Exploring Better Techniques 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
{rpat088@ec, beryl@cs, john-g@cs, ihaka@stat}.auckland.ac.nz

Abstract. A critical component of diagramming sketch tools is their ability to reliably recognise hand-drawn diagram components. This is made difficult by the presence of both geometric shapes and characters in diagrams. The goal of our research is to improve sketch recognition by improving the accuracy in grouping and classifying strokes in a diagram into text characters and shapes. We have done this by identifying the most significant features of strokes that can be used to distinguish shapes from text using a decision tree based partitioning technique. Implementation and evaluation of this new "shape divider" using these features against InkKit's existing divider and the Microsoft divider has shown that our divider is more accurate at dividing text and shape strokes and can therefore improve overall sketch recognition.

Keywords: Human Computer Interaction, Pattern Recognition

1. Introduction

Computers can be used to produce formal diagrams e.g. software designs, user interface designs, CAD specifications, hierarchy charts and so on. However, in the beginning of a project it can be better to use pen and paper. The reason is that these simple, human-centric design tools are far more flexible which encourages creativity and in turn this produces better designs in the end. However a computer offers easy editing and distribution features and greater formality to the look of a diagram and so sometimes the important pen and paper design stage is overlooked.

Imagine being able to sketch diagrams directly onto a computer screen and have them accurately translated into formalized representations. The Tablet PC supports sketching of characters and recognition of these but has much weaker support for general diagram drawing and accurate recognition. InkKit [8] is a sketch tool for the Tablet PC that has been designed to bridge the gap between pen and paper and computers. InkKit allows you to sketch any type of diagram using the Tablet PC's stylus. Its recognition engine then identifies each component of the diagram and transforms it into a formal, tidy version in a specified format such as HTML or a Word drawing object. InkKit has been used as a foundation for this research.

2. Motivation

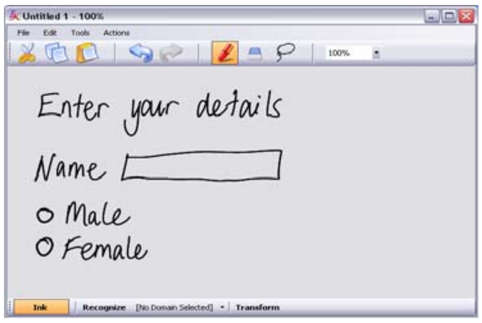


Fig 1. Sketch of a user interface design

A critical part of diagramming sketch tools is being able to reliably recognise the hand-drawn diagram components. The purpose of this Masters project is to improve the recognition algorithms that currently exist.

One of the reasons why recognising a diagram is difficult, is because we are dealing with words and shapes at the same time. Figure 1, shows a typical user interface design that can be used to highlight this problem. For example, what is it that makes the circles in front of the words "Male" and "Female" circles indicating radio buttons and not the letter o?

The existing InkKit system automatically divides the ink into words and shapes, and then recognises the words using the Tablet Operating System recognizer and shapes using a variant of Rubine's algorithm [5]. Once the basic shapes and words have been recognised they are combined back together to suggest the most probable diagram component. This method of dividing the words and shapes is preferred over using a modal interface to separate the two as it preserves human's natural sketching approaches used on pen and paper [3]. The aim of our research has been to greatly improve the algorithm that divides the ink into either text or drawing as this is the area where we can expect the most improvement in diagram recognition [8].

Most sketch recognition mechanisms measure various features of the strokes in a sketch to guide its recognition. This is also evident in the recognition engine of InkKit. However what is lacking is a definitive, principled set of the most significant features that can be used to provide an accurate division of the shape and text strokes in a sketch. Therefore our goal is to find these features in order to improve division of shape and text strokes and in doing so improve the accuracy of sketch recognition as a whole.

We first detail the methodology we have used to identify the most significant features of sketches that will aid division of strokes. Section 4 will briefly describe the selected feature set and then detail the experiment conducted to identify which features from the set are significant. The implementation and evaluation of the resulting divider of shape and text strokes is presented in Section 5. Finally we conclude with a discussion of the results and suggestions for future work.

3. Methodology

In this section we provide an overview of the approach we have used beginning with the investigation of possible sketch features, how feature data is collected and analysed, and the implementation and evaluation of the resulting divider of text and shape strokes.

Feature Discovery

Our first step was to identify all the possible 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. Many stroke features have been documented by past work to be significant to solving various sketch recognition problems [2, 5, 6], and several of these features we have already implemented in InkKit's existing recognition engine [8]. Therefore it was in our interest to include as many of these features as possible in our investigation. Newly available hardware allows us to consider features that have not been widely studied before such as pen pressure and tilt.

