

The Power of Automatic Feature Selection: Rubine on Steroids

Rachel Blagojevic, Samuel Hsiao-Heng Chang, Beryl Plimmer
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand

ABSTRACT

Digital ink features drive recognition engines. Intuitively, we understand that particular features are of more value for some problems than others. Likewise, inclusion of poor features may be detrimental to recognition success. Many different ink features have been proposed for ink recognition, and most work well for the context that they are employed. However given a new problem it is not clear which of the already defined features will be most useful. We have assembled and categorized a comprehensive feature library and use this with attribute selection algorithms to choose the best features for a specified problem. To verify the effectiveness of this approach the selected features are used to train a Rubine's recognizer. We show that a set of complementary features is most effective: poor features adversely affect recognition as do two or more aliases of good features. We have composed a variant of a Rubine recognizer for 3 different datasets and compared these with the Rubine's original features, a variant on this InkRubine and \$1. The results show that feature selection can significantly improve recognition rates with this simple algorithm thus verifying our hypothesis that the right combination of features for a problem is one key to recognition success.

Categories and Subject Descriptors (according to ACM CCS): I.7.5 [Document Capture]: Graphics recognition and interpretation, I.4.7 [Image Processing and Computer Vision]: Feature Measurement - *feature representation*

1. Introduction

Gesture, or single stroke, recognizers are a core component of most digital ink recognition. Even geometric based recognizers such as [PH08, YC03] employ single stroke recognition techniques to differentiate primitives such as lines and curves. Such recognizers join or fragment strokes before or after the single stroke recognition is applied. The process of joining and fragmenting strokes we therefore considered a separate problem from single stroke recognition.

Computable features of the ink stroke are at the heart of most gesture recognizers. Johnson et al [JGH*09] group the techniques for sketch recognition into hard coded recognizers, visual matching algorithms and those based on textual descriptions. Visual matching is further separated into feature based approaches and those based on graphical templates. Paulson et al [PH08] categorize recognition techniques as feature based or geometrically based approaches. Although all techniques are not explicitly regarded as feature based, they all rely on information provided by various measurements of the digital ink strokes [FPJ02, HD02, HCE*97, PH08, PF07, Rub91, SD05, SSD01, YC03], as well as specific algorithms to combine and select the appropriate features.

Many features have been proposed for many different recognition problems. Each problem, (diagram type, gesture set etc) has its own peculiarities. There are many examples of people adding and removing features from recognition

algorithms which have been designed for different problems to improve the recognition rate for their specific problem. Variations of Rubine's [Rub91] recognizer is an example of this [LNH*00, Pli04]. Trainable recognition algorithms, such as Rubine's seek to optimize to the best features and ignore poor features. However there is a computational cost for each feature; therefore reducing the number of features employed will increase the efficiency of recognition.

To explore the effects of different features we have composed a comprehensive library of ink features from previous work in sketch recognition, as well as formulating new features that we believe could measure important characteristics of strokes. The features have been carefully scrutinized to develop a taxonomy of feature types. In total there are 114 features in the library, each with code to calculate the feature.

The importance of features is well established in AI community, but has had little attention in respect of digital ink features. To explore the effect of feature selection we use attribute selection algorithms in Weka [WF05] to identify the best features for single stroke recognition in a particular context. The features are then used in Rubine's algorithm to compare recognition rates across datasets. Rubine's algorithm is popular because of its simplicity and computational efficiency. However reported recognition rates are usually around 86% [Pli04]. We will show that selecting a set of effective features, and using these in Rubine's algorithm significantly improves accuracy rates.

2. Related work

The choice of features is critical to the success of recognition, yet heuristics currently form the basis of most selections. Little rigorous analysis has been applied when identifying the features used in each recognition technique. Typically feature and algorithm selection is made heuristically [LC02, PH08, Rub91, SSD01, YC03]. Fonseca et al [FPJ02] report using percentile graphics for each possible feature which show the statistical distribution of feature values for different shape classes. Long et al [LLR*00] uses multi dimensional scaling analysis and regression to identify distinguishing features for determining the similarity of gestures.

There are many features used by these recognition systems, but the question is which ones are really significant to the recognition problem at hand? Various studies of ink features have attempted to answer this question, one for a writing drawing divider [PPG07] and the others for basic shape and gesture recognition [PRD*08, WNG*09, ZS07].

The writing-drawing divider developed in [PPG07] used a decision tree with a library of 46 potential ink features. These features were from previous work and included their own additions. The features were categorized into seven categories: pressure, time, intersections, size, curvature, tablet OS recognition (probabilities generated by the Microsoft Tablet OS recognition engine), and inter-stroke gaps (includes features measuring spatial and temporal context). It was found that features measuring inter-stroke gaps, size and curvature were important for distinguishing between writing and drawing. This was demonstrated by the accuracy of the new divider using these features in comparison to two other dividers.

