

MARQS: RETRIEVING SKETCHES USING DOMAIN- AND STYLE-INDEPENDENT FEATURES LEARNED FROM A SINGLE EXAMPLE USING A DUAL-CLASSIFIER

Brandon Paulson, Tracy Hammond

Sketch Recognition Lab, Texas A&M University, College Station, Texas, USA
{bpaulson; hammond}@cs.tamu.edu

ABSTRACT

Mouse and keyboard interfaces handle traditional text-based queries, and standard search engines provide for effective text-based search. However, everyday documents are filled with not only text, but photos, cartoons, diagrams, and sketches. These images can often be easier to recall than the surrounding text. In an effort to make human computer interaction handle more forms of human-human interaction, sketching has recently become an important means of interacting with computer systems. We propose extending the traditional monomodal model of text-based search to include the capabilities of sketch-based search. Our goal is to create a sketch-based search that can find documents from a single query sketch. We imagine an important use for this technology would be to allow users to search a computerized laboratory notebook for a previously drawn sketch. Because such as sketch will have initially been drawn only a single time, it is important that the search-by-sketch system 1) recognize a wide range of shapes that are not necessarily geometric nor drawn in the same way each time, 2) recognize a query example from only one initial training example, and 3) learn from successful queries to improve accuracy over time. We present here such an algorithm. To test the algorithm, we implemented a proof-of-concept-system: MARQS, a system that uses sketches to query existing media albums. Preliminary results show that the system yielded an average search rank of 1.51, indicating that the correct sketch is presented as either the top or second search result on average.

KEYWORDS

sketch recognition – sketch-based interfaces – search by sketch – feature-based recognition

1. INTRODUCTION

Due to the emergence of sketch and gesture recognition in the user interface community, tools have been developed to allow for easy integration of sketch recognition technologies into traditional user interfaces [1][2][3][4]. Sketch and gesture recognition deals with combining machine learning with feature-based and geometrical-based techniques to identify hand-drawn symbols and shapes, usually with the aid of a Tablet PC or Smartboard. These technologies are important because it has been shown that gestures or shapes are typically easier to remember than basic textual commands [5], thus motivating researchers to improve the integration and recognition of these forms of interaction into user interfaces. Our goal is to recognize unconstrained sketch symbols, drawn with any number and order of strokes, in any scale and rotation, and with no restrictions on the types of diagrams drawn.

Low-level, single-stroke recognizers have been developed using linear classification techniques [6][7]. However, a downfall of these systems is that their feature sets make the systems user- and style-dependent. In order for strokes to be recognized,

they must be drawn in the same manner each time. For example, Figure 1 shows two sketches that should be the same, but because they were drawn differently, are typically misrecognized by these classifiers. In our system we try to choose features which place little or no constraints on how users must draw strokes.

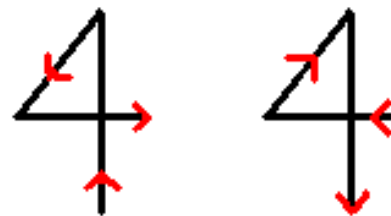


Figure 1: Two symbols that look identical but are drawn in two different directions. Examples like these are typically misrecognized by many recognizers who choose features which place drawing constraints on the user.

Much work has been done in expanding these single-stroke recognizers into a multi-stroke domain [8][9][10], but most of these techniques require extensive training before a query can be executed in the classifier. Furthermore, some of the systems only recognize a limited number of multi-stroke symbols. Like most recognition domains, training can be tedious for most users, causing them to stray from using the application. In our system, we have implemented and combined two separate classifiers: 1) a classifier that learns and classifies from a single example to create a sketch system that is immediately usable after a single example, and 2) a classifier that takes advantage of multiple examples as they become available from queries, creating a sketch system that becomes more accurate through use. By combining these two classifiers, we create a recognition algorithm that is immediately usable after a single example, but continues to learn from future interactions.

This paper presents an algorithm that identifies multi-stroke sketches using a set of global features that are both domain- and style-independent. Because our features are domain independent, we can use our techniques to recognize sketch symbols in a variety of domains. To give a real-world example of how our approach could be implemented into a sketch-based interface, we have created an application called **MARQS**, **Media Albums Retrieved by a Query Sketch**. The MARQS application contains personal photo and music albums that can be retrieved by users through multi-stroke sketches, which users designate during album creation. In addition, the system uses search queries as a way to train itself as the user interacts with the program. Similar techniques could be employed in other sketch interfaces as the user corrects recognition error. This form of retrieval could prove to be a more intuitive way of searching for me-

dia rather than using textual, metadata keywords as visual recall and muscle memory play an important role in recall for some users. As the user creates a new album he or she associates a sketch with the album that can be used as a search key at a later time. The system requires only one instance of the sketch to be drawn by the user at the time the album is created. Furthermore, sketches can be drawn using multiple strokes, in any order, to any scale, and in any rotation, making the system more user-friendly. As the user interacts with the system and performs searches, the system will train itself from query sketches. This training will increase recognition accuracy over time and occurs unobtrusively in the background of the application.

