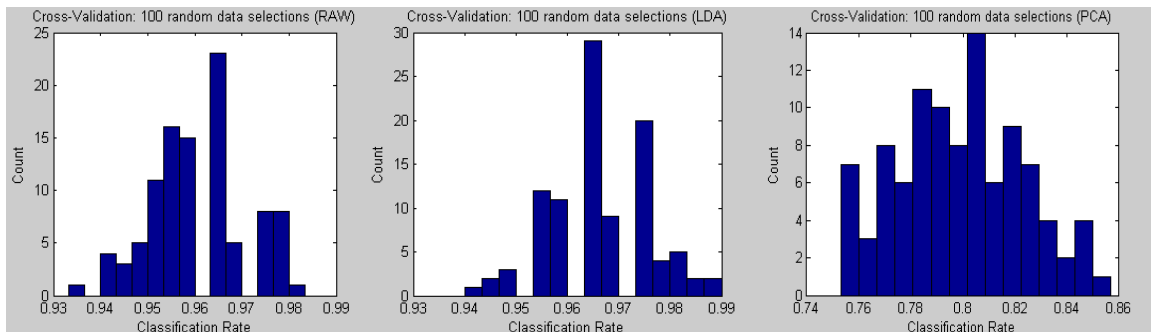


Figure 2: Histograms showing the accuracy of a quadratic classifier through 100-fold cross validation experiments with random sub-sampling on the raw feature space (left), LDA space (middle), and PCA space (right). The raw feature space had an average accuracy of 96.3%, LDA space 96.7%, and PCA space 79%.

Our initial findings, as seen in Figure 2, indicate positive results with classification rates of about 96% when using the original raw features and when using the LDA projections. However, PCA projections did not achieve the same high accuracy, with performance at about 79% successful classification. Sub-optimum classification performance on PCA space indicates that the discriminatory information is not contained in the variance of the data. However, the sub-optimum performance on PCA space is still significantly higher than the chance expectancy of about 11% since we have nine different classes.

Feature Subset Selection

Further optimization to our classification performance can be achieved by feature subset selection, a technique used to analyze the features that characterize each data sample in order to determine what the optimum subset of features are that maximize classification. We implemented a wrapper subset selection algorithm with our quadratic classifier as the objective function and performed the process for each of the three feature spaces. The motivation for finding an optimum subset of features serves a dual purpose by improving classification performance through only using the optimum features, and second by reducing computational requirements through decreasing operations when extracting features. Our implementation of feature subset selection was accomplished through sequential forward selection which consists of a greedy approach that starts from an empty set, and sequentially adds features by incorporating the feature that optimizes performance when added to the set at the current iteration. The rationale for implementing forward selection is that we expect an improvement in performance due to the greedy nature of forward selection; especially since our raw feature space consists of sufficient independent features.

