Using Entropy to Distinguish Shape Versus Text in Hand-Drawn Diagrams

Akshay Bhat, Tracy Hammond
Sketch Recognition Lab
Department of Computer Science and Engineering
Texas A&M University
{akb2810, hammond}@cs.tamu.edu

Abstract

Most sketch recognition systems are accurate in recognizing either text or shape (graphic) ink strokes, but not both. Distinguishing between shape and text strokes is, therefore, a critical task in recognizing hand-drawn digital ink diagrams that contain text labels and annotations. We have found the 'entropy rate' to be an accurate criterion of classification. We found that the entropy rate is significantly higher for text strokes compared to shape strokes and can serve as a distinguishing factor between the two. Using a single feature – zero-order entropy rate – our system produced a correct classification rate of 92.06% on test data belonging to diagrammatic domain for which the threshold was trained on. It also performed favorably on an unseen domain for which no training examples were supplied.

1 Introduction

Sketch recognition involves recognizing free-hand sketches, which are usually drawn on a tablet PC or some other pen-based input device. Users are allowed to draw as they would naturally, without constraining their drawing to a prescribed manner, as in Rubine [1991]. As pen-based input devices are becoming increasingly common, many researchers are working on developing sketch recognition systems for different domains, such as mechanical engineering drawing [Alvarado, 2000; Stahovich, 1996; Landay and Myers, 1995], circuit diagrams [Alvarado and Davis, 2004], UML class diagrams [Hammond and Davis, 2002; Damm *et al.*, 2000], GUI design [Caetano *et al.*, 2002], course of action diagrams [Pittman *et al.*, 1996] etc.

In most systems, recognition is typically achieved through several processing steps, ranging from low level pixel manipulation and interpretation to high level semantic understanding. The existing systems do a reasonable job of identifying geometrical shapes [Gross and Do, 1996; Fonseca *et al*., 2002; Hammond and Davis, 2003; Alvarado and Davis, 2004; Gennari *et al*., 2005] or in identifying text [Xu *et al.*, 1992; Anquetil and Lorette, 1995; Bellagarda *et al.*, 1994;]; but, these systems have difficulty in recognizing diagrams that contain both. Most diagrams, however, such as those in mechanical engineering, UML diagrams, military course of action diagrams, etc., contain both shape and text strokes intermingled. For example, a mechanical engineering drawing may have graphical content in the form of shapes specifying machine parts and textual content specified by dimensions written in numeric characters, names of the different machine parts, or some other annotation. Therefore, one important task in sketch recognition is to separate shape and text strokes, and then feed them into the appropriate shape or text recognizer. We shall label the task of separation of shape and text strokes as: *shape vs. text*.

2 Prior Work and Analysis

Most sketch recognition systems deal solely with the recognition of either text or shape. Rubine [1991] trained a linear classifier on a set of 13 structural and temporal features. The classifier recognizes alphanumeric symbols and gestures with very high accuracy (alphabets with 97.1%, digits with 98.5% and gestures with 100% accuracy). However, each symbol has to be drawn in a prescribed order, and the classifier does not allow for freeform input. Tahuti [Hammond and Davis, 2002] recognizes freeform text within UML class diagrams, but identifies text solely by size and context within the diagram. [Hammond and Davis, 2003; Alvarado and Davis, 2004] use a geometry-based recognizer, which first splits the shapes into primitives and then recognizes them on the basis of a grammar. This requires the shapes to be geometrically decomposable, which proves difficult for freehand text (although the system has been successful at recognizing more diagrammatic text, such as Asian characters). Corey and Hammond [2008] use a combination classifier based on Rubine [1991] features and a geometrical recognizer [Hammond and Davis 2003] to recognize a greater class of shapes and text. However, the accuracy of their system is still lower than what is expected from an individual recognizer dedicated to either text or shape.