Data Collection

We wanted to determine which features are actually the most significant to use when dividing text and shape strokes. This enables us to sample and use only those features when classifying strokes in InkKit's divider and not the myriad of other features that have little or no effect on stroke classification. To be able to test which are important features and which aren't we collected measurements of each feature from sketches and compiled these measurements into a dataset.

We identified a set of diagrams to be used for our experiment that displayed a wide range of common characteristics of diagrams. This allowed us to sample our stroke feature set on a realistic group of diagrams so that we are more likely to achieve a more accurate identification of significant features.

The next step was to get people to sketch examples of this set of diagrams using InkKit on Tablet PC's. We needed to collect diagram sets from as many people as possible to avoid bias in our samples of individual variation that may occur in their sketching. Once these diagrams were collected we processed them by calculating all stroke features identified from each stroke of each diagram. The resulting data was then collated into a dataset ready for statistical analysis.

Analysis

Once all the sketches were collected and processed into a dataset they were analysed to determine the features of strokes that are significant and should be used to divide text and shape strokes. We wanted to use a formal technique for this analysis so that we would gain a clear, accurate and principled view of the degree of significance each feature has in distinguishing between shapes and text in a hand-drawn diagram.

The use of trees has become a common and effective way to assist in decision making problems, our decision being whether to classify a stroke in a diagram sketch as a text or shape stroke. A tree-based partitioning technique was used to analyse the dataset and construct a classification tree [1, 7]. A classification tree has decision variables at each node, which would correspond to the most significant features found, and a classification label at each leaf, which would be either text or shape in our case (see Figure 2). 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.

Implementation & Evaluation

Once the significant stroke features had been identified through formal statistical analysis a new shape divider was implemented for InkKit. InkKit's existing divider remains as we wanted to compare its performance and the Tablet PC Microsoft divider to our new shape divider. We evaluated these three divider implementations to determine which is more accurate at dividing text and shape strokes.

4. Experiment

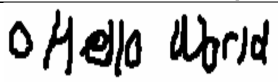
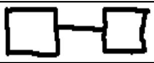
All the possible stroke features that may be helpful in distinguishing text from shape strokes were compiled together into a feature set. They came from related work in sketch recognition [2, 5, 6, 8]; features we thought may be significant; and the discovery of features from newly available Tablet PC hardware. We attempted to include as many features of strokes as possible to ensure that all avenues would be explored by the statistical analysis and the most significant features could be identified – 52 features were selected in all. The 52 features can be grouped into seven categories, size, time, intersections, curvature, pressure, Tablet Operating System recognition values and others.

Our stroke feature set was then analysed to discover which of these features provide the greatest contribution to dividing text and shape strokes in a diagram. We describe this experiment beginning with the collection and processing of data from a range of sketched diagrams, then analysing this data using a tree-based partitioning technique that consequently constructs a classification tree containing the most significant features for division of text and shape strokes. This process allowed us to draw conclusions as to which features are most significant to division.

Data Collection & Processing

To perform a full analysis of the feature set sketched diagrams were collected from as many people as possible. A set of nine diagrams were identified to be used for analysis, illustrated in Table 1. 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 features of strokes for division to be identified for a general-purpose, reusable 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. Each person completed a set of 9 sketches. Each sketch was then processed to obtain the 52 features from our feature set, forming a final dataset ready for statistical analysis with 1519 observations in all. We manually categorized each stroke as SHAPE or TEXT as base data for the statistical analysis.

Table 1. The Diagram Set

Shape Description	Example Sketch
Circle	
Button: rectangle with label "Button" inside it.	
Text: "Hello how are you", without punctuation.	
Radio-text: radio button with label "Hello world" next to it.	
Text-radio: same as radio-text above but label written before radio button (spatial ordering is the same).	
Combo box: rectangle with a triangle inside.	
Resistor: spiked line, from electrical diagrams.	
Hexagon: six-sided polygon.	
Connector: two rectangles with a line connecting them in the middle.	

Analysis

The analysis of the dataset was performed using the R statistical package, pre-release Version 2.5.0 [9]. The dataset was used as training data for the rpart function [7] which applies a tree-based partitioning technique to identify the significant features to use as decision variables in a binary classification tree and determines how to split those features so that they can accurately classify strokes as either text or shape strokes. For each feature rpart is provided with each example stroke's feature data e.g. length, average speed, curvature, average pressure value etc, and the known classification of the stroke i.e. SHAPE or TEXT. From the total training set the rpart function constructs a binary classification tree starting with the most significant feature, then the next, then the next and so on.