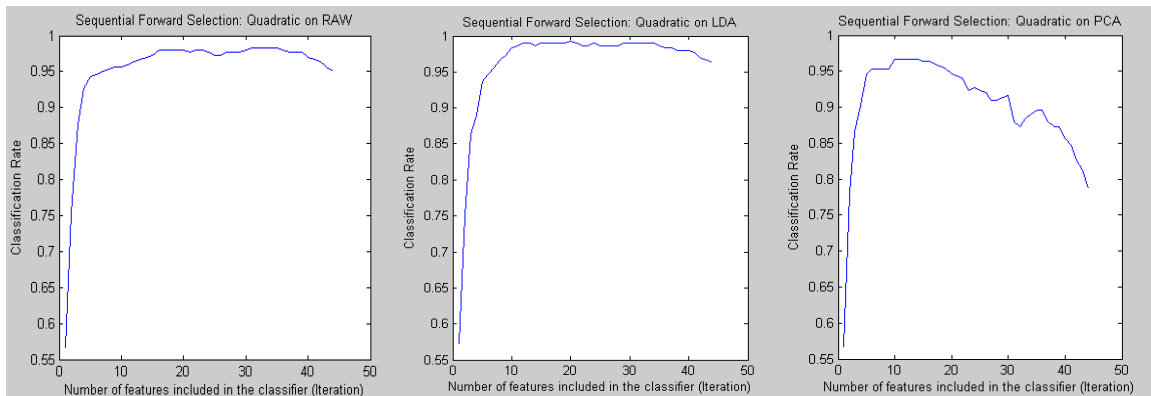


Figure 3: Classification rates as features are added through sequential forward selection for the raw feature space (left), LDA space (middle), and PCA space (right). The raw feature space reached an optimum accuracy of 98.33% with 33 features, LDA space 99.33% with 20 features, and PCA space 96.67% with 10 features.

Our results from subset selection (seen in Figure 3) indicate that both the raw features and LDA projections achieve highly accurate classification rates after adding just a few features. Both achieve about 90% success with only the first five features. Afterwards,

performance does not experience statistically significant change when adding additional features. We found an optimum rate of around 98-99% for both raw features and LDA projections. In addition, subset selection when projecting to PCA space achieved over a 96% classification rate with only the first 10 optimum features projected to five dimensional PCA space. The interesting phenomenon is that performance degrades to the original 79% estimate when all the features were added to the set. This trend occurs because only a subset of the raw features contain discriminatory information in the variance of the data, yet the remaining features that do not contain information from the variance hinder classification performance when incorporated to the feature subset. This occurs in PCA space but not in LDA or original feature space, because PCA does have a mechanism for weighting individual features.

We further evaluated the optimum feature subset for each feature space through multiple fold cross-validation with random sub-sampling of the data. This process verified our results, indicating no statistically significant difference of classification rates of the raw features and LDA space when classification was based on the optimum subset for each space rather than the full feature set. The average classification performance was about 96% for both raw and LDA feature spaces when testing with the previously specified conditions. However, classification performance of PCA projected data when using the optimum subset resulted in a significant increase from the original 79% to about 94%, indicating that the 10 selected features for PCA space contain discriminatory information in the variance.

The effect from forward selection of raw and LDA space, where performance quickly improves and remains constant indicates that most features are decent and contain enough discriminating information since performance does not significantly degrade with the full feature set. However, the phenomenon from forward selection of the PCA space, where performance is optimized quickly and then monotonically degrades when adding subsequent features, captures the effect produced when the features that contain discriminatory information are mixed with noisy features.

Ensemble Learning

After obtaining highly accurate results through training and optimizing quadratic classifiers on various feature spaces, our next approach to further improve classification rates was to implement an ensemble learning system, with the motivation of enhancing successful classification rates by combining various independent classifiers. Our approach consisted of a parallel architecture with three quadratic classifiers and a voting mechanism as the combiner function. This architecture with the voting mechanism achieves results with higher confidence if multiple independent classifiers result with the same class assignment. Yet, at the same time, computing resources remain unchanged, as each classifier operates on a smaller subset of features. Our implementation was designed such that each of the three independent classifiers operated on different feature subsets, the first classifier operated with the first 22 raw features, the second operated with the remaining 22 raw features, and our third classifier consisted of a quadratic classifier trained on the LDA projections of the full set of raw features. Results from the ensemble learning system were consistent with our expectations, where we increased

performance to 98% when two of the three classifiers agreed with the output, and we obtained even higher rates of 99.9% successful classification when all three classifiers agreed with the output. The drawback from this architecture is that uncertainties arise when all classifiers disagree on an output label, furthermore, 99% of the time at least two classifiers agreed, but only 85% of the time resulted in a unanimous decision.