Patel *et al.* [2007] were the first to develop a context-free algorithm to distinguish shape and text. They developed a shape versus text algorithm, using a statistical approach to identify the most important features of ink that could be

used to distinguish text versus shapes, and they built a decision-tree based classifier from these features. Their algorithm greatly improves upon the Microsoft text recognizer; however, the rate of misclassification was still high (42.1% for shapes and 21.4% for text). Bishop *et al.* [2004] built an HMM that uses a combination of features from the current stroke, in concert with spatial, timing, and contextual recognition information, to distinguish shape versus text. Jain *et al.* [2001] use a hierarchical stroke clustering approach for segmenting a document page into text and non-text regions. This method works with high accuracy; however, their system would not be able to identify textual labels which are interspersed within a diagram.

3 Our Approach

The dominant approach in solving the problem of shape vs. text has, until now, been to arbitrarily select certain features through observation and use these in some form of classifier. In some cases, filtering is done based on a statistical approach, which removes features that do not vary significantly between shape and text [Patel *et al.*, 2007]. Nevertheless, the initial set of features is selected arbitrarily from observation. The questions faced here are: Why should a particular feature be selected? When should addition of more features stop? Trial and error is one such approach: arbitrarily select features, build a system using them, test the system, and then remove the statistically insignificant features; loop through this process until we see no further improvement in system accuracy. This process, however, could be time consuming.

In our approach, we set out first to find the single logically coherent feature which distinguishes shape from text. We observed that, when using any general set of coordinate equations, handwritten text symbols are more difficult to describe than common shapes (which are geometrically simpler). In that sense, text strokes are more randomly structured. Thus, the entropy measure (generally, a measure of the degree of randomness of an information source) of text strokes is higher than shape strokes. In other words, text strokes are more information dense than shape strokes.

In this paper, we define the information theoretic concept of entropy in the context of digital ink. We have defined an alphabet to represent strokes as strings of the letters from this alphabet. This entropy serves as a single encompassing feature, embodying the structural characteristics of shape and text strokes, and can be used to produce a classification between them.

3.1 Entropy

In the context of information theory, Entropy is defined as the measure of uncertainty associated with a random variable. The term usually refers to the Shannon entropy, which quantifies - in the sense of an expected value - the information contained in a message [Shannon, 1948]. To define entropy for hand drawn digital ink strokes, we modeled the process of drawing strokes in the form of a stochastic process. We first defined an entropy model 'alphabet' to characterize the language of hand-drawn sketches. Each point in a stroke is assigned a symbol based on the angle it makes with neighboring points [Figure 1]. This symbol is the random variable on the basis of which we can calculate entropy.

We use a zero-order entropy in this paper, in which each symbol's probability of occurrence is determined independent of the previous symbols. We also attempt to see the effect of using higher order entropy (in the form of GZIP entropy) for classification. Detailed investigation into this, however, is left for future work.

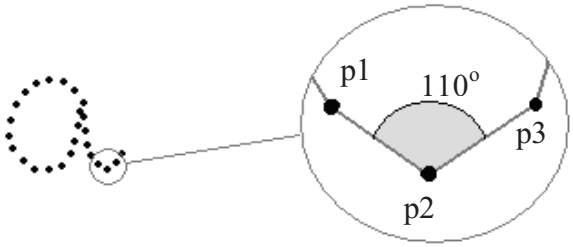

Figure 1: Each point is assigned a symbol from our alphabet based on its angle with the temporally adjoining points. Here, point p2 makes an angle of 110 degrees with p1 and p3. Looking in Table 1, we find the angle lies in the range for which the assigned symbol is 'D'.

3.2 Entropy Model 'Alphabet'

In order to calculate the entropy of a group of strokes, we created an entropy model 'alphabet' containing 7 symbols. Each symbol corresponds to a range of angles [Table 1]. Any stroke can be described (to some degree of resolution) by strings of symbols from an alphabet of this type [Figure 2]. If we had chosen a range $[0, 2\pi)$, we could fully describe any ink stroke using the alphabet. However, in this paper we were more concerned about the curvature of angles, for which a range of $[0, \pi)$ was sufficient. In Figure 2, we observe that the representation of freehand text is more varied, in terms of the variety of symbols it contains, than the representation for the rectangle. This gives us an idea of greater randomness in text strokes compared to shapes.