2. PREVIOUS WORK

One could argue that the very beginning of the field of sketch and gesture recognition began with Ivan Sutherland's Sketchpad system. Sketchpad was developed as a way for users to interact with computers with a light pen instead of communicating with typed instructions. Furthermore, the light pen used with Sketchpad was capable of not only drawing pictures to the screen but also included tracking technology that was used to identify and locate parts of previously drawn images. While Sketchpad was not a true "sketch recognizer," it introduced the idea of communicating with a computer using gestures and sketches rather than typed commands and provides motivation for much of the current work in sketch recognition [11].

2.1. Feature-Based Recognizers

Progress in the realm of actual gesture recognition was achieved with the work of Dean Rubine. In [7], Rubine developed a gesture recognition toolkit named GRANDMA. GRANDMA allowed single stroke gestures and their semantics to be trained and later recognized. In order for this to occur, Rubine proposed thirteen features that could be used to describe a gesture. The values of these features could then be put into a vector and later evaluated in a linear function using trained weights. In addition, Rubine used two different rejection methods which could be used to reject "bad" gestures [7]. Rubine's work was later extended by Long, et al. with a series of experiments used to determine how users perceive similar gestures [4]. Based on these results, a computational model was formed and multi-dimensional scaling was used to determine the relevant geometric properties. In the end, a feature set was chosen that consisted of eleven of the features used by Rubine, along with six new features [6]. Both of these systems input their feature sets into a linear classifier in order to determine the classification of a given stroke. Part of the recognition system described in this paper will use a similar linear classification technique; however, a set of global features describing the sketch as a whole will be used rather than a set of features describing a single stroke. This will allow the system to support sketches drawn using multiple strokes. A major downfall of using a linear classification technique is that extensive training is required in order to produce an accurate system. The described system resolves this by only resorting to the linear classifier whenever enough examples are provided to give accurate recognition. These examples will be learned unobtrusively as the user queries sketches in the database. This will eliminate the need for users to initially enter multiple training examples whenever they create a new media album.

Multi-stroke, feature-based recognizers have already been created [8][9][10]. In [10], a recognizer based on the Hidden Markov Model was created which used a combination of global features and local features. However, these HMM chains require extensive training and adapt based on individual users' drawing

habits, making it style-dependent. In [8], two recognizers were created: one that required training, and one that did not. The recognizer that required training required many examples to train itself, and the recognizer that did not require training did not allow users to specify new gestures without hand-coding. While our approaches differ, the work presented in [9] by Kara and Stahovich is probably most similar to what we present in this paper. A recognition system was implemented using a combination of four different recognizers. Like our approach, their algorithm could recognize shapes after only a single example; however, it is not an interactive learning system and does not learn from future sketches as our system does.

2.2. Geometric-Based Recognizers

Other single-stroke recognizers have been created that do not use linear classification techniques, but rather, use geometric properties to classify strokes in basic primitives. Two such systems were created by Sezgin et al. and Yu et al. In Sezgin [12], a system was described that was composed of 3 stages: approximation (fitting the most basic geometric primitives), beautification (modifying the output in order to make it more visually appealing and to aid recognition), and basic recognition (producing interpretations of strokes). In the approximation stage the system first detected vertices by locating points in the stroke that were either speed minima or curvature maxima. After using an average based filtering technique, they then generated hybrid fits between candidate speed and curvature points and tested that fit using an orthogonal distance squared error metric. In the end they developed a system that had a 96% accuracy rate, but was limited to primitive shapes that consisted of curves and polylines. In Yu [13], similar work was accomplished. However, when detecting vertices Yu uses only curvature data to determine breaking points of the stroke which avoids having to use thresholds. To create more complex shapes from these low-level geometric recognizers, LADDER was created [3]. LADDER is a language that can be used to specify sketch symbols hierarchically as a combination of low-level primitives meeting certain constraints. However, a downfall of such a system is that many sketch symbols can be hard to define geometrically as seen in Figure 2.



Figure 2: A Kanji symbol which would be hard to describe geometrically.

While our work does not currently use geometric features to classify sketches, it does use the concept of curvature to help determine the similarity of sketches. Curvature has been shown by both Sezgin and Yu to be an important feature when recognizing single strokes. The algorithm described in this paper will use the average global curvature across all strokes as one way to describe the similarity between two sketches. Furthermore, a technique similar to that in [12] is used to detect perceived corners.

