Grouping Strokes into Shapes in Hand-Drawn Diagrams

Eric J. Peterson and **Thomas F. Stahovich**
Mechanical Engineering Department
University of California
Riverside, California 92521
epeterson@engr.ucr.edu, stahov@engr.ucr.edu

Eric Doi and **Christine Alvarado**
Computer Science Department
Harvey Mudd College
Claremont, California 91711
eric_doi@hmc.edu, alvarado@cs.hmc.edu

Abstract

Objects in freely-drawn sketches often have no spatial or temporal separation, making object recognition difficult. We present a two-step stroke-grouping algorithm that first classifies individual strokes according to the type of object to which they belong, then groups strokes with like classifications into clusters representing individual objects. The first step facilitates clustering by naturally separating the strokes, and both steps fluidly integrate spatial and temporal information. Our approach to grouping is unique in its formulation as an efficient classification task rather than, for example, an expensive search task. Our single-stroke classifier performs at least as well as existing single-stroke classifiers on text vs. non-text classification, and we present the first three-way single-stroke classification results. Our stroke grouping results are the first reported of their kind; our grouping algorithm correctly groups between 86% and 91% of the ink in diagrams from two domains, with between 69% and 79% of shapes being perfectly clustered.

Introduction

One of the most difficult challenges in sketch understanding is clustering the strokes into distinct objects. Often there are no clear spatial or temporal boundaries between the objects in a freely-drawn sketch, and the primary clue that a group of strokes is supposed to be grouped together is that they form a meaningful shape. This is the inherent chicken-and-egg problem for sketch recognition — shapes cannot be recognized until their strokes have been grouped together, but the strokes cannot be grouped until the shapes have been recognized.

The clustering problem is so challenging that many existing recognition systems avoid it by placing constraints on the way users draw. For example, some systems require the user to provide explicit cues, such as button clicks or pauses, to demarcate each object (e.g., (Hse and Newton 2005)); others require each symbol to be drawn with a single stroke (e.g., (Rubine 1991)) or a temporally contiguous sequence of strokes (e.g., (Gennari, Kara, and Stahovich 2005)). While these constraints aid recognition, they do not generally match the way people naturally draw (Alvarado and Lazzareschi 2007).

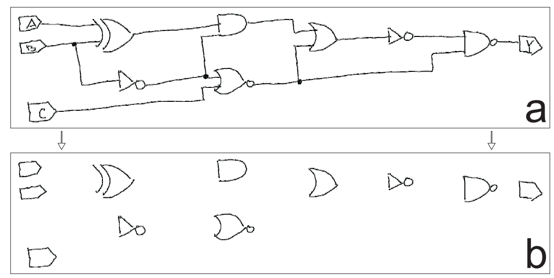

Figure 1: (a) A digital logic sketch. (b) The strokes in the sketch that are classified as gates.

To solve the problem of simultaneous grouping and recognition, Kara *et al.*'s (2004) mark-group-recognize technique relies on "marker symbols" — symbols that can be accurately and inexpensively extracted from a continuous stream of pen strokes, and that tend to separate the remaining symbols. While this approach is efficient, it is limited to domains that have effective markers.

Other recent work has focused on the problem of single-stroke classification. Qi *et al.* (2005) present a method for using conditional random fields to classify strokes in organizational chart diagrams as either connectors or boxes. Addressing a similar problem, Bishop *et al.* (2004), Patel *et al.* (2007), and Bhat and Hammond (2009) present methods that integrate shape and temporal information for classifying individual strokes as either text or drawing strokes. Wang *et al.* (2007) improve on Bishop *et al.*'s method.

The goal of most previous single-stroke classification techniques is to identify the text strokes so they can be sent to a character recognizer, while the shape strokes (i.e., strokes comprising graphic objects) are left ungrouped. Our approach goes further, grouping the shape strokes as well. We aim to achieve accurate grouping in domains for which marker symbols do not exist and single-stroke classification is more involved. We present a two-stage clustering algorithm that first classifies pen strokes into different classes of objects, and then groups strokes with like classifications into clusters representing individual objects. Figure 1 illustrates our approach. In the first step of processing (Figure 1a), individual strokes are classified as belonging to text, gate, or wire objects. This classification spatially and temporally separates individual objects of the same class — as in Figure 1b, which shows only strokes classified as gates — making the strokes easier to cluster.