SYMBOL	RANGE
A	$[0, \pi/6)$
B	$[\pi/6,2\pi/6)$
C	$[2\pi/6,3\pi/6)$
D	$[3\pi/6,4\pi/6)$
E	$[4\pi/6,5\pi/6)$
F	$[5\pi/6,\pi)$
X	End points

Table 1: The table shows symbols from the alphabet and the range of angles which they correspond to.

XDFFFFFEEFFFFFFFFFFFFFXXFFFFFFEDFFFFFAFF
EEFFXXFFDEFFFEEFFXXFFFFFXXFFFXXEFFFEEFFF
FXXFFFFFFEDFFFFFFFFX

XFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFFFFFEFFF
FFFFFFFFFFFFFDFFFFFFFFFFFFFFFFFFFEX

Figure 2: A representation of ink strokes in our alphabet. Each character in the string representation of the strokes matches the letter assigned to each ink point based on the angle it makes with the temporally adjoining points. Notice that the representation of the text stroke is more varied in terms of the different letters it contains than the representation of the shape stroke below it (the rectangle).

4 Implementation

4.1 Stroke Grouping

In our implementation, we consider a diagram as a combination of stroke groups. Each stroke group consists of strokes which were intended by the user to belong to the same entity. Our intuition is that such strokes will be drawn in quick succession and will be spatially close to each other. We group together all strokes which are below a spatial and temporal threshold. The threshold for the temporal difference is kept at 100 milliseconds, or 400 milliseconds when the strokes overlap. These thresholds were designed to constrain the grouping algorithm to group only strokes which are a part of the same text letter, and to those stroke letters that correspond to a single word of text. It should be noted that with this simplistic grouping algorithm we have deliberately used a threshold which under-groups the strokes. We do this because, while we can still recognize incomplete text grouping correctly, we cannot recover from incorrect grouping; we did not want a slack threshold to result in an erroneous grouping of shape and text strokes together. Our grouping algorithm correctly placed text and shape strokes in separate groups 99.78% of the time.

Once the grouping is done, we first resample the stroke to smooth the substroke angles and so that the points are equidistant from each other (we choose 4 pixels). We then transform each stroke into a string of symbols from our alphabet, assigning an entropy model alphabet symbol [Figure 1] to each resampled point [Figure 2]. We calculate the probability estimate for a particular symbol by dividing the number of times that symbol occurs in the stroke group by the total number of symbols in that stroke group. For example, in Figure 2, the probability estimate for occurrence of symbol X is 14/100 in the freehand sketch and 2/75 in the rectangle shape. Then, within each stroke group, we sum up the probabilities of the symbols assigned to the points according to the formula given below:

$H(S) = -k \Sigma_S P(x_i) \log P(x_i)$ [Shannon, 1948],

where $P(x_i)$ is the probability of label assignment of $LABEL(x_i)$ to point x_i in the input stroke S, and k is a constant. For example, after multiplying with a suitable factor, the entropy value for the freehand text in Figure 2 was 10.32, and for the rectangle, the value was 1.23.

4.2 Classification

The resulting value, when averaged over the bounding box diagonal (to ensure our values are scale independent), gives us an estimate of the entropy rate of that stroke group [Figures 3 and 4]. Based on a threshold calculated from our training dataset, we classify the input stroke group as either shape, text, or unclassified. Strokes are labeled as 'unclassified' when their entropy rate value lies below the boundary threshold for text, but above the threshold for shapes. Leaving ambiguous strokes unclassified allows us to have high confidence on those strokes that we have classified.

4.3 Confidence Measure

In order to produce a classifier which can be easily integrated into other sketch recognition systems, we realized that a measure of confidence was necessary, which could reflect the authority with which each classification decision was made. Strokes having entropy value lying close to the threshold between shape and text can be classified with less confidence than the strokes that have entropy values farther away from the threshold. The confidence is thus modeled appropriately (as shown in our results section) with the following arctan function:

$C(x \mid TEXT) = 0.5 + (\arctan(x-b)) / \pi$

$C(x \mid SHAPE) = 1 - C(x \mid TEXT)$

$C(x \mid TEXT)$ refers to the confidence with which a stroke with entropy x can be classified as text. b is a parameter set to mirror the actual distribution of the entropy in the training data and represents the entropy value for which confidence of classification decision of text is 0.5.