Discussion

Feature Subset Selection

One of most interesting observations we made was that we can still achieve a high classification rate, around 95%, with only the top ten features (denoted by asterisks in Table 1). These features contain a lot of information and are highly independent of one another. Even more interesting, however, is the list of features that were not chosen by feature subset selection. Before performing our experiments, we hypothesized which features we believed contained a lot of information, and which features we believed were less relevant and would likely be removed. We were surprised to find that some of the features that we initially believed would be important were never added by the subset selection algorithm. Upon further observation we determined that although these features contained a lot of information, that information was redundant with that of another feature. Because we employed a greedy, sequential forward search for feature subset selection, new features were only added if they maximized accuracy at that particular iteration. Since redundant features would provide no additional information, accuracy would be the same as the previous iteration. Therefore, if another feature exists (even if it is not as “good” as a redundant feature) and it provided even a small bit of extra information, then it would be chosen over the redundant feature. So even though one feature by itself may contain more information than another feature by itself, it is possible that it will not be chosen by a feature subset selection algorithm simply because its information content is the same as a previously added feature.

Of the 31 features from the geometric-based recognizer, five were never added by the subset selection algorithm. The first, average curvature, was a value we added for this paper only. It had not been shown to be helpful in previous works, but we added it to our list to see if it possibly contained any information; obviously it did not. The two tests which were used by the geometric recognizer to determine whether a stroke was a line or not (line least squares error and line feature area error) were also not added by the subset selection algorithm. Upon further investigation, we can say that these features are redundant because another feature (endpoint to stroke length ratio – which happened to be the best overall feature according to the subset selection algorithm) has previously been shown to be a good indicator for a line test [7]. The radius of an arc fit and the radius of a spiral fit can also be considered redundant features because they are computed in the same way as the radius of a circle fit. The only difference between the three values is the manner in which the center points are estimated. Finally, the feature area error of an ellipse fit was not added. Again, this feature is redundant and provides no more information than the feature area error of a circle, which was added first.

Seven of the original thirteen gesture-based features were left out by the feature subset selection algorithm. The cosine of the starting angle was never added, arguably because it contains no more information than the sine of the starting angle. The cosine and sine of the angle between the endpoints and the starting points were also not added. This must indicate that this angle carries no (or negative) information when applied to freely drawn sketched shapes that can be drawn to any scale or rotation. Furthermore, two features which are scale-dependent were not added (length of the bounding box diagonal and total stroke length). Since shapes could have been drawn to different scales, it is likely that these features actually contributed negatively to recognition. Not surprising, these features were also shown to be insignificant through the multi-dimensional scaling technique used in [3]. Finally, two other rotation-related features were left out – absolute rotation and rotation squared. These features proved to be redundant as soon as total rotation was added to the feature list.

User Robustness

One of the fundamental questions we have asked is whether or not our approach is user independent. We also have posed the question of whether or not we can derive specific relationships between users based on our given feature set. To begin answering these questions we first looked at how well a user performed as an individual trainer and how well that same user performed as an individual tester. By identifying “good” users from “bad” users we can derive an optimum sequence of users that can be used for training which will give us a baseline to compare a more realistic random sub-sampling of users to. To identify “good” training users we trained with a single user and tested with the remaining nineteen users. Likewise, to identify “good” test users we tested with a single user and trained with the remaining. Figure 4 shows the results from these tests.