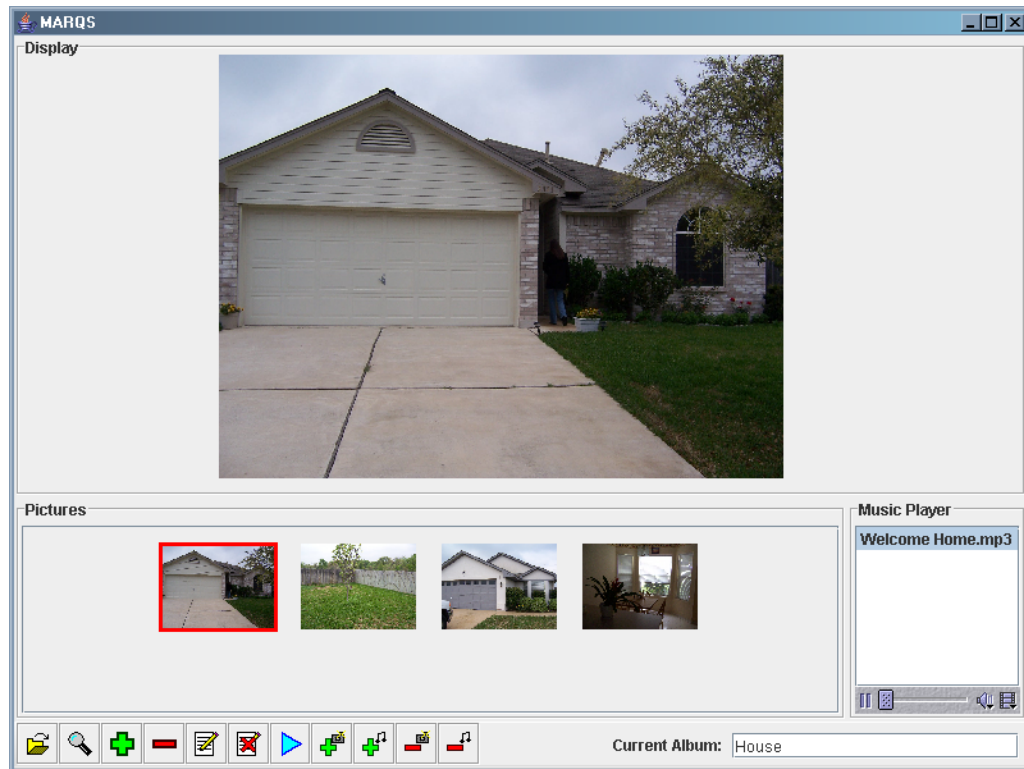


Figure 3: Screenshot of the MARQS application.

2.3. Media Retrieval through Sketch

Much work has already been done in the area of image retrieval and computer vision. One such example can be found in [14]. This work by Kato et al. describes a system that can be used to retrieve a full color image given a user sketch of that image. The system starts by creating a black and white abstraction of the full color images in the database. These abstractions are then used as search keys when the user enters a query sketch. In order to determine the similarity of a sketch and an image abstraction the authors divide them into 8 by 8 blocks and determine a local correlation of pixels between each block. These are then summed to obtain a global correlation which can be used to determine similarity. This idea of pixel or ink density is also described in [15]. A similar concept is used as a feature by our recognition algorithm. However, instead of dividing sketches into 8 by 8 blocks and finding local correlations, a simple global correlation is used in which the pixels contained in the bounding boxes of the sketches will be compared. This addresses a problem that can occur when using grids, which is when the user forgets exactly how they may have drawn an original sketch. For example, the user may draw a house with a chimney on the left side of the roof, but may later query with a house that has a chimney on the right side of the roof. If these images were broken into grids the local correlations may not match up well. When using a global correlation it becomes irrelevant which side of the house the chimney is drawn. In [16], similar problems may occur as the authors describe a sketch retrieval system that uses the spatial relationships between strokes as a means of comparing sketches. However, the authors of this paper describe the idea of using an unmatched cost to counteract a matching score. This idea would be an interesting addition for future work.

3. RECOGNITION ALGORITHM

The algorithm uses two different classifiers depending on the number of training examples currently available. Each time the user executes a search within a system that uses our algorithm, the query sketch can be added to the list of training examples for its particular sketch group. Both classifiers use the same feature set which is used to describe any given sketch. The feature set is based on global features of the sketch, allowing them to be drawn to any scale and orientation, in any order, and with multiple strokes. The features we chose for our algorithm allow users much freedom when sketching. We were careful not to choose features (such as speed, stroke length, number of strokes, etc.) that would constrain how the user could draw a particular symbol. The values for the features we chose should be relatively similar for two symbols even if they are drawn at different orientations and to different scales.