Paulson et al [PRD*08] conducted a small study of geometric features versus gesture based features for basic shape recognition. Geometric features are considered as those measuring the shape of strokes whereas gesture based features concentrate on how strokes are drawn. They compiled 44 features from PaleoSketch [PH08] and Rubine's work [Rub91] (used for basic shape and gesture recognition). Two quadratic classifiers were trained using a 50/50 split of the data, where one used the full feature set and the other a selected subset. Another two quadratic classifiers were trained using 25-fold cross validation on the full feature set and the selected subset. The subset of features was chosen using a 10-fold greedy sequential forward selection technique using the quadratic classifier as a wrapper. The results showed that the quadratic classifiers using the selected feature subset performed better than those using the full feature set. The results also showed that geometric features were more successful for basic shape recognition than gesture based features. Only one gesture based feature (total rotation) was found to be optimal.

Willems et al. [WNG*09] conducted a study to evaluate the effect of different feature sets for multi-stroke gesture recognition. Three groups of features were used: one had 48 features from previous research which contains geometric features; another extended those 48 features by finding the mean and standard deviation of various stroke groups to generate a total of 758 features; the last contained features

used extensively in character recognition. They report the extended feature set improves the accuracy between 10% and 50% compared with only the 48 geometric features; and between 25% and 39% when compared with two previous recognition techniques. This demonstrates that the use of a good feature set can produce improvements in recognition accuracy.

Zhang and Sun [ZS07] conducted a small study comparing features with three algorithms. The features they tested were mostly from previous work in sketch recognition and included Rubine's features [Rub91], centroid radius, curvature, normalized curvature, compositional features (include dynamic features, and means and standard deviations of other features), speed and a modified turning function. They tested the accuracy of three algorithms; support vector machine (SVM), hidden Markov model (HMM), and a Bayesian belief network (BN); when each feature is used as input to see how successful each feature (or group of features) is at recognizing their data. The data they used for training and testing came from electrical diagrams and consisted of 10 different graphical symbols of varying degrees of complexity collected from four participants. Unfortunately the data was highly biased as participants were asked to draw symbols in specific styles. The study found that speed and turning features produced high recognition results for all algorithms. For BN the compositional features had optimal results. For SVM, curvature was very successful. The centroid radius and Rubine's features did not produce good results for any algorithms. Overall BN was found to be the most suitable algorithm for multi-stroke recognition as was able to achieve over 92% correct with only 100 samples and had the shortest training time of all.

None of the above feature studies have a feature library that is as comprehensive and wide ranging as ours. Although Willems et al [WNG*09] had a feature set of 758 features, these were simply extensions of a base set of 48 features. We have also developed a taxonomy to complement our library – organizing things into categories and naming the groups is a well proved method for providing new ways to think about the object. To our knowledge there are no comprehensive studies exploring the application of multiple feature selection algorithms to improve sketch recognizers.

3. Feature Library

The feature library consists of 114 features; the complete list is presented in appendix 1 together with a reference to its source. In order to better understand the type of things that the features are measuring we have developed the taxonomy shown in table 1. In some cases a feature reflects more than one entry in the taxonomy, for example the entropy feature is considered to be a measure of density, however it can also be a part of the divider results as it was developed as a one feature divider by Bhat et al [BH09].

In an approach derived from grounded theory [GS67], we developed the taxonomy by first grouping features measuring similar characteristics of ink. Once groups were formed, category names were assigned to each group according to the types of features that belonged to that group. By categorizing features in this way we tried not to

fit features into a particular group but form the group around the features that shared similarities.

This taxonomy helps us to gain a better understanding of the information we can gain from ink by providing a clear overview of various characteristics of ink. When we examine the subsets chosen by feature selection methods, having a taxonomy to refer to is a valuable tool. Grouping the feature subset using the taxonomy helps us gain a more intuitive understanding of the features that are significant to the problem rather than overwhelming us with each individual feature’s characteristics. With this knowledge we learn more about the types of features that are important to the problem at hand, therefore helping us to better solve the problem in the future. It may also help us identify characteristics of ink that are not measured and thus devise new features.

Table 1. Summary of stroke feature categories.







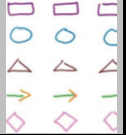
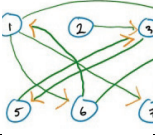
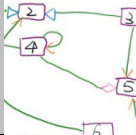
<p>1. Curvature (e.g. the line above has a greater curvature than the line below).</p> 	<p>6. Pressure (measure the pressure applied to the screen when drawing a stroke. Pressure is dependent on the capabilities of the hardware).</p>
<p>2. Density (e.g. the spiral below has a higher density of points than the line).</p> 	<p>7. Size</p> 
<p>3. Direction (this is related to the overall slope of the stroke).</p> 	<p>8. Spatial context (with sub categories: curvature, density, divider results, intersections, location and size).</p>
<p>4. Divider Results (these features provide the results of text/shape divider algorithms).</p> 	<p>9. Temporal context (with sub categories: curvature, density, divider results, length, location/distance and time/speed).</p>
<p>5. Intersections (e.g. the diagram below shows intersecting strokes).</p> 	<p>10. Time / speed (includes total, average, maximum and minimum times or speed).</p>