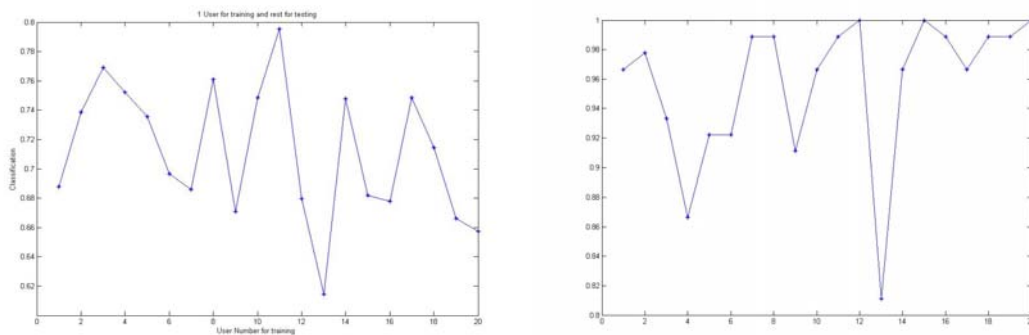


Figure 4: Results of training with a single user and testing with the remaining users (left) along with the results of testing with a single user and training with the remaining users (right). Notice that the two graphs are not necessarily correlated. Just because a user is a “good” trainer does not mean that they will also be a “good” tester. One exception, however, is user 13 who was both the worst trainer and tester.

When training with a single user, we observed that there was great variability in the styles of the users because the performance of each user as a trainer was not consistent. We believe that the user whose style of drawing is the closest match to the drawing style of most of the users will be a better trainer while users with entirely different styles will be

worse trainers. We observed that user number 11 was the best trainer whereas user number 13 was the worst trainer.

When testing with a single user, we observed that there is not a one to one relationship between how well a user performs as a trainer and how well he or she performs as a tester. In other words, if a user is a “good” trainer it does not mean that he or she is a “good” tester. Previously, we had seen that user number 11 was the best trainer, but in this experiment he fared at 98% classification as a tester compared to the close to 100% classification achieved by user numbers 12, 15, and 20 (each of which had a very poor record of close to 68% classification as an individual trainer). This lead us to the conclusion that drawing style of user numbers 12, 15 and 20 might be well captured by the combination of other user styles. Therefore, they performed very well as a tester, but vice versa it is not the case. Likewise, user number 11 as a trainer was able to capture most of the drawing styles, but there were certain aspects of his style which were not captured by the combined user data. Therefore, his performance as the tester was not quite 100%. One consistency was user number 13, who continued to be the worst performer in both experiments. We concluded that the drawing style of user 13 is completely unique and is not captured by the drawing styles of any of the other users.

11	3	8	4	17	10	14	2	5	18	6	1	7	15	12	16	9	19	20	13
11	10	13	12	4	5	2	9	3	6	1	8	14	17	16	7	18	15	19	20

Table 2: Top row indicates the rank of each user (from left to right) as an individual trainer. Bottom row indicates the order in which users were added to the training data through user subset selection. Notice, for example, that although user 13 is the worst overall trainer his data is added very early in the selection process, thus showing that the classifier performs better with a higher variability of styles.

After identifying “good” trainers, we implemented a sequential forward selection procedure (much like we did for our feature subset selection) to determine the order in which we could add data from a particular user to our training set, such that we maximize recognition for the remaining test examples. We selected the best trainer first and then sequentially selected the user whose addition to the training set improves the overall classification at that particular iteration. Through this procedure we noticed that the best trainers were not necessarily added in the same order as their rank as individual trainers, as seen in Table 2. It was surprising that the addition of worse trainers to the training data was improving the overall classification. For example, user number 13, which from the previous two discussions was a worst trainer as well the worst tester was added very early at the third position. Also, user number 10 which was the sixth best trainer was added second. This indicates that the classifier will perform much better if the training data has much more variability in it. This is in perfect agreement to the basic understanding of pattern classification fundamentals, which is that the more independence and variability we can bring to the data, the better it will generalize and perform overall.

By performing user subset selection, we can determine the optimal sequence in which we can add a particular user’s data to our training set such that we maximize classification.

However, knowing beforehand this sequence is unlikely. Therefore, we compared this optimal sequence to a random cross-validated sequence which most closely resembles a real-life scenario (Figure 5). We observed that after the addition of four users' data to the training data the performance of the randomly sampled data started converging closely to the optimum user selection sequence. Hence, we believe that our classifier is robust enough to give a decent performance, provided we have training data to represent at least four independent styles.