Before calculating a sketch's features, we first calculate the major axis of the sketch. We say the major axis is simply the line that connects the two points in the sketch which are furthest apart. Once finding this line, we rotate the points of the sketch so that the major axis lies horizontally. This operation ensures that all sketches will be oriented in a similar fashion when searching. Furthermore, since we do not use templating or a grid-based approach, our system is capable of handling local correlation discrepancies with ease. Because of this, we do not need to worry about rotations which may cause sketches to be rotated 180 degrees differently than its counterparts. Once point rotation is performed, we then calculate the feature set for the sketch. Currently, the system uses only four global features to describe a sketch.

1. Bounding box aspect ratio: the total width of the sketch divided by the total height of the sketch
2. Pixel density: the ratio of filled (black) pixels to total

pixels within the sketch's bounding box

3. Average Curvature: the sum of the curvature (as described in [12] and [13]) values of all the points in all strokes divided by the total sketch length (sum of the stroke lengths of all strokes in the sketch)
4. The number of perceived corners across all strokes of the sketch (using the segmentation technique presented in [12])

Whenever there exists only a single example of a particular sketch associated with any sketch group, a simple classifier ("Single Classifier") is run which calculates the features from above for the query sketch. These features are then compared to those of each sketch in the database. Errors are computed for each feature as the absolute difference between the value of the feature for the query and the value of the feature for the sketch in the database. This is then divided by the value of the feature for the query sketch in order to normalize the errors. These errors are then summed together to give a total error. Sketches are ranked and those with the lowest error are displayed as search results. Once the system has obtained at least 2 examples of a sketch for every album in the database, then a linear classifier ("Linear Classifier") is used. The linear classifier is implemented in much the same way as described in [7].

4. MARQS

In order to test our recognition algorithm in a real-world system, we implemented the **MARQS**, **M**edia **A**lbums **R**etrieved by **Q**uery **S**ketch system. Figure 3 shows a screenshot of the MARQS system. The system is a simple photo/music album application that contains buttons that allow operations like:

- Adding or deleting an album
- Adding or removing pictures/music from an album
- Playing a slideshow of the pictures in the album
- Opening an album
- Searching for an album (by sketch)
- Editing the album name and/or sketch associated with the album
- Removing training examples from the album's sketch group

When the search button is pressed, a small panel is brought up for the user to sketch on. Once the user executes the search they are then shown the top 4 sketches selected by the system. The user can then select the correct image (by clicking it) and the album is retrieved (see Figure 4). Other sketches, not in the top 4, can be displayed by clicking a Next button.

5. RESULTS

In order to gather data for preliminary testing of our recognition algorithm, 10 people were asked to draw 10 examples of a sketch symbol of their choice, with no constraint placed on how they should draw it. The first drawn sketch was designated the search key, while the other nine were used as query sketches to search for the initial sketch. In addition, the system was also populated with 10 examples of 5 sketch symbols drawn by one of the authors, giving a total of 150 sketch symbols. These examples were added to test sketches drawn to various scales and rotations since many users did not draw examples with a high degree of variance in these respects. The goal of the experiment was to prove not only that the system produced accurate recognition rates, but that recognition improved as more training examples were learned through querying. Examples of some sketches

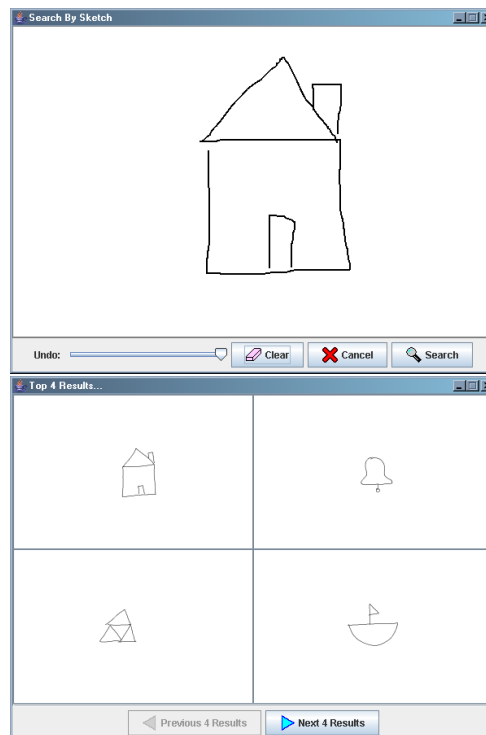


Figure 4: Search panel and the results that follow.

collected are shown in Figure 5. Variations between queries for the same sketch for one example can be seen in Figure 6.

