Ink Features for Diagram Recognition

Rachel Patel[1], Beryl Plimmer[1], John Grundy[1, 2], Ross Ihaka[3]

[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
[3]Department of Statistics
University of Auckland
Private Bag 92019, Auckland, New Zealand

Abstract

The ability to automatically recognize a sketch accurately is important to computer-based diagramming. Many recognition techniques have been proposed but few researchers have reported the use of formal methods to select the most appropriate ink features for recognition algorithms. We have used a statistical approach to identify the most important distinguishing features of ink for dividing text and shapes. We implemented these into an existing recognition engine and conducted a comparative evaluation. Our feature set more successfully classified a range of common diagram elements than two existing dividers.

Categories and subject descriptors: I.4.7 [Image Processing and Computer Vision]: Feature Measurement - *feature representation*

1. Introduction

Computer-based sketch tools, particularly diagramming tools, show promise as an alternative to paper for capturing early-phase designs. They retain the advantages of paper - such as an unconstrained drawing space that allows ambiguity and quick construction. They also have the key advantages of computer tools - such as digital storage, transmission and archiving. However, despite suitable hardware being available for some time, diagramming sketch tools are yet to achieve general acceptance. One of the outstanding challenges is the need for far more accurate recognition.

Recognition of sketches is an important aspect of computer-based diagramming: it allows the software to support tasks such as intelligent editing, execution, conversion and animation of the sketches. The syntactic and semantic elements of a diagram are both important. Only with accurate recognition will the full potential of computerization of sketches be realized.

There are three main approaches to the recognition: bottom-up, top-down or a combination of both. Bottom-up attempts to recognize individual ink segments and then progressively join these into larger and more complex groups thus developing an overall semantic understanding of the diagram. Top-down starts with a high-level analysis of the structure and uses this information to aid recognition of the composite parts. Combinations work both the primitives and layout to try to resolve ambiguities.

Regardless of the approach, sketch recognition is comprised of capturing *ink features* and *algorithms* to combine these features. The features measure aspects of an ink stroke's curvature, size, time, intersections and use similar aspects to detect relationships between strokes. To date, a wide range of algorithms have been used to recognise hand drawn shapes and text. However, little research has so far been done into the relative effectiveness of different approaches and use of different ink features in recognition. Furthermore, considering that stroke features are such an important part of recognition, there is little evidence of the use of formal methods to identify the most significant ink features to use. Most reports suggest a reliance on ad-hoc heuristics and empirical trial and error.

Most previous work in sketched diagram recognition concentrates on basic shape and gesture recognition. There are few diagramming tools that allow for both shape and text recognition together. Many of the tools that do support both are either modal interfaces or limited in functionality. However, the flexibility of sketch tools must be increased if they are to equal the performance of designing with pen and paper [Bla90; Goe95; BK03; PA03]. Regardless of whether a top-down or bottom-up approach is taken, text and shapes are semantically different and need to be treated separately during the recognition process. Handwritten characters need to be clustered into words and phrases in preparation for character recognition while shape combinations need to be identified as components and the relationships between the components explored.

In the following section we review sketch recognition techniques currently being used. Following this we describe an experiment that we conducted to find distinguishing ink features of text and shape strokes for a text/shape divider. We discuss the use of these features to implement an improved divider algorithm and compare its performance to two existing algorithms. We then discuss the findings of our experiment in the wider context of diagram recognition and conclude with areas for further research.

2. Background

Most sketch diagramming tools include some recognition [Lan95; GD96; DHT00; HL00; BKC01; FPJ02; CGH03; LL03; PA03; PA03a; PA03b; CFK*04; PA04]. Most diagram recognition engines have adopted a bottom-up approach. Bottom-up attempts to recognize individual ink segments or strokes and then progressively join these into larger and more complex groups thus developing an overall understanding of the diagram. Top-down starts with a high-level analysis of the structure and uses this information to aid recognition of the composite parts. Only a few current diagramming tools provide integrated writing and drawing recognition [HD02; CGH03; Lan03; LVZ04; Pli04; CMP05; You05; PTY06; FP07].

Bottom-up approaches start with stroke or sub-stroke recognition. Rubine's work [Rub91] in the area of gesture recognition has been used by numerous sketch recognition systems. He proposed the use of Hidden Markov Models (HMM) for single stroke ink recognition. He reported a set of 13 stroke features selected by empirical analysis and heuristics. While Rubine reported a 96.8% success rate, our experiments that re-implement Rubine's features have been lower 86% [Pli04] and 84% [You05]. However his algorithm has been widely adopted [LM95; LM96; DHT00; LNH*00; CGH03; PA03b; Pli04; CMP05; You05; PTY06; FP07] with various alterations to the feature set reported. The HMMs work best on a single stroke, techniques have been developed to join strokes [You05] and to split strokes [HSN04]. Nakai & Sudo et al [NSS*02] also use HMM and include pressure information as features to assist in recognition of Japanese characters.