Results

The binary classification tree resulting from our rpart analysis of our full training set is shown in Figure 2. 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, each one is described in Table 2. The majority of these features are measuring some element of size, as five out of eight of the features come from this category. Features of time have also registered their importance as making up two out of eight of the significant features. Also one feature of curvature has been identified as important to the division of shape and text strokes.

This resultant decision tree is used to classify strokes into SHAPE or TEXT. For example, consider classifying a given stroke. First we sample its bounding box width. If its bounding box width is >= 1848 we follow the left branch of the tree. We then measure its total angle. If its total angle is < 10.1 the left branch is taken once again and the stroke is classified as a SHAPE. Taking another stroke, we calculate its

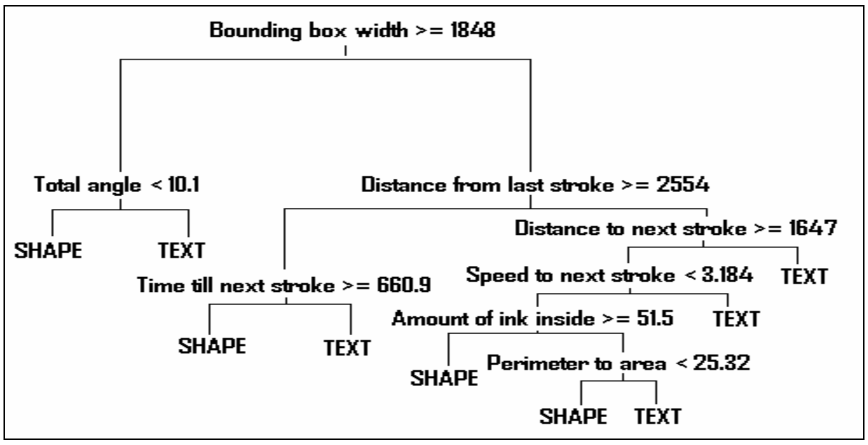

Bounding box width >= 1848
Total angle < 10.1
SHAPE
TEXT
Distance from last stroke >= 2554
Distance to next stroke >= 1647
Speed to next stroke < 3.184
TEXT
Time till next stroke >= 660.9
SHAPE
TEXT
Amount of ink inside >= 51.5
TEXT
SHAPE
Perimeter to area < 25.32
SHAPE
TEXT

Fig 2. Classification tree for text and shape divider

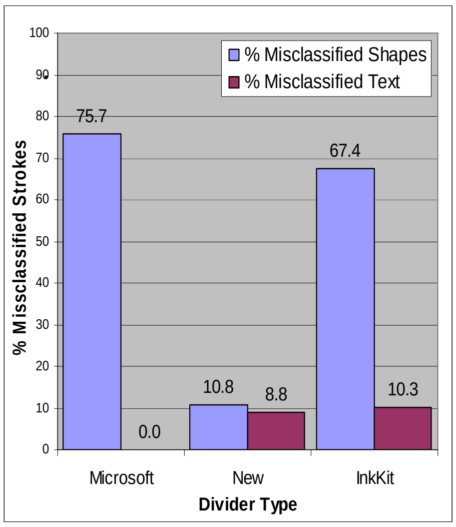
Table 2. Description of the significant features identified in the classification tree

Feature	**Description**
Bounding box width	Width of the bounding box of the stroke.
Distance from last stroke	Distance the pen travels between the current stroke and the previous stroke.
Distance to next stroke	Distance the pen travels between the current stroke and the next stroke.
Amount of ink inside	Amount of ink inside the strokes bounding box.
Perimeter to area	Ratio of perimeter to area of the strokes convex hull.
Time till next stroke	The time between the current stroke and the next stroke in the sketch.
Speed to next stroke	Speed (distance/time) between the current stroke and the next stroke in the sketch.
Total angle	Total angle traversed by the stroke.

bounding box width and if this is < 1848 we follow the right hand side branch. We then calculate the strokes distance from the last stroke, and if this is < 2554 we again follow the right hand side branch. We then calculate its distance to the next stroke, and if this is < 1647, we classify the stroke as TEXT.

5. Implementation and Evaluation

A new shape divider based on the classification tree from Figure 2 was implemented in InkKit, as an alternative to InkKit's existing divider and the Microsoft Tablet PC divider. An evaluation was carried out comparing the three dividers performance using our example diagrams to establish which would be able to divide text and shape strokes most accurately, i.e. with the lowest misclassification rate, where the misclassification rate is a measure of the proportion of strokes that were incorrectly classified.