As each query sketch was executed the rank of the expected sketch in the search results was recorded. Ideally, each sketch would rank 1st amongst the list of search results (search rank = 1.0) with the worst search rank = n , where n is the number of sketch symbols in the database. Once each query was executed, that query sketch was added to the example set for that sketch symbol and the system was retrained. Because the order in which sketches are added to the recognizer may play a role in recognition, we conducted 10 separate tests in which sketches are queried in random order. We then averaged the results across all 10 experiments. In the end, 1350 different sketch queries were performed (15 sketches, 9 queries each, 10 tests). On average, the system used the Single Classifier 27% of the time and used the Linear Classifier 73% of the time. The average search rank for the Single Classifier was 1.7, the Linear Classifier was 1.44, and the entire system yielded an average search rank of 1.51. Furthermore, out of the 1350 total queries that were executed we found that the system produced the top search result 70% of the time; 87.6% of the time the correct sketch was ranked the top 2; 95.5% of the time it was ranked in the top 3; and 98% of the time the correct sketch was ranked in the top 4. Only 2% of all total queries did not contain the correct result in the top 4 search results. Graphing the average search result rank for each query, it can be seen that as more examples are added to the training set the average search result rank improves over time (see Figure 7).

6. DISCUSSION/FUTURE WORK

The results from the experiment above demonstrate that the system in which sketch symbols are defined by global features is capable of producing accurate recognition results. Furthermore,

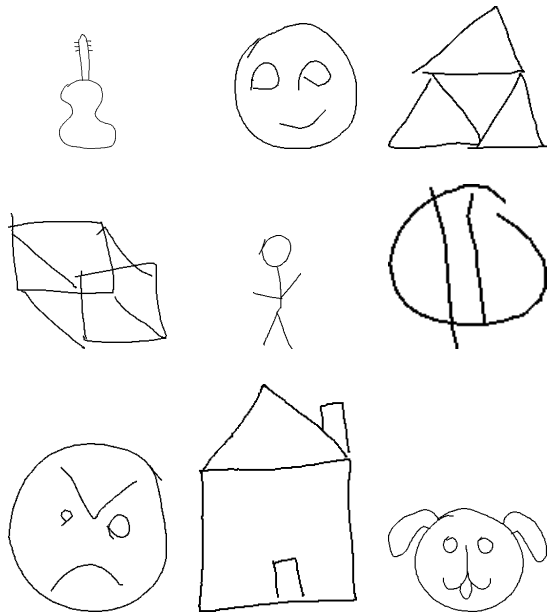


Figure 5: Examples of sketch symbols created by users.

recognition rates can be improved over time by allowing queries or error correction techniques to aid in informing the system of other examples of a sketch symbol. An advantage of our technique is that it allows users to draw sketches with any number of strokes, in any order, and with various degrees of scale and rotation. Some examples of sketches found which have varying scales and rotations can be seen in Figure 8. Furthermore, because of the features we have chosen, our recognition algorithm is also able to handle sketches that have local orientation differences, as seen in Figure 9.

For our system, we find this to be a benefit since users may not always remember the exact local orientation in which they drew a sketch; however, for some domains in which sketch symbols are well-defined and local orientation differences distinguish one sketch from another, this may not be desirable. In these cases, our ink density feature could be modified to use a grid-based approach much like that described in [14].

While our recognition algorithm has many advantages, in its current state, it does have a few notable downfalls. The first issue we discovered through the course of using the application was the slowdown (about 2-3 seconds) during query time and reduction in accuracy over time with the Single Classifier. While we believe that the use of two classifiers is a viable trade-off from having to initially train the system, we recognize that over time more and more symbols will be added to sketch domains. It therefore becomes imperative that we improve the downfalls of the single example classifier. One obvious improvement would be to implement techniques to generalize the feature values of a symbol across all examples of that symbol rather than comparing the query sketch to all examples of all sketch symbols in the database. This will likely save time, but may have an adverse effect on accuracy.