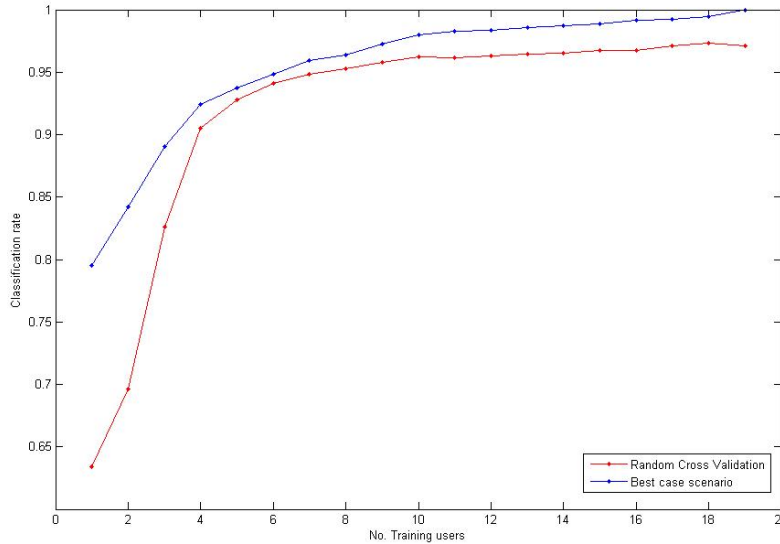


Figure 5: Classification over time as more users are added to the training set. The blue line indicates the optimal sequence obtained through user subset selection, while the red line indicates the random cross-validated addition of random user data (more realistic). Notice once we have four users represented in our training set that the cross-validation line approaches the optimal line.

User Style Similarities/Differences

As a final experiment, we tried to analyze if we can use our feature set to derive some relationships between the different users (and their styles) based on the data available to us. For this purpose, we trained our classifier with a single user and tested on other users individually to see the relationship between each user. We obtained a 20 by 20 relationship matrix which showed the trainer/tester relationship between each pair of users (Figure 6). As we analyzed the matrix we observed that our best trainer (user number 11) has a good similarity of sketching (shown as red in Figure 6) with all other users whereas our worst performer (number 13) did not share many similarities with any of the other users (hence the mainly blue squares). This also shows why this user was the worst trainer as well as the worst tester. Finally, we can see that the matrix is not symmetric thus further illustrating the concept that the relationship between training and testing is not the same.

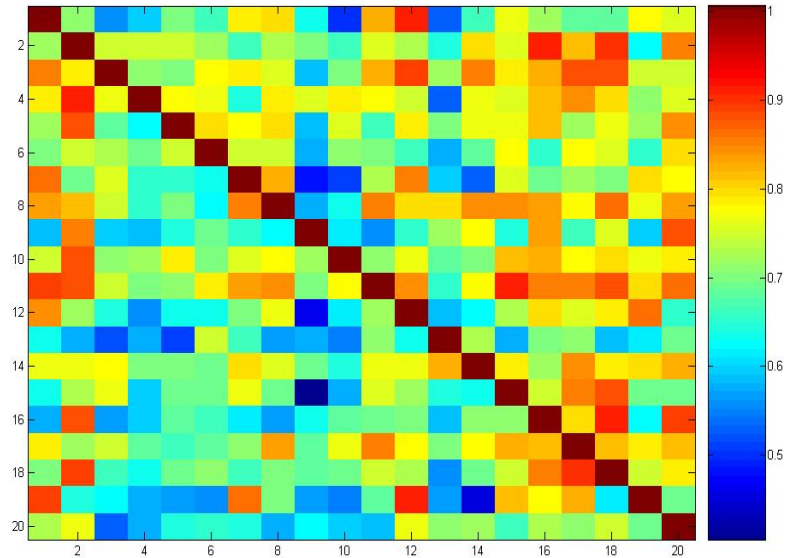


Figure 6: Relationship matrix between individual pairs of users for a single trainer (y axis) and a single tester (x axis). Note that this matrix is not symmetric, for example training with user 1 and testing with user 3 is not the same as training with user 3 and testing with user 1.