4.4 Data collection and Testing

The system was tested on data collected through SOUSA [Paulson, *et al.*, 2008]; SOUSA allows data collection by users over the web, provided they have a drawing tablet. Data from two different domains: hand-drawn military course of action (COA) [Wikipedia] symbols [Figure 5] and free body mechanics diagrams [Figure 6] were used. Both these domains consist of diagrams containing both shape and text strokes. The COA data was collected from 6 users. Each user was asked to draw 2 diagrams each of 16 COA symbols. We collected a total of 162 diagrams (since the users perform the study remotely, some users did not finish the entire user study). From the total of 162 diagrams, we removed the diagrams that were drawn with a mouse instead of a pen (since users specified this in SOUSA) and gathered a final set of 130 diagrams. In each diagram we manually

labeled the strokes as 'shape' or 'text' using a labeling tool. The dataset had a total of 756 strokes, out of which 225 were shape strokes and 531 were text strokes. The system was trained and tested on the data from the COA domain using a ten-fold cross validation technique. The data was divided into 10 groups of 13 diagrams each. Within each iteration, data was trained on 9 groups and tested on the remaining group. This process was repeated until all the diagrams were tested exactly once. The final accuracy of classification was calculated by averaging the accuracy values obtained in all the iterations.

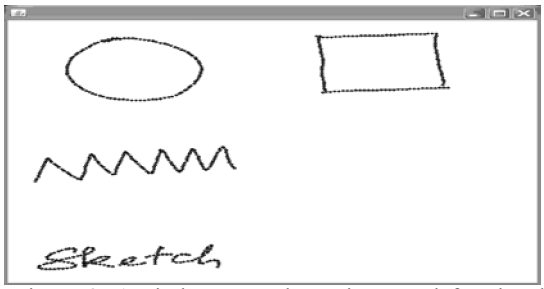

Figure 3: A circle, rectangle, resistor, and free hand text drawn on the draw panel of our system.

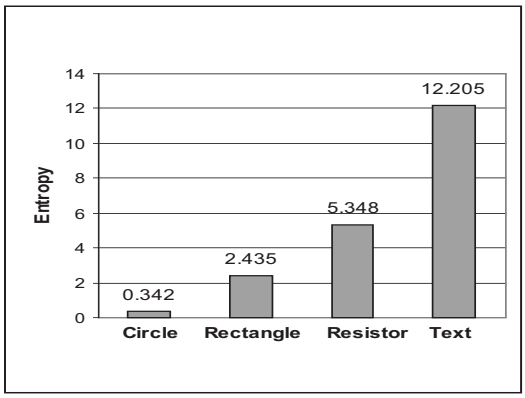

Figure 4: A bar graph showing the entropy rate values of the shapes shown in Figure 3. Notice that even the resistor symbol, which looks visually similar to freehand text, has entropy significantly less than the text stroke, due to less variety (randomness) in the curvature of the angles.

The free body diagrams were collected from 7 users, each of whom was asked to draw one diagram. The dataset consisted of a total of 197 strokes, out of which 70 were shape strokes and 127 were text strokes. We deliberately did not train our system on this data. For the purpose of testing our system on this data, we used the threshold values obtained on training the system on COA data. This allowed us to measure the performance of our system (and our trained thresholds) on diagrams from unseen/untrained domains.