One way in which accuracy could be improved for the algorithm overall would be the addition of more features. Our current implementation indicates decent recognition results with only four features. However, previous feature-based recognizers of single strokes have contained upward of 13-22 features [6] and [7]. In the future, we wish to integrate some global features mentioned in previous works into our own algorithm. The

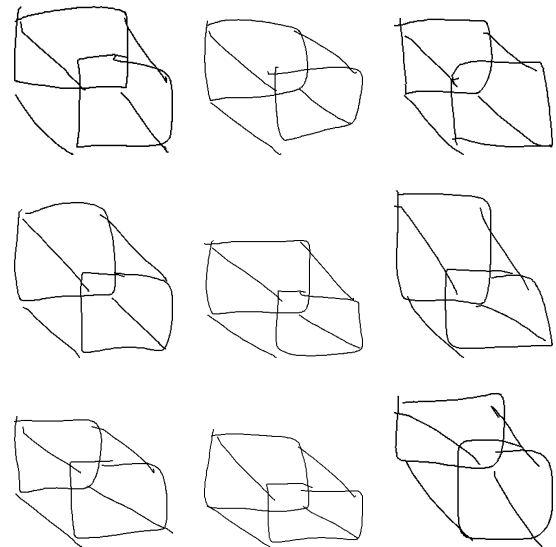


Figure 6: Variations in queries for the “cube” sketch. The original sketch can be seen in Figure 5.

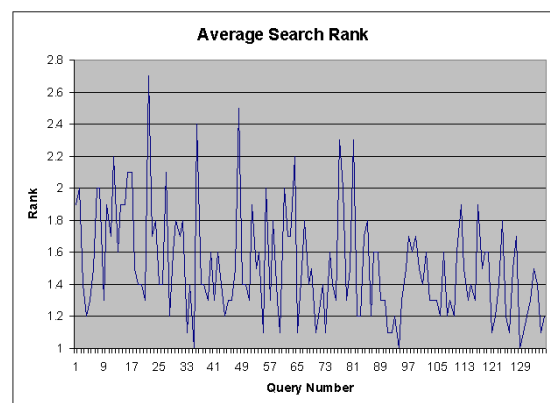


Figure 7: Average search result ranks across 135 queries. As more examples are learned from previous queries, the overall accuracy improves. While spikes still exist, notice that after about query 83 the average search rank no longer spikes above 2.0.

features selected in this paper were meant to allow symbols to be drawn in multiple strokes, as well as, be user and drawing-style dependent. Some features may violate these conditions. For example, we would not want to choose stroke length as a feature because a symbol should be capable of being drawn to any scale. Speed is another feature that we may wish to avoid because certain users may draw at different speeds. Discovery of other features that can be used to describe symbols within violating these conditions is another area of future work for us.

We would also like to examine scaling issues and the possibility of overfitting over time. Currently, we have yet to test to system with an extreme number of sketch symbols. It is plausible that over time, overfitting will occur, causing recognition rates to decrease. In the future we hope to test the system further to determine where overfitting is likely to occur. Examining this information could lead to the discovery of a cut-off point where query sketches are no longer needed to provide accurate results. These cut-off point discoveries could also be used to speed recognition as more sketches are added to the domain.

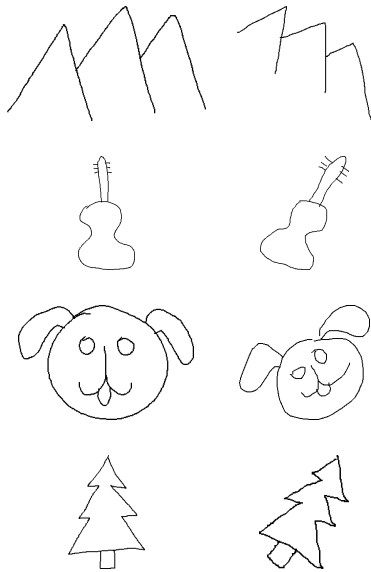


Figure 8: Examples of original sketches (left) and query sketches (right) which differ global orientations.

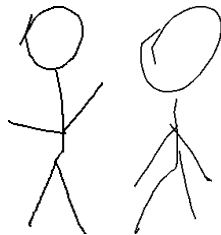


Figure 9: Example of sketches with similar global orientations but different local orientations.

Ideally though, we intend our recognition approach to be employed in domains that have a relatively small amount of symbols, such as in applications like: searching a personal notebook for sketches, creating basic diagrams (UML, circuit, mechanical engineering, etc.), document editing, etc. As an alternative to simply throwing out all query sketches at a cut-off point, we could use heuristics such as only adding query sketches that were "different" (ones that were not the top search result) once a cut-off point is reached.

Another area of future work is analyzing the usefulness and possible improvements that other classifiers could yield. Our current system combines a 1-nearest neighbor classifier with a linear classifier. We chose to use the linear classifier presented in [7] as it is one of the more prominent algorithms used in classifying single strokes in the sketch recognition field. In the near future, we wish to test other classification methods such as support vector machines, Bayesian classifiers, and other non-linear classification techniques.