Future Work

One of the most important things we would like to do for future work is to see how our approach generalizes against other classes of complex shapes. For our tests, we used a complex shape consisting of one line and one arc (the complex shape used in [4]). In order for our recognizer to be used in current sketch applications, it must support the capability of recognizing multiple complex interpretations. Obviously, it would be impractical to create a class for every possible primitive combination, which is infinite. In theory, a complex shape is simply an outlier class that occurs when it is determined that a shape is not one of the basic primitive shapes. Therefore, it is possible that the addition of other complex shapes would still be recognized using our approach. However, there is also the possibility that our recognizer is simply recognizing one line, one arc interpretations and not a generalized complex interpretation. We hope to collect more complex shape data to determine whether or not this is the case.

Also, as mentioned before, we were able to determine stylistic differences and similarities between users. This observation leads to the curious question of whether or not our technique could be used to classify users in addition to their sketches. This would be beneficial in various collaborative settings when user identification is just as important as sketch identification. Furthermore, techniques such as k-means clustering could be used to determine different clusters representing users with similar styles. By classifying an individual user into one of these clusters, default thresholds could be modified to represent those of the cluster, thus improving recognition accuracy on a per user basis. As seen in our user robustness discussion, the addition of four users to the training set

made the accuracy of the random sub-sampling approach closely to the optimum sequence. This observation could potentially tell us that four clusters would be a good starting point for k-means clustering.

Conclusion

We have taken two traditional approaches to low-level sketch recognition and combined them to create a hybrid recognition scheme. By combining gesture-based and geometric-based features into a statistical classifier, we have created a recognizer that can produce highly accurate classification while maintaining user independence and allowing users to sketch freely. Furthermore, our approach is mathematically sound and can be generalized across a large number of users and styles. We have also shown, through subset selection, which features are statistically significant and which features contain redundant information compared to others. By choosing a smaller subset of features, we can speed computation time while sacrificing little accuracy. In addition, we were able to derive relationships between different users and their styles, as well as determine how users perform both as trainers and as testers.

References

- [1] Hammond, T. and Davis, R. LADDER, A Sketching Language for User Interface Developers. *Computers & Graphics* 29, 4 (2005), 518-532.
- [2] Long, Jr., A.C., Landay, J.A. and Rowe, L.A. "Those Look Similar!" Issues in Automating Gesture Design Advice. In *Proc. of the 2001 Workshop on Perceptive User Interfaces*, ACM Press (2001), 1-5.
- [3] Long, Jr., A.C., Landay, J.A., Rowe, L.A. and Michiels, J. Visual Similarity of Pen Gestures. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press (2000), 360-367.
- [4] Paulson, B. and Hammond, T. PaleoSketch: Accurate Primitive Sketch Recognition and Beautification. In *Proc. of the 13th International Conference on Intelligent User Interfaces*, ACM Press (2008), to appear.
- [5] Rubine, D. Specifying Gestures by Example. In *Proc. of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press (1991), 329-337.
- [6] Saund, E., Fleet, D., Larner, D. and Mahoney, J. Perceptually-Supported Image Editing of Text and Graphics. In *Proc. of the 2003 ACM Symposium on User Interface Software and Technology*, ACM Press (2003), 183-192.
- [7] Sezgin, T.M., Stahovich, T. and Davis, R. Sketch Based Interfaces: Early Processing for Sketch Understanding. In *Proc. of the 2001 Workshop on Perceptive User Interfaces*, ACM Press (2001), 1-8.
- [8] Sutherland, I.E. Sketch Pad A Man-Machine Graphical Communication System. In *Proc. of the SHARE Design Automation Workshop*, ACM Press (1964), 6.329-6.346.
- [9] Yu, B. and Cai, S. A Domain-Independent System for Sketch Recognition. In *Proc. of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, ACM Press (2003), 141-146.