Template matching is an alternative approach [Gro94; SDS95; Sta97; CSK*02; AD04; KS04]. A first approximation phase uses various stroke features to fit lines and curves to the sketch. These features are typically determined by empirical observation. These primitives are matched to shape templates. Often with this approach the sketch is transformed (beautified) into a formal representation as a part of the recognition process.

Once the primitives are identified, a range of different algorithms are used to combine these primitives into basic components (a node containing text for example). These algorithms including: Semantic networks [CSK*02], Bayesian networks [AD04], fuzzy logic [FPJ02; Qin05], and rules based on spatial and temporal relationships [HD03; You05]. Fonseca et al [JF99; FJ00; FJ01; FPJ02] report using percentile graphics for each possible feature which show the statistical distribution of feature values for different shape classes. This is one of the few ink feature sets that is scientifically-based.

The top-down approach first identifies high-level document structure. Kara and Stahovich [KS04] examine network diagrams consisting of connectors and nodes first identifying the connectors and then remaining ink is clustered into nodes for recognition and matched using templates. In a related field of hand-written notes including diagrams [SWR03] the recognizer starts with the assumption that all strokes are text and joins strokes into words, lines etc and then classifies groups as either drawing or writing assuming that the groups will not overlap.

Many of the more sophisticated recognizers combine bottom-up and top-down. Parsing in one direction and then the other to reduce ambiguity.

A range of approaches are used during the recognition process employing numerous combinations of algorithms. Surprisingly, given that stroke features are so fundamental to recognition, there is little evidence of the use of formal methods to identify significant features. We have identified 46 different features that could be used to support basic ink segment recognition. Most of these have been selected for use in previous sketching tools heuristically [Rub91; SSD01; YC03].

Our approach is to use formal statistical analysis methods to identify key ink features to improve recognition. As a first step we have applied a classification of ink features to the problem of dividing writing and drawing as this is fundamental to non-modal, mixed text/shape diagram recognition. The following section outlines the methodology used to identify these features.

3. Ink Feature Analysis

We provide an overview of our approach beginning with our investigation of the range of possible ink features, how feature data is collected and analysed, our identified set of key ink features, and the initial results of an evaluation of a text/shape divider based on this key ink feature set.

3.1 Candidate Feature Set Formation

Our first step was to identify possible ink features that could be useful in distinguishing between text and shape strokes in a sketched diagram. The origin of these features were from (1) related work in sketch recognition; (2) stroke features we felt may be useful in classifying strokes; (3) and stroke features from newly available hardware (e.g., pressure sensitive Tablet PC screens).

Forty-six features were selected in total, grouped into seven categories: size, time, intersections, curvature, pressure, operating system [MC05] recognition values, and inter-stroke gaps. (See the Appendix for a full listing of the candidate feature set).

3.2 Data Collection

To test which are important features, we collected sample sketches and calculated measurements of each feature from these sketches. The measurements formed the dataset used for analysis.

We identified a set of nine diagram types and their graphical/textual notations (Figure 1) to be used for our experiment. In compiling this set we looked for examples of shapes and text that would represent those typical of a wide range of diagram types and therefore would allow us to identify the most significant ink features of strokes for division in a general-purpose, reusable text/shape divider. Our diagram set includes basic shapes and text, complex shapes, composite shapes and various combinations and ordering of shapes and text. Sketches were gathered from 26 people using InkKit [CMP05; You05; PTY06; FP07] on a Toshiba Portege M200 Tablet PC. Each person completed

a set of 9 sketches. Each sketch was then processed to measure the 46 features from our candidate feature set, forming a final dataset with 1519 strokes for statistical analysis. We manually categorized each stroke as shape or text to form the base data for our statistical analysis.

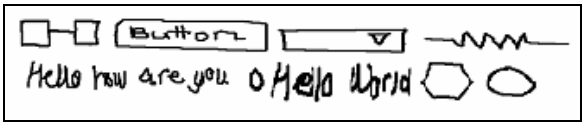

Figure 1: *The diagram set.*

3.3 Analysis

The dataset prepared above was analysed to determine the ink features of strokes that are significant and thus should be used to assist algorithms in a text/shape divider. We used a formal statistical analysis technique to gain a clear, accurate, and principled view of the degree of significance of each ink feature in distinguishing between shapes and text in a hand-drawn diagram.

One way of finding the significant features is to employ a statistical partitioning technique [BFO*84; VR02]. This involves taking a dataset and finding which features can be used to split the data into the required groups most accurately, i.e. into text or shape strokes, based on each observation's measurement of the features. These can then be arranged into a decision tree with the most significant features at the root.