Table 2. Dataset descriptions

	Shape	Graph	Class
Example			
Collection method	Isolated	In-situation	In-situation
Rectangle	80	-	170
Ellipse	80	146	-
Triangle	80	-	57
Line (arrow shaft)	80	172	215
Arrowhead	80	173	96
Diamond	80	-	60
Total	480	491	598

4. Datasets

Three datasets have been used as exemplars of different types of single stroke data. Shape consists of basic shapes drawn in isolation (see examples in table 2). These are

representative of simple drawing artefacts or functional gestures. Graph and Class are drawn as diagrams where each shape has spatial and temporal relationships, and they differ in the number of shape classes. In each case data was drawn by 20 participants. They were instructed to draw each class in a single stroke, any multi-stroke classes are excluded from the dataset. The breakdown of the number of instances of each class in each dataset is shown in table 2.

5. Data Mining for Feature Selection

In effect we undertake a two step training process: the first is to select the best feature subset through the feature selection functionalities supported by Weka. However, Weka supports many feature selection methods, each performing the selection by focusing on different aspects of a feature subset. We have evaluated the performance of eight feature selection algorithms by training Rubine’s algorithm with each feature subset and observing the recognition accuracy. For our purpose Rubine has the advantage of being a simple statistical method without the sophistication to fully ignore poor features or aliases.

For each stroke in the datasets all 114 features were computed. For each dataset with features calculations, we firstly applied each feature selection method chosen from Weka [WF05]: ChiSquared, Filtered, GainRatio, InfoGain, OneR, ReliefF, Significance and SymmetricalUncert. The selected attributes are then used to train Rubine’s algorithm by gradually increasing the size of the feature subset from one to the maximum number selected. The results in figure 1 show incremental increase in accuracy, as the feature set size increases, and also shows the number of attributes where sub-optimal features begin to be selected, reflected by significant drops in accuracy. To choose a good feature selection method, we decided the priority is that it should not select sub-optimal features and can select the most important features which increase accuracy with a minimal number of features.

All algorithms apply a ranked search, which weights the features. Among these algorithms we found through testing, ReliefF gives near optimum performance without decrease if the weight is set to 0.3, which means the features of little importance are not used. The steep drops in figure 1 are because of Rubine’s sensitivity to bad features. We found it is also very sensitive to inappropriate feature combinations. For example, table 3 shows the results of different feature combinations and the accuracy of the trained classifier. The feature numbers in table 3 correspond to the full feature list in the appendix. Comparing the first two combinations, the only difference is the use of feature 1.16, which suggests that this feature is detrimental to recognition. However, combination 3 also uses feature 1.16 but gives a good result. With further analysis, we observe that the removal of either feature 2.2 or 2.5 as shown in 4 and 5 can produce good results. This shows that feature selection not only has to consider the individual importance of features, but also the relationships between them. While most other attribute evaluation algorithms assume the independence of the attributes, ReliefF considers the dependencies between the participating attributes. This gives it a strong advantage. ReliefF, when configured to use weight higher than 0.3, always produces good results. The 0.3 weighting was

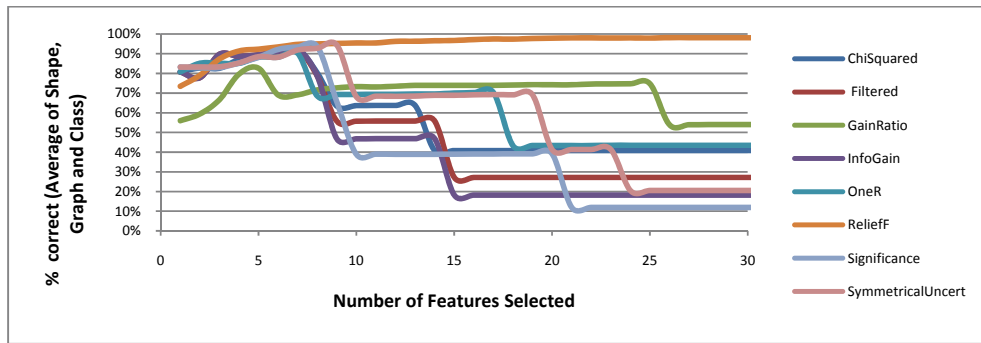


Figure 1. The accuracy of each feature selection algorithm for different feature set size

Table 3. Accuracy of different feature combinations (features in appendix 1)