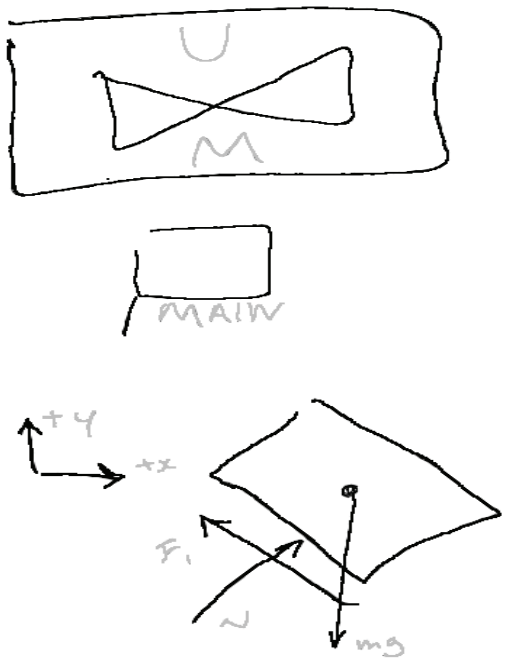

Figure 5: Military course of action symbols containing handwriting above and a freebody mechanics diagram below. The strokes in light gray color were classified as text and the strokes in black were classified as shapes.

5 Results

We first trained (using 10-fold cross validation) the system to produce maximum accuracy, allowing it to leave a maximum of only 25% of the strokes unclassified. The system, on an average, produced a classification 77.51% of the time, and it was correct 95.56% of the time. Our system was more accurate with text strokes than with shapes. In first part of our experiment (in which the classification percentage was >=75%) text strokes were classified correctly 97.62% of the time, whereas the accuracy for shape strokes was 91.18%.

In the second part of the experiment, we trained the system to produce more classifications. We allowed the system to be non-committal for not more than 10% percent of the strokes. In this case, we found different optimum threshold values for different iterations of the tests. The system, on average, produced a classification 91.53% of the time, and it was accurate 92.91% of the time. When we forced the system to always produce a classification, the accuracy was 92.06%. We also produced a confidence value with each classification. The function that was used based on our training data was:

$$C(x \mid TEXT) = 0.5 + (\arctan(x-3.55)) / \pi$$

The value b=3.55 was obtained by plotting the observed confidence of classification vs. entropy, then interpolating the curve to find the entropy value for which a classification of text could be produced with a confidence of 0.5 [Figure 6]. Simply put, we found the entropy value for which, if

each stroke in the training set having this entropy value were classified as text, the resulting accuracy would be 50%.

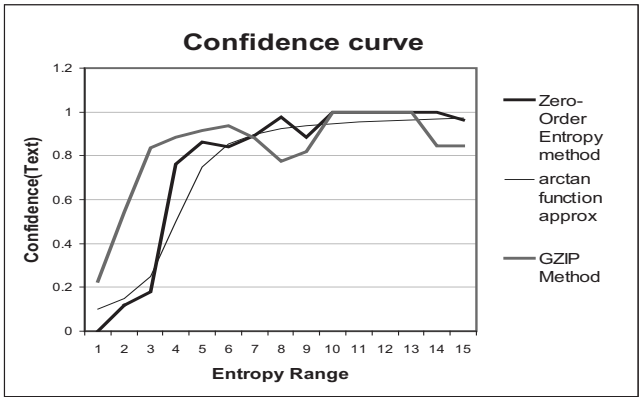

Figure 6: The graph shows the curve obtained by plotting confidence value for classification of *TEXT* for the strokes in the training data against their entropy values. On the zero-order entropy curve (dark bold line), the entropy value 3.55 corresponds to a confidence value of 0.5; in other words, equal numbers of text and shape strokes had an entropy = 3.55 in the training data. The bold gray line shows the same plot when GZIP entropy was used. GZIP entropy values have been scaled for comparison.

In the third part of the experiment we tested our system on the freebody diagrams dataset using the thresholds trained on the COA diagrams data. Although the classification rate was low (71.06%), the overall accuracy for the strokes classified was still high at 96.42%, with 99.21% of text strokes and 69.23% of shape strokes being classified correctly. Note that most of the unclassified strokes came from strokes in dotted lines, which were grouped together because of low temporal and spatial separation and caused them to have an ambiguous entropy rate.

We also tested the idea of using higher-order entropy rate. We classified strokes from COA dataset on the basis of GZIP entropy (measured by the bits required to represent each symbol in a stroke). The accuracy of classification was 82.8% which was lower than the accuracy achieved with zero-order entropy. We found GZIP entropy to be high for short strokes (less data) due to GZIP file overhead. Therefore, short line segments had higher GZIP entropy than with our method and were sometimes misclassified as text.