The classification tree has decision variables at each child node, which correspond to the other most significant features found, and a classification label at each leaf. In our case, as there are only two classes of interest, the leaf is either text or shape (Figure 3). Employing this technique allowed us to clearly identify significant features to help division, and also provided us with the most optimal combination of features for implementation.

The analysis of the dataset was performed using the R statistical package, [RDC06]. The dataset was used as training data for the *rpart* function [VR02]. The *rpart* function applies tree-based partitioning to identify the significant features. For each feature (e.g., length, average speed, curvature, average pressure value, etc.) *rpart* is provided with the value for each stroke in the dataset and the known classification of the stroke, i.e., *shape* or *text*.

The aim is to find the most optimal position for a split to be made so that there are a minimal amount of misclassified strokes. If this is done for all features in the feature set, using the observations in the dataset, then the features that most accurately split the data into text and shape stroke groups, with the least amount of misclassified strokes, will be identified as the significant features for division of text and shape strokes. Figure 2 shows a partition of the dataset observations for the bounding box width feature.

Using this partitioning technique, the *rpart* function constructs a binary classification tree starting with the most significant feature, then the next, then the next and so on. The best partitioning feature is chosen by minimizing a measure of purity using the Gini index which is a measure of the misclassification rate for that partition [VR02]. The partitioning process is continued until the number of observations at each leaf is < 20 or they are all known to be either *shape* or *text* strokes.

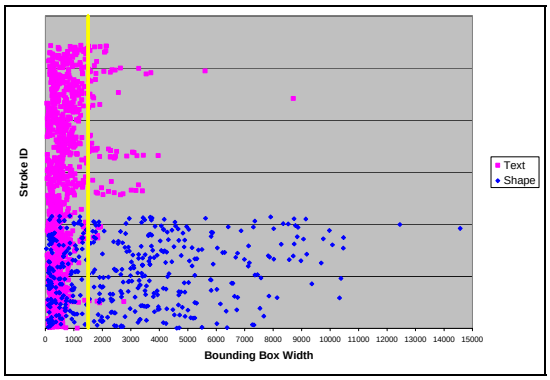

Figure 2: *Proposed partition for bounding box width.*

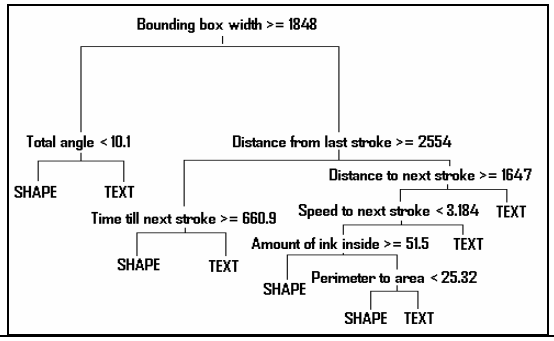

Figure 3: *Classification tree for text and shape divider.*

The binary classification tree resulting from our *rpart* analysis of our full training set is shown in Figure 3. Eight different features of strokes, named in each node of the classification tree, were identified from the feature set as being significant for dividing shape from text strokes. They measure the inter-stroke gaps, size and curvature of a stroke (Table 1). Applying the resultant decision tree now classifies strokes into *shape* or *text* using these significant ink features.

For example, consider classifying a new stroke. First we sample its bounding box width. If it's bounding box width is >= 1848 Himetric units (HU) we follow the left branch of the tree. We then measure its total angle. If its total angle is < 10.1 radians the left branch is taken once again and the stroke is classified as a *shape*.

3.4 Initial Results

The new divider was implemented in the InkKit [CMP05; You05; PTY06; FP07] generic sketching tool alongside InkKit's existing divider and the Ink SDK Microsoft divider [MC05] to determine which is more accurate at dividing strokes. This was assessed by determining which has the lowest misclassification rate, where the misclassification rate is a measure of the proportion of strokes that are incorrectly classified.

Each of the dividers were used to divide the original training set of diagrams, shown in Figure 1, into shape and text segments. We also applied them to a new set of diagram examples (Figure 6) that included more complex

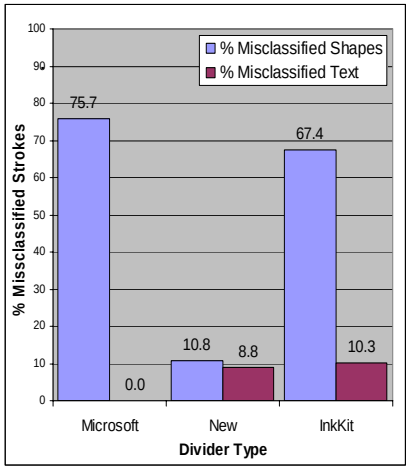

Figure 4: *Percentage of misclassified shape and text strokes for each divider using the training diagram set.*