	Feature combination	% Correct
1	1.12, 2.8, 7.17, 2.6, 7.7, 7.18, 7.16, 2.5, 2.2 , 1.17	97.3
2	1.12, 2.8, 7.17, 2.6, 7.7, 7.18, 7.16, 2.5, 2.2 , 1.17, 1.16	3.63
3	1.6, 1.11, 1.15, 1.3, 1.22, 2.6, 2.8, 3.1, 3.2, 3.3, 7.7, 1.16	98.6
4	1.12, 2.8, 7.17, 2.6, 7.7, 7.18, 7.16, 2.5, 1.17, 1.16	97.3
5	1.12, 2.8, 7.17, 2.6, 7.7, 7.18, 7.16, 2.2, 1.17, 1.16	97.3

Table 4. Top 20 features found with ReliefF for each dataset (features in appendix 1)

Dataset	Top 20 features
Shape	1.12, 2.8, 2.6, 1.16, 7.16, 5.1, 7.17, 7.2, 7.7, 7.13, 7.10, 1.3, 1.1, 1.9, 7.5, 1.7, 1.17, 7.11, 4.1, 1.23
Graph	1.12, 2.8, 2.6, 7.16, 1.6, 1.11, 1.18, 1.21, 1.4, 1.23, 7.7, 7.2, 4.1, 5.1, 1.13, 1.16, 7.17, 1.9, 1.19, 7.10
Class	1.12, 5.1, 2.8, 1.16, 2.6, 7.16, 7.13, 7.2, 7.7, 7.14, 7.11, 1.7, 7.17, 7.10, 1.13, 1.3, 1.17, 1.4, 10.5, 1.11

decided by trialing values between 0.1 and 0.9.

Generally the more features used the higher the accuracy. As the accuracy never decreases under our setting, we decided to select all features with a weight of more than 0.3. We also observe that approximately 20 features can guarantee high performance. The top 20 features selected using ReliefF with each dataset independently is shown in table 4. We have only listed the top 20 features as we have observed that the top 20 features produce optimal results for ReliefF as illustrated in figure 1. The features shown in bold are those that are common between all three datasets. Half of the top 20 features for each dataset are common to all subsets; however there may be further common features if we look at those ranked below the top 20.

6. Evaluation

To train the recognizers, we split each dataset in half and trained on one half and tested on the other and then reversed the sets to perform a second run. This ensures the testing process will have different participants from the training process. The three datasets shown in table 2 are used in the evaluation process.

Three trainable recognizers are used for comparison: \$1 recognizer [WWL07], the original Rubine algorithm [Rub91], and InkRubine [PF07]. Table 5 shows the evaluation results of each algorithm trained and tested with each dataset. The Data Manager Evaluator [SPB09] is used to perform the evaluations.

The attribute selected Rubine algorithm shows a much

higher average accuracy than all other algorithms tested. A z-test was conducted between the attribute selected Rubine and InkRubine, with $n = 1569$. It shows a standard error of 0.006 and a p-value of 4×10^{-15} . This shows that the attribute selected Rubine algorithm is significantly better than InkRubine and all other algorithms used as they have lower average accuracies.

Table 5. Evaluation results

Algorithm	Dataset	% Correct			
		Shape	Graph	Class	Average
Rubine (AttribSelected)		97.70	98.40	94.50	96.87
InkRubine		95.00	96.40	85.90	92.43
Rubine (Original)		85.60	97.90	80.60	88.03
\$1		85.40	94.00	84.20	87.87

7. Discussion

Many features have been proposed for sketch recognition. The taxonomy presented here clarifies the types of measures that have been used. When developing the taxonomy we used a method derived from grounded theory [GS67] where we grouped the features first into those that have similarities, and then formed categories around those groups. This method was chosen so that we could avoid having preconceived ideas of how the taxonomy should be formed. In essence we wanted a taxonomy that would fit the feature library rather than trying to fit the features into a pre-categorized system.

By looking at the taxonomy we can see that the top 20 features in table 4 are overwhelmingly from the curvature (1) and size (7) categories. There are also features from density (2), divider results (4) intersections (5) and time/speed (10). Some categories are not used at all in the top 20, and from some only one feature is used. Given the range of recognition problems that are yet to be fully investigated and the relative similarity of our datasets it is too soon to come to any conclusions on the overall usefulness of particular features. However with the use of our taxonomy we can begin to form a greater understanding of the types of features that are important for this problem, therefore promoting further development of features in these areas.

Our results show that feature selection is valuable. It can select good features and reduce the number of features required for an algorithm without compromising on accuracy. Reducing the number of features to only those

that are significant can save on computation time, which is especially important for eager recognition applications. However, feature selection should be carried out with care to ensure that accuracy is preserved when discarding features. For example when presented with a set of good features, the best features may be selected and the less important ones discarded. Yet, in many cases these less important features may work in combination with other features to produce good results. We found that the ReliefF algorithm gave consistently good results with our data sets.

The same attribute selection method can select different subsets when training with different datasets due to the nature of each dataset. Our analysis found that although features tend to have different rankings, there is a large overlap of common features between the subsets. Some consistently good features appear in all versions of our algorithm: Table 4 shows that half of the top 20 features are common to all datasets. This highlights the importance of these common features. We still find features that are unique to a particular problem, for example features 6.3 (minimum pressure) and 1.5 (sum of the angles²) only appear in the selected feature subset for the Shape dataset, although they are ranked below the top 20.