6 Discussion

When we required our system to always produce a classification, the accuracy was still high at 92.06%. Only 5.76% of the text strokes and 11.97% of the shape strokes were missclassified. This is an improvement over Patel *et al*. [2007], which had a misclassification rate of 42.1% for shapes and 21.4% for text. We implemented their decision tree-based classifier and found it to have an accuracy of 78.8% on our dataset, which is lower than the accuracy of our system, while requiring eight features for classification as opposed to our one. However, we should note that Patel's dataset included and was trained on filled-in music notes, a shape certainly to be misrecognized using our method. Our system was accurate when tested on untrained domains using the entropy thresholds trained from another domain. These results imply that entropy thresholds do not vary much for different domains and that a system using this method need not be retrained for each domain.

While the higher-order GZIP entropy model did not perform as well as our zero-order entropy model, we expect that subtler higher-order entropy-based systems will be able to identify repeating patterns, characteristic of certain shapes, such as the unclassified dashed lines in the freebody diagrams. For example, a symbol for a resistor consists of repeating patterns and, thus, is structurally less random. However, zero-order entropy will not capture these patterns, and will produce a relatively high value for the resistor, even though it is a 'shape'. This can result in misclassification. For example, in some of the test diagrams, our system incorrectly classified a 'dot' [Figure 7] as text even though the filled-out dot consisted of repeated circular strokes. (Note that our entropy algorithm did not misclassify the dot in Figure 5 because our grouping algorithm grouped the arrow and the dot.) In Figures 3 and 4, our system calculated the zero-order entropy value of a resistor symbol and found it to be near the boundary threshold between shape and text. This type of a stroke risks misclassification. Such a misclassification, however, could, be mitigated by associated low confidence value of classification. The confidence associated with classifying the resistor symbol as text was 0.83, whereas, the confidence associated with classifying the freehand text as text was 0.96.

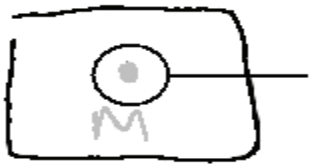

Figure 7: A misclassified example from COA dataset.

7 Future Work

Future work includes further investigation of higher-order entropy models as well as improved grouping models that can handle dashed lines. Additionally, we would like to test the affect on accuracy when combining entropy-based features in combination with other known features and additional context. Finally, we are interested in seeing how our model performs on larger data sets and other domains.

8 Conclusion

In this paper we presented a single feature to distinguish between shape and text strokes in the form of zero-order entropy. We found that the entropy rate tends to differ markedly in shape and text strokes and may be used as an important criterion of classification between them. We also

found that thresholds trained on one domain produce reasonable accuracy on another untrained domain. From this, we conclude that the entropy rate can serve as a domain-independent distinguishing factor between shape and text.

9 Acknowledgements

This work is supported in part by NSF grant 0757557.

References

[Alvarado, 2000] C. Alvarado. A natural sketching environment: bringing the computer into early stages of mechanical design. Master's thesis, MIT, 2000.

[Alvarado and Davis, 2004] C. Alvarado and R. Davis. SketchREAD: a multi-domain sketch recognition engine. *In UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, ACM Press, New York, NY, USA, 23–32.

[Anquetil and Lorette, 1995] E. Anquetil and G. Lorette. On-Line Cursive Handwritten Character Recognition Using Hidden Markov Models, *Traitement du Signal,* vol. 12, no. 6, pp. 575-583, 1995.

[Bellagarda *et al.*, 1994] E.J. Bellagarda, J.R. Bellagarda, D. Nahamoo, and K.S. Nathan. A Fast Statistical Mixture Algorithm for On-Line Handwriting Recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 12, pp. 1,227-1,233, Dec. 1994.

[Bishop *et al.*, 2004] C.M. Bishop, M. Svensen, G.E. Hinton. Distinguishing text from graphics in on-line handwritten ink. *In: Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop.* Volume, Issue , 26-29 Oct. 2004 p: 142 – 147.