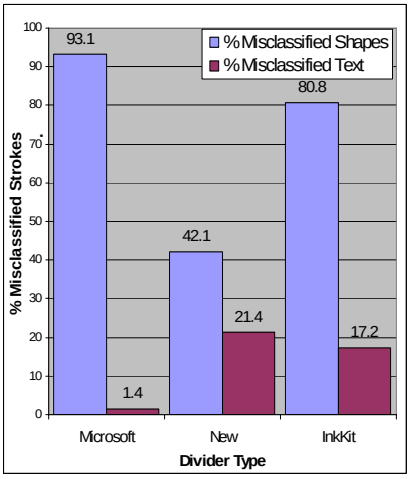

Figure 5: *Percentage of misclassified shape and text strokes for each divider using the new diagram set.*

notations exhibiting characteristics not considered previously. Musical notes were included because a student had explored using InkKit for writing music; however they are a specialized script, rather than a diagram.

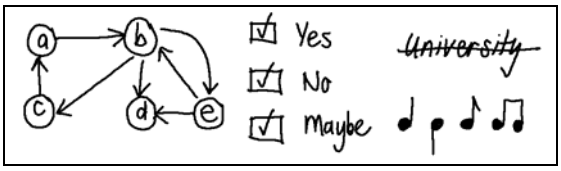

Figure 6: *The new diagram set.*

The percentage of shape and text strokes that were misclassified in the training set of diagrams for each divider is shown in Figure 4. The Microsoft divider has the highest percentage of misclassified shape strokes at 75.7% and the lowest percentage of misclassified text strokes; note that no text strokes were incorrectly classified. Our new divider has the lowest proportion of misclassified shape strokes when compared with the other dividers at 10.8%, and the second lowest proportion of misclassified text strokes at 8.8%. The InkKit divider has a higher misclassification rate for shape strokes at 67.4%, coming in as the second highest of all dividers, however in contrast it has a low percentage of misclassified text strokes at 10.3%. All dividers showed much greater accuracy in classifying text strokes than shape strokes.

Using the new diagram set the percentage of misclassified shape and text strokes for the three dividers are shown in Figure 5. The Microsoft divider once again has the worst rate of misclassification of shape strokes where 93.1% were incorrectly classified and the best percentage of misclassified text strokes at 1.4%. This follows the pattern shown in the evaluation results for the training diagram set shown in Figure 4. Also, following the results of the first evaluation, our new divider has the lowest misclassification rate for shape strokes at 42.1%, although this is still very high. The new divider has the highest percentage of misclassified text strokes at 21.4% however this is only slightly above InkKit at 17.2% for text strokes. InkKit's rate of misclassification for shape strokes comes in at 80.8%. The music notes were a confounding factor, removing them from the set would have improved the recognition rates for all dividers. Again, all dividers show a greater degree of accuracy in classifying text strokes than shape strokes.

4. Discussion

There have been many sketch recognition techniques described in the literature. However, their accuracy rates are still far less than optimum. A review of the literature has shown that these techniques are comprised of two elements: extraction of some ink features and various algorithms that are applied successively to the diagram in either a bottom-up, top-down or mixed approach. There has been little evidence of formal methods used to select the most significant ink features and recognition algorithms suited to different sketch recognition problems and target diagram notation domains.

The majority of diagramming sketch tools do not support text recognition. This may be because separating shape and text strokes so that they can subsequently be recognized by more specific recognition algorithms is a very difficult problem in general. As division is a logical first step for bottom-up recognition we chose to concentrate on identifying the distinguishing features of text versus shape strokes using a formal method for optimal ink feature selection.

The new divider showed promising results in terms of its accuracy when compared with InkKit's divider and the Microsoft divider [MC05]. From this trial the ink features that we found to be significant to the division problem were features measuring elements of stroke size, inter-stroke gaps, and stroke curvature as shown in Table 1.

Inter-stroke gaps have proved to be an important distinguishing feature when dividing shape and text strokes with half of the key feature set coming from this category. The inclusion of features measuring the distance between strokes indicates that the distance from shape-to-shape and shape-to-text is larger than from text-to-text. The features measuring inter-stroke timing may be underestimated as the participants copied the diagrams. The data suggests a much shorter gap between the letters of a word as opposed to

shape-to-shape and word-to-shape. If one was creating a new diagram the cognitive switching gaps between diagram elements may well be longer than in our dataset.