Attribute selection is beneficial for simple algorithms such as Rubine, or algorithms which consider all inputs

without a means to remove bad features. More sophisticated algorithms use voting techniques or tree generating algorithms as a mechanism for performing attribute selection themselves therefore the independent application of feature selection techniques do not result in gains in accuracy. However, because attribute selection is fast, an extra filter may still save on the computational cost of recognition.

Overall, adding good features is an important way to improve recognizer performance. With the existence of more features, calculation time may increase therefore attribute selection is a promising approach to prune these attributes and generate highly accurate algorithms.

8. Conclusion

The taxonomy of features presented here aids understanding the types of features useful for recognition problems and is likely to promote the discovery of more good quality features. Selecting appropriate features for the problem is also important. We have shown that automatic selection of those features using data mining techniques produces highly accurate results. An evaluation of our attribute selected classifiers against three other trainable recognizers shows significantly more accurate results.

Appendix 1

	Feature	Description & Origin
1	Curvature (23)	
1	# Bezier cusps	Number of bezier cusps [PPG*07]
2	# Direction Changes	Number of changes in the direction of a stroke. (New)
3	# Polyline cusps	Number of polyline cusps [PPG*07]
4	\sum angle at each point	Sum of the absolute value of the angle at each point of the stroke. [Rub91]
5	\sum (angle at each point) ²	Sum of the squared value of the angle at each point of the stroke. [Rub91]
6	Abs Curve Larg. Frag.	The total absolute curvature of the largest fragment. [BSH04]
7	Angle of bbox diagonal	Angle of the bounding box diagonal. [Rub91]
8	Avg Curvature	Average curvature (total angle / number of stroke points). [PRD*08]
9	Cos from 1st to last pt.	Cosine of the angle between the first and last point of the stroke. [Rub91]
10	Cos of initial angle	Cosine of the initial angle of the stroke. [Rub91]
11	Curviness	\sum abs value of the angle at each stroke pt below 190 threshold. [LLR*00]
12	Distance frm 1st - last pt	Distance from the 1st point of the stroke to the last point of the stroke [Rub91]
13	Least Squares Error	Orthogonal distance squared between the least squares fitted line and the stroke points / stroke length. [PRD*08, SSD01]
14	Max Curvature	Maximum curvature of the stroke. [PRD*08]
15	NDDE	Normalised distance between direction extremes. [PH08]
16	Number of Fragments	# of fragments in a stroke (fragmented according it's to corners) [BSH04]
17	Openness	Distance from 1st -last pt of the stroke / size of the stroke's b. box. [LLR*00]
18	Overtracing	Total angle / 2π . [PH08]
19	Sin from 1st to last pt	Sine of the angle between the first and last point of the stroke. [Rub91]
20	Sin of initial angle	Sine of the initial angle of the stroke. [Rub91]
21	Total angle	Total angle traversed by the stroke. [Rub91]
22	Total Angle & Lgth Ratio	Total angle / stroke length. [LLR*00]
23	Total Angle Ratio	Total angle / \sum angle at each point . [LLR*00]
2	Density (8)	
1	Amount of ink inside	Amt of ink inside the strokes b. box (count the # of pts inside b. box) [You05]
2	Density 1	Stroke length / distance between first & last point. [LLR*00]
3	Density 2	Stroke length / area of bounding box. [LLR*00]
4	Entropy	Stroke information density [BH09]
5	Length Ratio	Cumulative distance between stroke points/ length from stroke start to end point. Adapted from [Rub91]
6	Length:Perimeter ratio	Stroke length / perimeter of the stroke's convex hull. [FPJ02]
7	Point Ratio	# of points in the stroke's convex hull / # of points in the stroke. [LC02]