Our work makes two significant contributions over previous approaches to stroke-level classification and grouping. First, while previous approaches were applied to only two-way classification (usually text vs. non-text), our approach is highly accurate on three-way classification of text and two different types of graphics. In addition, our two-way classification accuracy is as good as the best previously reported results. Second, the separation between objects that results from our single-stroke classification technique enables our novel formulation of the grouping problem as an inexpensive classification task.

Related Work

In addition to the work described above, a growing body of free-sketch recognition research involves simultaneous stroke grouping and symbol recognition. Some grouping techniques rely directly on geometric properties of the strokes. For example, Saund *et al.* (2003) decompose a sketch into sequences of contiguous line segments corresponding to line art, and "blobs" of dense ink corresponding to text. They use Gestalt principles to group these objects into larger structures. The approach is computationally expensive for dense diagrams, and is intended to produce groupings suitable for interactive manipulation rather than object recognition.

Other techniques tightly integrate the process of stroke grouping and symbol recognition by searching over all possible stroke clusterings, using the results of recognition to guide the search. Shilman and Viola (2004) use A* search to generate candidate groupings and then evaluate them with a recognizer to distinguish valid objects from meaningless combinations of pen strokes. In earlier work, Shilman *et al.* (2002) use a manually-coded visual grammar to guide the search for stroke clusterings, while Alvarado and Davis (2005) use the probabilities produced by dynamically constructed Bayesian networks.

All of the above search-based approaches rely on recognition to help prune the exponential space of possible groupings, and none scale well to complex drawings, especially if the search gets "off track" early in the process.

Approach

Our approach consists of two steps: (1) classifying single strokes into two or more different classes and (2) clustering strokes of the same class into individual objects.

Single-Stroke Classification

Our goal at this stage is to classify strokes into general categories to facilitate stroke grouping. We use a feature-based machine-learning approach with a standard classification algorithm and a feature set that extends the set presented in Patel *et al.* (2007).

Our classifier uses AdaBoost with decision trees and is trained using WEKA (Hall et al. 2009). Specifically the classifier is AdaBoostM1 using 10 iterations, a seed of 1, no resampling, and a weight threshold of 100. The base classifier is a pruned c4.5 decision tree using a confidence value of 0.25, and the minimum number of instances in a leaf is 2. AdaBoost was chosen because it performed better than other methods that were tested, namely multi-layer perceptrons and c4.5 decision trees.

The main contribution of this paper is not any specific feature, rather our complete stroke- classification/stroke-grouping approach. However, we find that the specific combination of features we used outperforms previous systems (e.g., (Patel et al. 2007)) so we present our entire feature set here for completeness.

Each pen stroke is captured as a series of sampled points containing position and time information, from pen-down to pen-up. For classification we extract 27 features that characterize the stroke's size and location, its shape, the drawing kinematics, and the relationships between it and the other strokes in the sketch.

Both the shape of a stroke and the context in which it appears are critical to classification. For example, in digital-circuit sketches small curved strokes are not likely to be wires, but a straight stroke might be part of a wire or the back of an 'AND' gate, depending on context. To incorporate both shape and context, our features represent both the intrinsic properties of the stroke (e.g., length and curvature), and its relationships with the surrounding strokes (e.g., intersections and temporal gaps).

The first important property of a stroke is its size, which is represented by four features. *Bounding Box Width*, *Height* and *Area* are properties of the minimum, coordinate-aligned bounding box of the stroke, while *Arc Length* is the total length of the stroke measured as a sum of the distance between consecutive points. These features are normalized by their average values in the sketch, allowing the classifier to learn the importance of relative stroke size in classification.

One insight not captured by the features in (Patel et al. 2007) is that, in many domains, particular kinds of objects often appear in preferred locations on the drawing canvas (defined here as the bounding box of the entire sketch). For example, diagrams may be drawn in the center of the canvas, with text near the periphery. This phenomenon is captured by two positional features. *Distance to Left/Right* is the minimum distance between the stroke and the closer of the left or right edge of the canvas, divided by the width of the canvas. *Distance to Top/Bottom*, which is the location relative to the top or bottom of the canvas, is defined analogously.

Eight features describe the shape of a stroke. The first three describe its topological properties. *EndPtRatio* measures the degree to which the stroke forms a closed path. It is defined as the Euclidean distance between the endpoints of the stroke divided by the arc length. *Self Enclosing* is a binary form of the *EndPtRatio*. If *EndPtRatio* is less than a threshold, T (we use a value of 0.15), the value of *Self Enclosing* is one, otherwise it is zero. *Self Intersections* is the number of times the stroke intersects itself.