Category	Origin	Feature
Inter-stroke gaps	New	Time till next stroke
		Speed till next stroke
	Adapted from [You05]	Distance from last stroke
		Distance to next stroke
Size	Adapted from [HD02;FPJ02]	Bounding box width
	[FPJ02]	Perimeter to area
	Adapted from [You05]	Amount of ink inside
Curvature	[Rub91]	Total angle

Table 1: *Significant feature categories and origin.*

Three of the significant ink features identified are size related. The data suggests that in general the size of shape strokes is much larger than text strokes reflected by the use of bounding box width, perimeter to area and amount of ink inside features. One point to note is that these values are specific to the tablet PC form factor. Size will have very different parameters when using a PDA or electronic whiteboard – however the ratio should remain similar.

The selection of the feature total angle is the only measure of curvature included in the key feature set. Its position in the tree is below the width decision, on the wider side, suggesting that curvature is relevant for differentiating joined up letters from shapes.

Figure 7 shows examples of strokes in a diagram that might go down each branch. Stroke (1) is wide with a small total angle and is classified as a *shape* stroke, whereas stroke (2) is also wide but has a larger total angle and is therefore classified as *text*.

Strokes (3-9) are all smaller in width than strokes (1-2). For strokes (3-4) their distance from the last stroke is large which could be because they are the first stroke of a shape or word. (3) is classified as a *shape* stroke as the time to the next stroke is large, and (4) a *text* stroke as this inter-stroke time is small which is highly likely if subsequent strokes are the remainder of the word.

(9) is likely to be a letter that is in the middle of a word as it has a small inter-stroke distance from the last and to the next stroke. The path of stroke (8) shows that it has a small distance from the last stroke as another letter probably came before it, a larger distance to the next stroke which could indicate that there is another word that follows this stroke, and a faster speed to the next stroke which further confirms that it is likely to be a letter in the middle of a sentence.

Stroke (7) is similar to (8) except that it is more likely to be the end of a word that is followed by a shape stroke as it has a slower speed to the next stroke indicating the time taken for cognitive shift, it also has a small amount of ink inside its bounding box indicating that it is a small stroke and has a large perimeter to area ratio.

In contrast, stroke (6), has a smaller perimeter to area ratio indicating that it is a *shape* stroke. The fact that it has a small distance from the last stroke and large distance to the next stroke indicates that it may be the last stroke of a shape component such as a check box. It also has a slow speed to the next stroke indicating a delay in drawing which could be caused by a large cognitive shift when beginning another shape or word. Note that (6) is smaller in width than stroke (1). Stroke (5) is similar to (6) except that it has a greater amount of ink inside its bounding box.

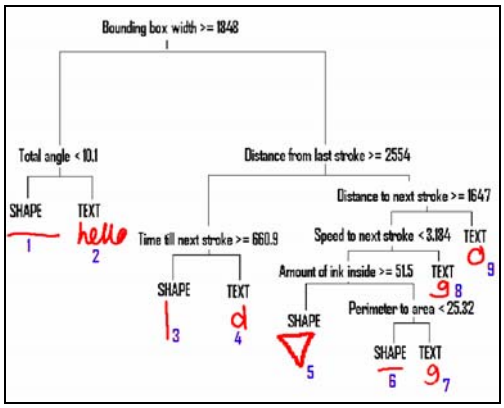

Figure 7: *Example stroke classifications.*

Many of the features used in other recognition engines, such as those that measure stroke pressure, intersections, time, and the Tablet OS recognition values were not considered important for a divider. These features may however be used for other recognition problems and may be appropriate in the area they are used.

The significant features identified have proved to be successful for text/shape division. However, feature selection for division is only one step of recognition. We now need to identify the best algorithm to use to meaningfully combine these features. The tree-based partitioning technique used to analyse the dataset of sketch features not only identifies feature sets for distinguishing text from shape strokes but also provides a method of utilising and combining those features in a practical way using a binary tree approach to classification.

Rpart partitioning is a good way to separate data into a small number of classes; therefore, it works well for the division problem. However, it may not be a suitable method of analysis when there are a lot of classes.

Although the classification tree works well at combining the significant features together to divide shape and text strokes, greater flexibility is needed in such an algorithm. Sketching is imprecise in nature therefore recognition algorithms need to be robust enough to accommodate variability. The binary classification trees used hard rules to classify observations which are based on a single dataset. An algorithm, such as HMM, that can be trained using these features would provide added flexibility.

Recognition engines use complex interrelationships of features and algorithms. By isolating and optimizing each part of the process future progress should be possible. This project is a first step in this direction.

5. Conclusion

Features identified by the statistical partitioning technique have improved division of text and shape strokes in our

example data set. The combination of the new feature set and optimised algorithms needs investigation to determine the best technique for a more flexible divider. However, the divider is only one part in the sketch recognition process. Further improvement to sketch recognition should be possible if a similar rigorous approach of feature analysis is applied to the basic shape and component recognition phases. The identification of significant feature sets for each of these phases together with an evaluation of various algorithmic approaches to combining these features should produce more accurate recognition results as demonstrated by this investigation and therefore, improve sketch recognition as a whole.