8	Total length/bounding box diagonal length	Length of the stroke divided by the length of the bounding box diagonal. Adapted from [Rub91]
3	Direction (4)	
1	DCR	Maximum change in direction / average change in direction. [PH08]
2	Direction	Direction of the stroke (eigenvector of the largest eigen value) [BSH04]
3	Eigen value ratio	The largest eigen value/ smallest eigen value. [BSH04]
4	Largest Frag. Direction	Direction of largest fragment (eigenvector of the largest eigen value).[BSH04]
4	Divider Results (2)	
1	Divider Result	Results of our text/shape divider on the current stroke. [PPG*07]
2	Tablet OS text prob.	Tablet OS text recognizer probability of the stroke being text.[Mic05]
5	Intersections (3)	
1	# Endpt self intersections	# of self intersections at the endpts of the stroke. Adapted from [Qin05]
2	# Other self intersections	# of self intersections that are not at the stroke's endpt. Adapted frm [Qin05]
3	# Self intersections	Number of points where the stroke intersects itself. Adapted from [Qin05]
6	Pressure (4)	
1	Average pressure	Mean average pressure of the stroke. Adapted from [NSS*02]
2	Max pressure	Maximum pressure value for the stroke. Adapted from [NSS*02]
3	Min pressure	Minimum pressure value for the stroke. Adapted from [NSS*02]
4	# Pressure minima	# of minima in pressure values for the stroke. [PPG*07]
7	Size (19)	
1	Arc Fit Radius	The radius of an arc fitted to the stroke. [PRD*08]
2	Aspect	$ 45\pi/180 - \text{angle of the bounding box diagonal} $. [LLR*00]
3	Bounding box area	Area of the bounding box of the stroke. Adapted from [FPJ02, HD02]
4	B. box diagonal length	Length of the bounding box diagonal line. [Rub91]
5	Bounding box height	Height of the bounding box of the stroke. Adapted from [FPJ02, HD02]
6	Bounding box width	Width of the bounding box of the stroke. Adapted from [FPJ02, HD02]
7	Convex hull area ratio	Ratio of area of convex hull to area of the enclosing rect of the stroke.[FPJ02]
8	Enclosing Rect. ratio	Ratio of strokes enclosing rectangle width to height. [FPJ02]
9	Largest Frag. Length	Arc length of the stroke's largest fragment. [BSH04]
10	Length	Total length of the stroke. [Rub91]
11	Log Area	Log of the stroke's bounding box area. [LLR*00]
12	Log Aspect	Log of the aspect feature. [LLR*00]
13	Log Length	Log of the total length of the stroke. [LLR*00, MFN93]
14	Log Longest Side Rect	Log of the length of the longest side of the stroke's bounding box. [MFN93]
15	Long Side of Enclosing Rect of Largest Frag.	The longest length of the largest fragment's enclosing rectangle. [BSH04]
16	Perimeter Efficiency	$2\sqrt{(\pi \text{ stroke's convex hull area}) / \text{stroke's convex hull perimeter}}$. [LC02]
17	Perimeter to area	Ratio of perimeter to area of the stroke's convex hull. [FPJ02]
18	Thinness ratio	$\text{Perimeter}^2 \text{ of stroke's convex hull} / \text{area of stroke's convex hull}$ [FPJ02]
19	Width to height ratio	Ratio of the stroke's bounding box width to height. Adapted from [FPJ02]
8	Spatial Context	
8.1	Curvature (2)	
1	Avg Curvature of Close End Pt Strokes	Average curvature (using total angle) of strokes close at end points to current stroke. [AWJ*07]
2	Avg Curvature of Close Strokes	Average curvature (using total angle) of strokes close to current stroke. [AWJ*07]
8.2	Density (2)	
1	Avg Density of Close End Pt Strokes	Average density (stroke length / bounding box diagonal length) of strokes close at end points to the current stroke. [AWJ*07]
2	Avg Density of Close Strokes	Average density (stroke length / bounding box diagonal length) of strokes close to the current stroke. [AWJ*07]
8.3	Divider Results (1)	
1	Divider Closest Stroke	Results of our text/shape divider for the closest stroke to the current [AWJ*07]
8.4	Intersections (5)	
1	# Other intersections	Number of points of intersection of the current stroke with other strokes (excluding self intersections). Adapted from [CSK*02]
2	# Other strokes intersecting	Number of other strokes that intersect the current stroke (excluding itself). Adapted from [FPJ02, HD02]
3	# Strokes Vert Overlapping	The number of strokes vertically overlapping the current stroke. [BSP09]
4	Total # intersections	Total # of intersections (includes self intersections). Adapted from [CSK*02]
5	Total # strokes intersecting	Number of strokes that intersect the current stroke (including itself) [FPJ02, HD02]
8.5	Location (9)	