Finally, work is still left to be accomplished in determining perceptual groupings of strokes. In our application, no grouping was needed because all of the strokes on the screen were combined to form a query symbol; however, in most sketch recognition interfaces this will not be the case, because many sketch symbols could be drawn to the screen at any given time. Some work on perceptual grouping has already been performed by Saund et al. [17]. This is a necessary next step that must be taken in order to integrate the ideas of this paper into traditional

sketch interfaces.

7. CONCLUSION

We have introduced a new dual-classifier recognition algorithm that uses four global features (bounding box ratio, pixel density, average curvature, and number of corners) to describe an entire sketch. Our approach allows sketches to be drawn with any number of strokes, in any order, and to any scale and rotation. The dual-classifier allows the recognition system to be initially trained with only a single example. Future queries of a system utilizing our approach can be used to add training examples to the classifier, thus improving recognition accuracy over time. To give an example of how our approach can be implemented into a sketch-based interface, we have created MARQS, media albums retrieved by query sketches. The system uses sketch symbols as a means of querying a database of existing media albums. The system has produced acceptable recognition rates, yielding an average search rank of 1.51. In addition, a search query was found to be in the top 4 search results 98% of the time.

8. ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of Paul Bogen III, Amanda Coots, Henry Choi, Katie Dahmen, Mark Eaton, Pankaj Rajan, Aditya Ramgopal, Vijay Singh, and Jennifer Weingarten for their helpful discussion and editing critiques throughout this entire process. Kanji image posted by an anonymous user on Wikipedia for free use.
<http://en.wikipedia.org/wiki/Image:%E6%9B%B8.svg>

9. REFERENCES

- [1] M. D. Gross and E. Y.-L. Do, "Ambiguous intentions: a paper-like interface for creative design", in *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, (New York, NY, USA), pp. 183–192, ACM Press, 1996. 1
- [2] M. D. Gross and E. Y. L. Do, "Demonstrating the electronic cocktail napkin: a paper-like interface for early design", in *CHI '96: Conference companion on Human factors in computing systems*, (New York, NY, USA), pp. 5–6, ACM Press, 1996. 1
- [3] T. Hammond and R. Davis, "Ladder, a sketching language for user interface developers", *Computers & Graphics*, vol. 29, no. 4, pp. 518–532, 2005. 1, 2
- [4] A. C. Long, J. A. Landay, and L. A. Rowe, "'those look similar!' issues in automating gesture design advice", in *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, (New York, NY, USA), pp. 1–5, ACM Press, 2001. 1, 2
- [5] P. Morrel-Samuels, "Clarifying the distinction between lexical and gestural commands", *Int. J. Man-Mach. Stud.*, vol. 32, no. 5, pp. 581–590, 1990. 1
- [6] J. A. Chris Long, J. A. Landay, L. A. Rowe, and J. Michiels, "Visual similarity of pen gestures", in *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 360–367, ACM Press, 2000. 1, 2, 5
- [7] D. Rubine, "Specifying gestures by example", in *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 329–337, ACM Press, 1991. 1, 2, 4, 5, 6

- [8] M. J. Fonseca, C. Pimentel, and J. A. Jorge, “Cali: An online scribble recognizer for calligraphic interfaces”, in *AAAI Spring Symposium on Sketch Understanding*, pp. 51–58, 2002. 1, 2
- [9] L. B. Kara and T. F. Stahovich, “An image-based trainable symbol recognizer for sketch-based interfaces”, in *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural*, pp. 99–105, 2004. 1, 2
- [10] Z. Sun, E. Jiang, and J. Sun, “Adaptive online multi-stroke sketch recognition based on hidden markov model”, *Lecture Notes in Artificial Intelligences*, vol. 3784, pp. 948–957, 2005. 1, 2
- [11] I. E. Sutherland, “Sketchpad a man-machine graphical communication system”, in *25 years of DAC: Papers on Twenty-five years of electronic design automation*, (New York, NY, USA), pp. 507–524, ACM Press, 1988. 2
- [12] T. M. Sezgin, T. Stahovich, and R. Davis, “Sketch based interfaces: early processing for sketch understanding”, in *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, (New York, NY, USA), pp. 1–8, ACM Press, 2001. 2, 4
- [13] B. Yu and S. Cai, “A domain-independent system for sketch recognition”, in *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, (New York, NY, USA), pp. 141–146, ACM Press, 2003. 2, 4
- [14] T. Kato, T. Kurita, N. Otsu, and K. Hirata, “A sketch retrieval method for full color image databases - query by visual example”, in *11th IAPA International Conference on Pattern Recognition*, pp. 530–533, 1992. 3, 5
- [15] L. M. Gennari, L. B. Kara, and T. F. Stahovich, “Combining geometry and domain knowledge to interpret hand-drawn diagrams”, in *AAAI Fall Symposium Series 2004: Making Pen-Based Interaction Intelligent and Natural*, pp. 547–562, 2004. 3
- [16] W. H. Leung and T. Chen, “Retrieval of sketches based on spatial relation between strokes”, in *Proceedings of the 2002 International Conference on Image Processing*, pp. I-908–I-911, 2002. 3
- [17] E. Saund, D. Fleet, D. Larner, and J. Mahoney, “Perceptually-supported image editing of text and graphics”, in *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, (New York, NY, USA), pp. 183–192, ACM Press, 2003. 6