The next four features describe the stroke's curvature. The curvature, θ_i, at point i is defined as the angle between the segment connecting point $i-1$ to point i, and the segment connecting point i to point $i+1$.The four curvature features are obtained by summing various functions of the curvature at each point along the stroke. *Sum of the (signed) Curvature* represents the total turning angle of the stroke, where turns in one direction cancel turns in the other. *Sum of the Absolute Value of the Curvature* provides a measure of how much the curve "wiggles," or deviates from a straight line. *Sum of the Squared Curvature* emphasizes corners, or points of high curvature. Conversely, *Sum of the Square Root of Curvature* emphasizes points of low curvature. The first three of these are from (Rubine 1991), while the last is of our own design.

Finally, *Ink Density* measures the compactness of the stroke. In previous work this was particularly useful in helping to distinguish wires from components in analog circuits (Gennari, Kara, and Stahovich 2005). It is defined as the ratio of the square of the arc length to the area of the minimum coordinate-aligned bounding box. Arc length is squared so that it scales in the same way as bounding box area.

Pen speed can provide important information about the classification of a stroke. For example, people might draw text strokes faster than diagram strokes, or wires faster than gates. The drawing kinematics are represented in terms of four speed-based features: the *Average Pen Speed*, the *Maximum* and *Minimum* instantaneous pen speeds, and the *Difference Between Maximum and Minimum* instantaneous pen speeds. Pen speed is near zero at the two endpoints of a stroke, so when computing the minimum, a few points at each end are ignored. Each speed-based feature value is normalized by the average stroke speed in the sketch. The final kinematic feature is the *Time to Draw* the stroke.

The remainder of the features characterize the geometric and temporal relationships the stroke has with other strokes in the sketch. The first four of these measure the number of different types of intersections the stroke has with other strokes: endpoint-to-endpoint (*'LL'*s), midpoint-to-midpoint (*'XX'*s), midpoint-to-endpoint (*'XL'*s), and endpoint-to-midpoint (*'LX'*s). We have found that this distinction between intersections involving endpoints and midpoints, which is not reported elsewhere in the literature, is important for accurate stroke classification. We use a distance tolerance to catch cases where strokes nearly intersect. In effect, the strokes are extended at each end by a small amount. A simple linear tolerance is too generous for long strokes, and too tight for short ones. Instead, our tolerance is derived from the sketch average arc length, L_{avg}, as follows:

$$L_{tol} = min(L_{avg}, \frac{L_i + L_{avg}}{2}) * T \qquad (1)$$

where L_i is the arc length of the stroke to be extended and T is the same threshold used for computing *SelfEnclosing* (a value of 0.15 works well in practice). This formula produces a proportionally larger tolerance for short strokes, and a proportionally smaller tolerance for long ones. Before extending a stroke, the small "hooks" at the endpoints are removed, using the algorithm described in (LaViola 2005), so that the direction at the endpoints is meaningful. If the intersection point lies within distance L_{tol} of the actual endpoint of the stroke, it is considered an endpoint intersection. Otherwise, it is considered a midpoint intersection. The case in which two extended strokes do not actually intersect, but their endpoints are within a distance L_{tol} of one another, is still considered an endpoint intersection.

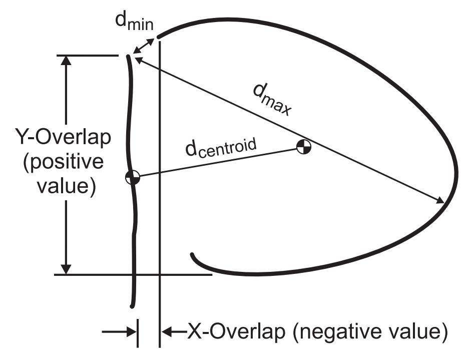

Figure 2: Features used by clustering classifier.

Our feature set includes two novel features that characterize higher-level geometric relationships. The binary *Closed Path* feature indicates whether or not the stroke belongs to some set of strokes that connect to each other via *'LL'* intersections to form a closed path. The binary *Inside Path* feature indicates whether or not the stroke is inside the minimum coordinate-aligned bounding box of some closed path. Using a bounding box to test for *Inside Path* can result in false positives, but is inexpensive and has worked adequately for our purposes.