1	# Close End Pt Strokes	# of strokes whose endpts are close to endpt of the current stroke. [AWJ*07]
2	# Close Strokes	The number of close strokes to the current stroke. [AWJ*07]
3	# Strokes Contained	# of strokes contained in the current stroke. [BSP09]
4	# Strokes Horiz Close	#of strokes horizontally close to curr stroke. [BSP09]
5	# Strokes On Same Horiz Plane	The number of strokes on the same horizontal plane as the current stroke. [BSP09]
6	# Vertically Close	The number of strokes vertically close to the current stroke. [AWJ*07]
7	Is Contained	If a stroke is contained by another stroke. (New)
8	Smallest Dist Btw Strokes from End Pt	The smallest distance to another stroke from the current stroke's end point. (New)
9	Smallest Dist Btw Strokes from Start Pt	The smallest distance to another stroke from the current stroke's start point. (New)
8.6 Size (4)		
1	# Strokes Similar Height	# strokes of similar height to current stroke. [BSP09]
2	Avg Length of Close End Pt Strokes	Average length of strokes close at endpoints to the current stroke. [AWJ*07]
3	Avg Length of Close Strokes	Average length of strokes close to the current stroke. [AWJ*07]
4	Length of Closest Stroke	The length of the closest stroke to the current stroke where the closest stroke is found by measuring distance between the middle of the b. box. (New)
9 Temporal Context		
9.1 Curvature (2)		
1	Curv. of Next Stroke	Total angle of next stroke. Adapted from [Rub91]
2	Curv. of Prev. Stroke	Total angle of previous stroke. Adapted from [Rub91]
9.2 Density (2)		
1	Density of Next Stroke	Length of the next stroke divided by the length of the next stroke's bounding box diagonal. Adapted from [Rub91]
2	Density of Previous Stroke	Length of the previous stroke divided by the length of the previous stroke's bounding box diagonal. Adapted from [Rub91]
9.3 Divider Results (2)		
1	Next Stroke Divider	Results of our text/shape divider for the next stroke. [PPG*07]
2	Previous Stroke Divider	Results of our text/shape divider for the previous stroke. [PPG*07]
9.4 Length (2)		
1	Length of Next Stroke	Total length of next stroke. [AWJ*07]
2	Length of Prev. Stroke	Total length of previous stroke. [AWJ*07]
9.5 Location/Distance (6)		
1	Distance from last stroke	Distance between current stroke and previous stroke. Adapted from [You05]
2	Distance to next stroke	Distance between current stroke and next stroke. Adapted from [You05]
3	X Diff between strokes	Difference in X co-ordinate between current stroke and next. [BSH04]
4	X Start point diff	Difference in starting X coordinates of current stroke to next stroke. [BSH04]
5	Y Diff between strokes	Difference in Y co-ordinate between current stroke and next. [BSH04]
6	Y Start point diff	Difference in starting Y coordinates of current stroke to next stroke. [BSH04]
9.6 Time/Speed (8)		
1	Log start time from prev	Log of time from start of previous stroke to start of current stroke. [BSH04]
2	Log start time to next	Log of time from start of current stroke to start of the next stroke. [BSH04]
3	Log time diff from prev	Log of the time between the current and previous stroke. [BSH04]
4	Log time diff to next	Log of the time between the current stroke and the next stroke. [BSH04]
5	Speed from last stroke	Speed (distance/time) between current stroke and previous stroke. [PPG*07]
6	Speed to next stroke	Speed (distance/time) between current stroke and next stroke. [PPG*07]
7	Time from last stroke	The time between current stroke and previous stroke. [PPG*07]
8	Time till next stroke	The time between current stroke and next stroke. [PPG*07]
10. Time / Speed (6)		
1	# Speed minima	#of extreme minima in the speed values for the stroke. Adapted from [SSD01]
2	Average Speed	Mean average speed when drawing the stroke. Adapted from [Rub91]
3	Max speed	Maximum speed when drawing the stroke. Adapted from [Rub91]
4	Max speed squared	Maximum speed of the stroke squared. [Rub91]
5	Min speed	Minimum speed when drawing the stroke. Adapted from [Rub91]
6	Total duration	Total duration of the stroke from pen up to pen down. [Rub91]

Acknowledgements

Thanks to Associate Professor Eibe Frank for expert advice on WEKA and data mining techniques. This research is partly funded by the Royal Society of New

Zealand, Marsden Fund. The program code and data files used in this project are available to other researchers at http://www.cs.auckland.ac.nz/research/hci/digital_ink/ink_recognition/index.shtml