[Caetano *et al.*, 2002] A. Caetano, N. Goulart, M. Fonseca, J. Jorge. JavaSketchIt: Issues in sketching the look of user interfaces, Sketch Understanding. *Papers from the 2002 AAAI Spring Symposium*.

[Corey and Hammond, 2008] P. Corey and T. Hammond. GLADDER: Combining Gesture and Geometric Sketch Recognition. *In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, Chicago, Illinois, USA, July 13-17, 2008.

[Damm *et al.*, 2000] C.H. Damm, K.M. Hansen, M. Thomsen. Tool support for cooperative object-oriented design: gesture based modeling on an electronic whiteboard. *In CHI 2000*, CHI; 2000. p. 518–25.

[Fonseca *et al.*, 2002] M.J. Fonseca, C. Pimentel, J.A. Jorge. CALI: An Online Recognizer for Calligraphic Interfaces. *Proc. of AAAI 2002 Spring Symposium: Sketch Understanding Workshop*.

[Gennari *et al.*, 2005] L. Gennari, L.B. Kara, T.F. Stahovich, K. Shimada. Combining geometry and domain knowledge to interpret hand-drawn diagrams, (2005) *Computers and Graphics (Pergamon),* 29 (4), pp. 547-562.

[Gross and Do, 1996] M.D. Gross and E.Y.L. Do. Ambiguous intentions: a paper-like interface for creative design, *Proceedings of the 9th annual ACM symposium on User interface software and technology*, p.183-192, November 06-08, 1996, Seattle, Washington, United States

[Hammond and Davis, 2002] T. Hammond and R. Davis. Tahuti: A geometrical sketch recognition system for UML class diagrams. *Papers from the 2002 AAAI Spring Symposium on Sketch Understanding* (March 25-27), 59–68.

[Hammond and Davis, 2003] T. Hammond and R. Davis. LADDER: a language to describe drawing, display, and editing in sketch recognition. *Proceedings of the 2003 International Joint Conference on Artificial Intelligence (IJCAI).*

[Jain, *et al.*, 2001] K. Jain, A.M. Namboodiri, J. Subrahmonia. Structure in on-line documents. *In ICDAR-6*, pages 844–848. IEEE, 2001.

[Landay and Myers, 1995] J.A. Landay and B.A. Myers. Interactive sketching for the early stages of user interface design. *In: Proceedings of CHI '95: Human Factors in Computing Systems*, 1995. p. 43–50.

[Patel *et al.*, 2007] R. Patel, B. Plimmer, J. Grundy, R. Ihaka. Ink features for diagram recognition. a language to describe drawing, display, and editing in sketch recognition. *In SIGGRAPH'07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling,* 2007. p. 131-138.

[Paulson, *et al.*, 2008] B. Paulson, A. Wolin, J. Johnston, T. Hammond. SOUSA: Sketch-based Online User Study Applet. *In EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling* (2008).

[Pittman *et al.*, 1996] J. Pittman, I. Smith, P. Cohen, S. Oviatt, T. Yang. Quickset: a multimodal interface for military simulations. *In: Proceedings of the Sixth Conference on Computer-Generated Forces and Behavioral Representation*, 1996. p. 217–24.

[Rubine, 1991] D. Rubine. Specifying gestures by example. *In SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 329–337.

[Shannon, 1948] C.E. Shannon. A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27 (July and October 1948), pp. 379-423 and 623-656. Reprinted in D. Slepian, editor, *Key Papers in the Development of Information Theory*, 974. Included in Part A.

[Stahovich, 1996] T.F. Stahovich. SketchIt: a sketch interpretation tool for conceptual mechanism design. Technical Report, MIT AI Laboratory, 1996.

[Wikipedia] APP-6A, Military Symbols for Land Based Systems. http://en.wikipedia.org/wiki/APP-6a

[Xu *et al.*, 1992] L. Xu, A. Krzyzak, C.Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition; *Systems, Man and Cybernetics, IEEE Transactions on Volume 22*, Issue 3, May-June 1992 Page(s):418 - 435