References

[AD04] Alvarado, C. and Davis, R.: SketchREAD: a multi-domain sketch recognition engine. Proceedings of the 17th annual ACM symposium on User interface software and technology, ACM Press (2004),23-32.

[BK03] Bailey, B. P. and Konstan, J. A.: Are Informal Tools Better? Comparing DEMAIS, Pencil and Paper, and Authorware for Early Multimedia Design. CHI 2003, ACM (2003),313-320.

[BKC01] Bailey, B. P., Konstan, J. A. and Carlis, J. V.: DEMAIS: Designing Multimedia Applications with Interactive Storyboards. ACM Multimedia, (2001),pp. 241-250.

[Bla90] Black, A.: Visible planning on paper and on screen: The impact of working medium on decision-making by novice graphic designers. Behaviour and information technology 9 (4)(1990), 283-296.

[BFO*84] Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J.: Classification and Regression Trees. Chapman & Hall / CRC Press, 1984.

[CSK*02] Calhoun, C., Stahovich, T. F., Kurtoglu, T. and Kara, L. B.: Recognising Multi-Stroke Symbols. AAAI Spring Symposium on Sketch Understanding, (2002),15-23.

[CGH03] Chen, Q., Grundy, J. and Hosking, J.: An E-whiteboard application to support early design-stage sketching of UML diagrams. Human Centric Computer Languages and Environments, IEEE (2003),219-226.

[CMP05] Chung, R., Mirica, P. and Plimmer, B.: InkKit: A Generic Design Tool for the Tablet PC. CHINZ 05, ACM (2005),29-30.

[CFK*04] Coyette, A., Faulkner, S., Kolp, M., Limbourg, Q. and Vanderdonckt, J.: SketchiXML: towards a multi-agent design tool for sketching user interfaces based on USIXML. Proceedings of the 3rd annual conference on Task models and diagrams, ACM Press (2004),75-82.

[DHT00] Damm, C. H., Hansen, K. M. and Thomsen, M.: Tool support for cooperative object-oriented design: Gesture based modelling on and electronic whiteboard. Chi 2000, ACM (2000),518-525.

[FJ00] Fonseca, M. J. and Jorge, J. A.: Using Fuzzy Logic to Recognize Geometric Shapes Interactively. Proceedings of the 9th International Conference on Fuzzy Systems (FUZZ-IEEE), (2000),

[FJ01] Fonseca, M. J. and Jorge, J. A.: Experimental Evaluation of an on-line Scribble Recognizer. Pattern Recognition Letters, (2001),1311–1319.

[FPJ02] Fonseca, M. J., Pimentel, C. e. and Jorge, J. A.: CALI: An Online Scribble Recogniser for Calligraphic Interfaces. AAAI Spring Symposium on Sketch Understanding, IEEE (2002),

[FPJ02] Fonseca, M. J., Pimentel, C. e. and Jorge, J. A.: CALI: An Online Scribble Recognizer for Calligraphic Interfaces. AAAI Spring symposium on Sketch Understanding, IEEE (2002),51-58.

[FP07] Freeman, I. and Plimmer, B.: Connector Semantics for Sketched Diagram Recognition. AUIC, ACM (2007),71-78.

[Goe95] Goel, V.: Sketches of thought. The MIT Press, 1995.

[Gro94] Gross, M.: Recognizing and interpreting diagrams in design. AVI 94, ACM (1994),88-94.

[GD96] Gross, M. and Do, E. Y. L.: Ambiguous intentions: a paper-like interface for creative design. UIST '96, ACM (1996),183-192.

[HD02] Hammond, T. and Davis, R.: Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. 2002 AAAI Spring Symposium on Sketch Understanding, (2002),

[HD03] Hammond, T. and Davis, R.: LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition. IJCAI, (2003),12-19.

[HL00] Hong, J. I. and Landay, J. A.: SATIN: a toolkit for informal ink-based applications. Proceedings of the 13th annual ACM symposium on User interface software and technology, ACM Press (2000),

[HSN04] Hse, H., Shilman, M. and Newton, A. R.: Robust Sketched Symbol Fragmentation using Templates. International Conference on Intelligent User Interfaces, (2004),pp. 156-160.

[JF99] Jorge, J. A. and Fonseca , M. J.: A Simple Approach to Recognise Geometric Shapes Interactively. In Proceedings of the Third Int. Workshop on Graphics Recognition (GREC'99), (1999),

[KS04] Kara, L. B. and Stahovich, T. F.: Hierarchical Parsing and Recognition of HandSketched Diagrams. UIST '04, ACM Press (2004),13 - 22.