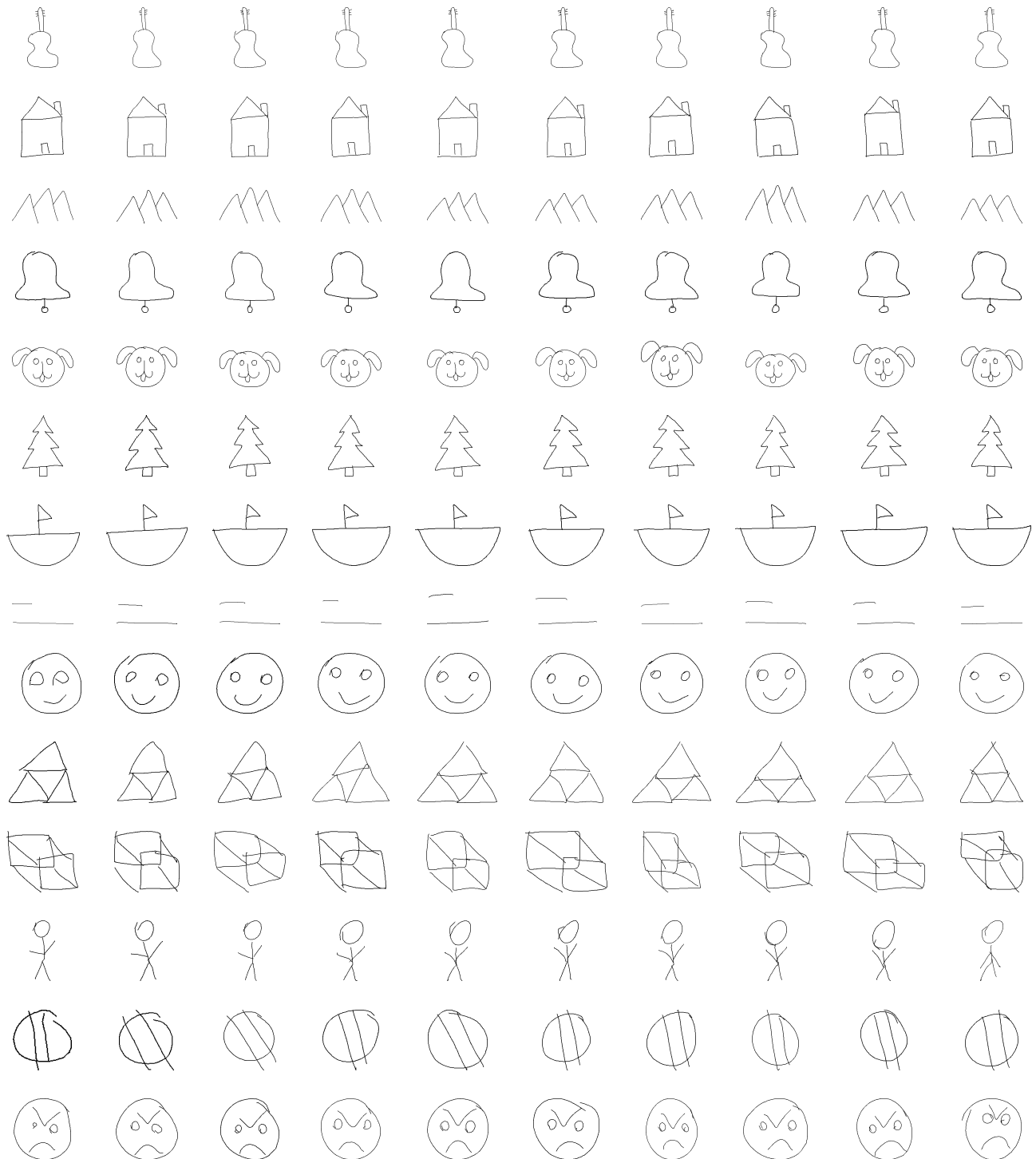


Figure 10: Various examples of shapes we collected during our study.