References

- [AWJ*07] AO X., WANG X., JIANG Y. DAI G.: Structuring and Manipulating Hand-Sketched Diagrams. Sketch Based Interfaces and Modeling (SBIM '07) (2007),
- [BH09] BHAT A. HAMMOND T.: Using Entropy to Distinguish Shape Versus Text in Hand-Drawn Diagrams. International Joint Conference on Artificial Intelligence (IJCAI '09) (2009), 1395-1400.
- [BSH] BISHOP C. M., SVENSEN M. HINTON G. E. Distinguishing Text from Graphics in On-Line Handwritten Ink. *Frontiers in Handwriting Recognition* (2004), 142-147
- [BSP09] BLAGOJEVIC R., SCHMIEDER P. PLIMMER B.: Towards a Toolkit for the Development and Evaluation of Sketch Recognition Techniques. *Intelligent User Interfaces (IUI'09) Sketch Recognition Workshop* (2009),
- [CSK*02] CALHOUN C., STAHOVICH T. F., KURTOGLU T. KARA L. B.: Recognising Multi-Stroke Symbols. *AAAI Spring Symposium on Sketch Understanding* (2002), 15-23.
- [Mic05] MICROSOFT CORPORATION. Microsoft Windows XP Tablet PC Edition Software Development Kit, <http://www.microsoft.com/downloads/details.aspx?familyid=B46D4B83-A821-40BC-AA85-C9EE3D6E9699&displaylang=en>
- [FPJ02] FONSECA M. J., PIMENTEL C. E. JORGE J. A.: CALI: An Online Scribble Recogniser for Calligraphic Interfaces. *AAAI Spring Symposium on Sketch Understanding* (2002), 51-58
- [GS67] GLASER B. G. STRAUSS A. L. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Publishing Company (1967).
- [HD02] HAMMOND T. DAVIS R.: Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. *2002 AAAI Spring Symposium on Sketch Understanding* (2002).
- [HCE*97] HUTTON G., CRIPPS M., ELLIMAN D., HIGGINS C. A.: A Strategy for On-line Interpretation of Sketched Engineering Drawings. *International Conference on Document Analysis and Recognition* (1997), 771-775.
- [JGH*09] JOHNSON G., GROSS M. D., HONG J. DO E. Y.-L.: Computational Support for Sketching in Design: A Review. *Foundations and Trends in Human-Computer Interaction*, 1, 2 (2009), pp. 1-93.
- [LC02] LEUNG W. H. CHEN T.: User-independent retrieval of free-form hand-drawn sketches. *Acoustics, Speech, and Signal Processing (ICASSP '02)*, (2002), 2, 2029-2032.
- [LNH*00] LIN J., NEWMAN M. W., HONG J. I. LANDAY J. A.: Denim: Finding a tighter fit between tools and practice for web design. *Chi 2000* (2000), 510-517.
- [LLR*00] LONG A. C., LANDAY J. A., ROWE L. A. MICHELS J.: Visual similarity of pen gestures. *CHI 2000* (2000), 2, 360-367.
- [MFN93] MACHII K., FUKUSHIMA H. NAKAGAWA M.: On-line text/drawings segmentation of handwritten patterns. *Document Analysis and Recognition* (1993), 710-713.
- [NSS*02] NAKAI M., SUDO T., SHIMODAIRA H. SAGAYAMA S.: Pen Pressure Features for Writer-Independent On-Line Handwriting Recognition Based on Substroke HMM. *Pattern Recognition (ICPR'02) Volume 3 - Volume 3* (2002), 30220.
- [PPG*07] PATEL R., PLIMMER B., GRUNDY J. IHAKA R.: Ink Features for Diagram Recognition. *Sketch-Based Interfaces and Modeling* (2007), 131-138.
- [PH08] PAULSON B. HAMMOND T.: PaleoSketch: Accurate Primitive Sketch Recognition and Beautification. *Intelligent User Interfaces (IUI '08)* (2008), 1-10.
- [PRD*08] PAULSON B., RAJAN P., DAVALOS P., GUTIERREZ-OSUNA R. HAMMOND T.: What!?! No Rubine Features?: Using Geometric-based Features to Produce Normalized Confidence Values for Sketch Recognition VL/HCC Workshop: Sketch Tools for Diagramming (2008), 57-63.
- [Pli04] PLIMMER B. Using Shared Displays to Support Group Designs; A Study of the Use of Informal User Interface Designs when Learning to Program. University of Waikato, PhD, (2004).
- [PF07] PLIMMER B. FREEMAN I.: A Toolkit Approach to Sketched Diagram Recognition. *HCI* (2007), 1, 205-213.
- [Qin05] QIN S.: Intelligent Classification of Sketch Strokes. *EUROCON* (2005), 1374-1377.
- [Rub91] RUBINE D. H.: Specifying gestures by example. *Proceedings of Siggraph '91* (1991), 329-337.
- [SPB09] SCHMIEDER P., PLIMMER B. BLAGOJEVIC R.: Automatic Evaluation of Sketch Recognition. *Sketch Based Interfaces and Modelling* (2009), 85-92.
- [SD05] SEZGIN T. M. DAVIS R.: HMM-based efficient sketch recognition. *Intelligent user interfaces (IUI'05)* (2005), 281-283.
- [SSD01] SEZGIN T. M., STAHOVICH T. DAVIS R.: Sketch based interfaces: early processing for sketch understanding. *Perceptive user interfaces* (2001), 1-8.
- [WNG*09] WILLEMS D., NIELS R., GERVEN M. V. VUURPIJL L.: Iconic and multi-stroke gesture recognition. *Pattern Recogn.*, 12, 42 (2009), pp. 3303-3312.
- [WF05] WITTEN I. H. FRANK E. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann (2005).
- [WWL07] WOBROCK J. O., WILSON A. D. LI Y. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In *User interface software and technology* (2007), 159-168.
- [You05] YOUNG M. InkKit: The Back End of the Generic Design Transformation Tool. University of Auckland, BEng, (2005).
- [YC03] YU B. CAI S.: A domain-independent system for sketch recognition. *Computer graphics and interactive techniques in Australasia and South East Asia* (2003), 141-146.
- [ZS07] ZHANG L. SUN Z.: An experimental comparison of machine learning for adaptive sketch recognition. *Applied Mathematics and Computation*, 2, 185 (2007), pp. 1138-1148.