[Lan95] Landay, J.: Interactive sketching for user interface design. Chi '95 Mosaic of Creativity, Doctoral Consortium, (1995),63-64.

[LM95] Landay, J. and Myers, B.: Interactive sketching for the early stages of user interface design. Chi '95 Mosaic of Creativity, (1995),43-50.

[LM96] Landay, J. and Myers, B.: Sketching storyboards to illustrate interface behaviors. CHI '96, ACM (1996),193-194.

[Lan03] Lank, E. H.: A Retargetable Framework for Interactive Diagram Recognition. Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 1, IEEE Computer Society (2003),185.

[LVZ04] LaViola, J. J. and Zeleznik, R. C.: MathPad2: a system for the creation and exploration of mathematical sketches. ACM Trans. Graph (2004),432-440.

[LL03] Lin, J. and Landay, J. A.: Damask: A Tool for Early-Stage Design and Prototyping of Cross-Device User Interfaces. CHI 2003 workshop on HCI Patterns: Concepts and Tools, (2003),

[LNH*00] Lin, J., Newman, M. W., Hong, J. I. and Landay, J. A.: Denim: Finding a tighter fit between tools

and practice for web design. Chi 2000, ACM (2000),510-517.
[MC05] Microsoft Corporation: Microsoft Windows XP Tablet PC Edition Software Development Kit (2005)
[MC07] Microsoft Corporation:MSDN Stroke.BezierCusps Property from http://msdn2.microsoft.com/en-us/library/microsoft.ink.stroke.beziercusps.aspx. (2007).
[MC07a] Microsoft Corporation: MSDN Stroke.PolylineCusps Property. from http://msdn2.microsoft.com/en-us/library/microsoft.ink.stroke.polylinecusps.aspx. (2007).
[NSS02] Nakai, M., Sudo, T., Shimodaira, H. and Sagayama, S.: Pen Pressure Features for Writer-Independent On-Line Handwriting Recognition Based on Substroke HMM. Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02) Volume 3 - Volume 3, IEEE Computer Society (2002),30220.
[Pli04] Plimmer, B.: Using Shared Displays to Support Group Designs; A Study of the Use of Informal User Interface Designs when Learning to Program. Computer Science, University of Waikato. PhD.(2004)
[PTY06] Plimmer, B., Tang, G. and Young, M.: Sketch Tool Usability: Allowing the user to disengage. HCI ACM (2006),164-167.
[PA03] Plimmer, B. E. and Apperley, M.: Evaluating a Sketch Environment for Novice Programmers. SIGCHI, ACM (2003),1018-1019.
[PA03b] Plimmer, B. E. and Apperley, M.: Freeform: A Tool for Sketching Form Designs. BHCI, (2003),2, 183-186.
[PA03c] Plimmer, B. E. and Apperley, M.: Software for Students to Sketch Interface Designs. Interact, (2003),73-80.
[PA04] Plimmer, B. E. and Apperley, M.: INTERACTING with sketched interface designs: an evaluation study. SigChi 2004, ACM (2004),Extended Abstracts, 1337-1340.
[Qin05] Qin, S.: Intelligent Classification of Sketch Strokes. EUROCON, IEEE (2005),1374-1377.
[RDC06] R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing.(2006)
[Rub91] Rubine, D. H.: Specifying gestures by example. Proceedings of Siggraph '91, ACM (1991),329-337.
[SSD01] Sezgin, T. M., Stahovich, T. and Davis, R.: Sketch based interfaces: early processing for sketch understanding. Proceedings of the 2001 workshop on Perceptive user interfaces, ACM Press (2001),1-8.
[SWR*03] Shilman, M., Wei;, Z., Raghupathy;, S., Simard, P. and Jones, D.: Discerning structure from freeform handwritten notes. Document Analysis and Recognition, (2003),60 - 65.
[Sta97] Stahovich, T. F.: Interpreting the engineer's sketch: A picture is worth a thousand constraints. AAAI Symposium on Reasoning with diagrammatic Representations II, (1997),31-38.
[SDS95] Stahovich, T. F., Davis, R. and Shrobe, H.: Turning sketches into working geometry. ASME Design Theory and Methodology, (1995),603-611.
[VR02] Venables, W. N. and Ripley, B. D.: Modern Applied Statistics with S. Springer, 2002.
[You05] Young, M.: InkKit: The Back End of the Generic Design Transformation Tool University of Auckland.(2005)
[YC03] Yu, B. and Cai, S.: A domain-independent system for sketch recognition. Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, ACM Press (2003),141-14