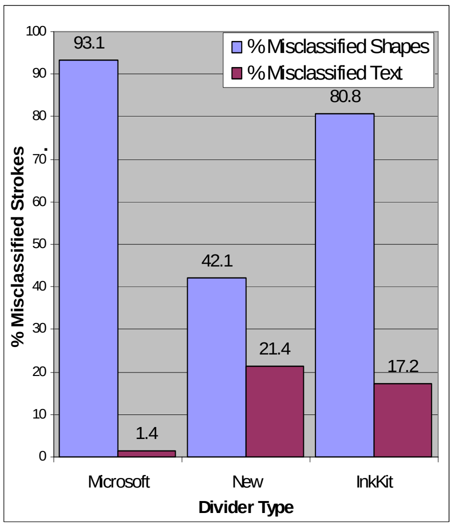


Fig 3. Percentage of misclassified shape and text strokes for each divider using the training diagram set

Fig 4. Percentage of misclassified shape and text strokes for each divider using the new diagram set

The dividers were used to divide the original training set of diagrams shown in Table 1 and also a new set of diagrams that included more complex diagrams exhibiting characteristics not considered previously. The new diagram set consisted of a directed graph, musical notes, check boxes and a crossed out word. Results were compiled based on the proportion of strokes that were correctly or incorrectly classified as text or shape strokes. The level of accuracy of each divider was then assessed from these results to ultimately determine if our new shape divider did in fact perform better.

Using the training set of diagrams, the percentage of shape and text strokes that were misclassified for each divider is shown in Figure 3. The Microsoft divider has the highest percentage of misclassified shape strokes at 75.7% and the lowest percentage of misclassified text strokes, where no text strokes were incorrectly classified at all. The 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 very high 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%, however when compared with the other dividers, InkKit's rate of misclassification for text strokes is the highest. 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 each of the three dividers are shown in Figure 4. The Microsoft divider once again has the worst rate of misclassification of shape strokes where 93.1% were incorrectly classified. It has 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 3. 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 a little above InkKit at 17.2% for text strokes. InkKit's rate of misclassification for shape strokes comes in at 80.8%. Again, all dividers show a greater degree of accuracy in classifying text strokes than shape strokes.

6. Conclusion

Eight features out of the 52 that were in the feature set were found to be significant to dividing text and shape strokes. These features were mostly measures of size and time, with one measuring curvature. The evaluation shows that overall the new shape divider was the most accurate at dividing the training set of diagrams when compared with the InkKit divider and the Microsoft divider. For the new diagram set it was considerably better at classifying shape strokes and marginally the worst at classifying text strokes. Overall, when compared with the InkKit and Microsoft dividers, the new divider was more accurate at dividing the new diagram set as well. Therefore we can conclude that the features selected can be used to improve the accuracy of division of text and shape strokes in a sketched diagram.

7. Future Work

The classification tree approach has worked well at combining the significant features together for more accurate division of strokes into character text and geometric shapes. Future work in this area could involve experimenting with other algorithms such as using Hidden Markov Models [4] for a more robust divider.

References

1. Breiman, L., et al., *Classification and Regression Trees*. 1984, New York: Chapman & Hall / CRC Press.
2. Fonseca, M.J., C.e. Pimentel, and J.A. Jorge. *CALI: An Online Scribble Recogniser for Calligraphic Interfaces*. in *AAAI Spring Symposium on Sketch Understanding*. 2002: IEEE.
3. Plimmer, B., *Using Shared Displays to Support Group Designs; A Study of the Use of Informal User Interface Designs when Learning to Program*, in *Computer Science*. 2004, University of Waikato.
4. Rabiner, L.R., *A tutorial on hidden Markov models and selected applications in speech recognition*, in *Readings in speech recognition*. 1990, Morgan Kaufmann Publishers Inc.
5. Rubine, D.H. *Specifying gestures by example*. in *Proceedings of Siggraph '91*. 1991: ACM.
6. Sezgin, T.M., T. Stahovich, and R. Davis. *Sketch based interfaces: early processing for sketch understanding*. in *Proceedings of the 2001 workshop on Perceptive user interfaces*. 2001. Orlando, Florida: ACM Press.
7. Venables, W.N. and B.D. Ripley, *Modern Applied Statistics with S*. Forth ed. 2002, New York: Springer.
8. Young, M., *InkKit: The Back End of the Generic Design Transformation Tool* 2005, University of Auckland.
9. R Development Core Team. *R: A Language and Environment for Statistical Computing*. 2006, Vienna. Austria, R Foundation for Statistical Computing.