Appendix: Feature Set

Feature	Description	Origin
Pressure Features		
Max pressure	Maximum pressure value for the stroke.	Adapted from [NSS02]
Min pressure	Minimum pressure value for the stroke.	
Average pressure	Mean average pressure of the stroke.	
# Pressure minima	Number of minima in the pressure values for the stroke, this excludes the minima that occur at the beginning and end of the stroke for pen up/down events.	New
Time Features		
Total duration	Total duration of the stroke from pen up to pen down.	[Rub91]
Max speed	Maximum speed when drawing the stroke.	Adapted from [Rub91]
Min speed	Minimum speed when drawing the stroke.	
Average Speed	Mean average speed when drawing the stroke.	
Intersection Features		
# Self intersections	Number of points where the stroke intersects itself.	Adapted from [Qin05]
# Endpoint self intersections	Number of self intersections at the endpoints of the current stroke.	
# Other self intersections	Number of self intersections that are not at the current stroke's endpoint.	
# Other intersections	Number of points of intersection of the current stroke with other strokes (excluding self intersections).	Adapted from [CSK*02]
Total # intersections	Total number of intersections (includes self intersections).	
# Other strokes intersecting	Number of other strokes that intersect the current stroke (excluding itself).	Adapted from [HD02;
Total # strokes	Number of strokes that intersect the current stroke (including itself).	

intersecting		FPJ02]
Size Features		
Length	Total length of the stroke.	[Rub91]
Bounding box area	Area of the bounding box of the stroke.	Adapted from [HD02; FPJ02]
Bounding box height	Height of the bounding box of the stroke.	
Bounding box width	Width of the bounding box of the stroke.	
Amount of ink inside	Amount of ink inside the strokes bounding box. This is calculated by counting the number of points of the stroke that are inside the bounding box.	[You05]
Bounding box diagonal length	Length of the bounding box diagonal line.	[Rub91]
Distance from first to last point	Distance from the first point of the stroke to the last point of the stroke.	
Total length/bounding box diagonal length	Length of the stroke divided by the length of the bounding box diagonal.	Adapted from [Rub91];
Perimeter to area	Ratio of perimeter to area of the strokes convex hull.	[FPJ02]
Convex hull area ratio	Ratio of the area of the convex hull to the area of the enclosing rectangle of the stroke.	
Rectangle ratio	Ratio of strokes enclosing rectangle width to height.	
Width to height	Ratio of the strokes bounding box width to height.	Adapted from [FPJ02]
Curvature Features		
Cos of initial angle	Cosine of the initial angle of the stroke.	[Rub91]
Sin of initial angle	Sine of the initial angle of the stroke.	
Angle of bounding box diagonal	Angle of the bounding box diagonal.	
Cos from 1st to last point	Cosine of the angle between the first and last point of the stroke.	
Sin from 1st to last point	Sine of the angle between the first and last point of the stroke.	
Total angle	Total angle traversed by the stroke.	
$\sum$ |angle at each point|	Sum of the absolute value of the angle at each point of the stroke.	
$\sum$(angle at each point)^2	Sum of the squared value of the angle at each point of the stroke.	
# Bezier cusps	Number of bezier cusps. Cusps indicate points where the direction of the stroke has changed; therefore Bezier cusps are the cusps for a Bezier curves control points for the stroke [MC07]. *	New
# Polyline cusps	Number of polyline cusps. Cusps indicate points where the direction of the stroke has changed; therefore polyline cusps are the cusps of the points of a stroke [MC07a]. *	
# Speed minima	The number of extreme minima in the speed values for the stroke, this excludes the minima that occur at the beginning and end of the stroke for pen up/down events.	Adapted from [SSD01]
Tablet OS Recognition Features		
Tablet OS text probability	Tablet OS text recogniser probability of the stroke being text, levels of probability are strong, intermediate or poor.	[MC05]
Tablet OS prediction	Tablet OS text recogniser best prediction of the text symbol that the stroke represents.	
Inter-Stroke Gaps		
Distance from last stroke	Distance the pen travels between the current stroke and the previous stroke. Not applicable to the first stroke in the sketch.	Adapted from [You05]
Distance to next stroke	Distance the pen travels between the current stroke and the next stroke. Not applicable to the last stroke in the sketch.	
Time from last stroke	The time between the current stroke and the previous stroke in the sketch. Not applicable to the first stroke of a diagram.	New
Time till next stroke	The time between the current stroke and the next stroke in the sketch. Not applicable to the last stroke of a diagram.	
Speed from last stroke	Speed (distance/time) between the current stroke and the previous stroke in the sketch. Not applicable to the first stroke of a diagram.	
Speed to next stroke	Speed (distance/time) between the current stroke and the next stroke in the sketch. Not applicable to the last stroke of a diagram.	

*For example take the letter “L”. This letter has three cusps (Bezier and polyline) one at the start and end of the letter and one at the corner [MC07, MC07a]