The final two features for single-stroke classification capture temporal relationships. *Time to Previous* is the elapsed time between the end of the previous stroke and the start of the current one. *Time to Next* is defined analogously.

Stroke Grouping

Classifying the individual pen strokes reduces the complexity of stroke grouping by decomposing the problem into smaller, easier problems, one for each class. However, even for the strokes in a single class, brute force grouping techniques, such as attempting to recognize all combinations of strokes, are still too expensive for interactive systems. Instead, we use a classifier to determine if each pair of strokes of the same class should be joined to form a cluster. If a stroke is joined with another stroke that is already part of a cluster, all of those strokes become a single cluster. We consider two different classification methods for stroke joining, one based on simple thresholds, the other based on inductive learning techniques. Both methods cluster strokes well when used in conjunction with our single-stroke classification step.

Our threshold grouping classifier joins two strokes if the minimum distance between them (d_{min} in Figure 2) is less than T_{JD} *and* the elapsed time between them is less than T_{JT}. We obtained suitable values for these thresholds via a user-holdout parameter search. For digital circuits, the best value of T_{JD} is 200 himetric units, and the best value of T_{JT} ranged from 7.0 to 10.0 sec. For family trees the best values of T_{JD} and T_{JT} are 10 pixels (about 240 himetric units) and 1.5 sec.

Our more sophisticated grouping classifier uses AdaBoost with decision trees, with the same learning parameters used for single-stroke classification. Here, pairs of strokes are classified into three categories: "don't join," "far join," and "near join." "Don't join" describes a pair of strokes that belong to different clusters. “Far join” describes pairs that belong to the same cluster, but are sufficiently far apart that they are connected via intermediate strokes via “chaining”. “Near join” describes the remaining pairs of strokes that belong in the same cluster. For example, if a circle were added to the right side of the AND gate in Figure 2 to form a NAND gate, the back side of the gate and the circle would be a far join, while the back and front sides of the gate, as well as the front side and the circle would be near joins. The distinction between near and far joins helps the program distinguish between distant strokes within a single cluster and strokes that are distant because they belong to different clusters, thereby resulting in substantially higher grouping accuracy.

When training the grouping classifier, a pair of strokes from a cluster is labeled as a near join if the minimum distance (d_{min}) between them is less than the near join threshold:

$$d_{NJ} = max\{T_{JD}, (1+T)*max\{SMD_A, SMD_B\}\} \quad (2)$$

where T_{JD} is the distance threshold used with the threshold grouping classifier and T is the threshold for computing stroke intersections. SMD_A and SMD_B are the Shape Minimum Distance for the first and second stroke of the pair, respectively, defined as the minimum distance from that stroke to any other stroke in the cluster.

The 13 features used by our more sophisticated grouping classifier include one temporal feature and 12 spatial features, five of which are illustrated in Figure 2. The temporal feature is the time delay between the two strokes. The spatial features measure various distances between the strokes. *X-overlap* and *Y-overlap* are the length of the intersection between the projections of the two strokes onto the x-axis and y-axis, respectively. Their values are negative if the projections do not intersect. d_{max} and d_{min} are the maximum and minimum distances, respectively, between the two strokes. d_{minLL} is the minimum distance between an endpoint of one stroke and an endpoint of the other. d_{minXL} is the minimum distance between an endpoint of one stroke and any point of the other. $d_{centroid}$ is the distance between the centroids of the strokes.

$Closeness_A$ and $Closeness_B$ compare the distance between the strokes to the distances between them and their nearest neighbors of the same class. If A is the minimum distance between the first stroke of the pair and any other stroke of the same class and B is the analogous distance for the second stroke in the pair, $Closeness_A$ is computed as:

$$Closeness_A = \frac{A + k}{d_{min} + k} \quad (3)$$

where k is a constant offset to avoid division by zero, and $Closeness_B$ is computed analogously. $Ratio_{LL}$ and $Ratio_{XL}$ give an indication of whether or not the strokes are closest to each other at their endpoints. For example, $Ratio_{LL}$ is the ratio of the minimum distance between the strokes and the minimum distance between the stroke's endpoints:

$$Ratio_{LL} = \frac{d_{min} + k}{d_{minLL} + k} \quad (4)$$

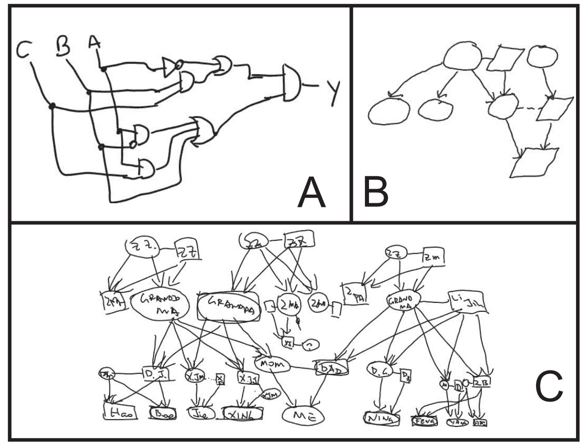

Figure 3: (A) Complex circuit diagram. (B) Simple family tree (FT) diagram. (C) Complex FT diagram.

where k is again a constant offset. $Ratio_{XL}$ is computed analogously.

Computing these pairwise features is an $O(n^2)$ process. However, the features are computed incrementally as each new stroke is drawn. Typically the features are computed far faster than the rate of drawing.

Data Sets

We tested both our single-stroke and clustering classifiers on freely-drawn sketches in two different domains: digital circuits and family trees. We collected 8 digital circuit sketches from each of 24 students at the University of California, Riverside and Harvey Mudd College for a total of 192 sketches. Half of these were copied from a picture of a circuit, while the rest were synthesized from a logical equation. Half of the sketches were drawn on a Tablet PC while the other half were drawn using a digitizing pen on paper. We balanced the order of the copy and synthesize tasks across users. Our family tree data is from the EtchaSketches corpus[1], consisting of a total of 27 sketches from 9 users, all drawn on a Tablet PC (sketches containing fewer than 5 strokes, or those which are subsets of other sketches, were not used). In all cases, users drew freely and received no recognition feedback.

Figures 1 and 3 show some simple and complex digital-circuit and family tree diagrams from the data. On average, each circuit sketch contains 51 strokes (median of 49) and each family tree sketch contains 61 strokes (median of 38). The distribution of strokes across all of the circuit diagrams is 42% wire, 14% text, and 44% gate. For family tree diagrams, the distribution is 37% link (lines or arrows), 37% text, and 26% people (boxes or ellipses).

Results

All results in the following section were obtained using user-holdout-out-cross-validation. The final accuracy is averaged across all users.

In order to compare our single-stroke classification method to existing techniques we performed two-way classification of text vs. non-text. We then compared the per-

[1] http://rationale.csail.mit.edu/ETCHASketches/

	Class		Classified As: Text	Classified As: NonText	Accuracy
Digital Circuits	Actual - Ours	Text	1255	142	89.8%
		NonText	133	8272	98.4%
				TOTAL	97.2%
	Actual - MS	Text	963	434	68.9%
		NonText	3152	5253	62.5%
				TOTAL	63.4%
	Actual - Ent.	Text	0	1397	0.0%
		NonText	0	8405	85.8%
				TOTAL	85.8%
Family Trees	Actual - Ours	Text	576	66	89.7%
		NonText	97	921	90.5%
				TOTAL	90.2%
	Actual - MS	Text	617	0	100.0%
		NonText	410	633	60.7%
				TOTAL	75.3%
	Actual - Ent.	Text	505	112	81.8%
		NonText	36	1007	96.5%
				TOTAL	91.1%

Table 1: Results of text vs. NonText. MS = Microsoft's Ink-Analyzer, Ent. = Entropy method.

	Class		Classified As			Accuracy
Dig. Circuits			Gate	Text	Wire	
	Actual	Gate	3977	119	180	93.0%
		Text	97	1289	11	92.3%
		Wire	160	24	3945	95.5%
					TOTAL	94.0%
Fam. Trees	Class		Classified As			Accuracy
			People	Text	Link	
	Actual	People	333	30	44	81.8%
		Text	22	577	43	89.9%
		Link	22	56	533	87.2%
					TOTAL	86.9%

Table 2: Results of three-way, single-stroke classification.

formance of our technique to that of a recent technique by Bhat *et al.* (2009), and Microsoft's InkAnalyzer — a state-of-the-art commercial algorithm. In the digital-circuit domain, our method achieved an overall accuracy of 97.2%, while the InkAnalyzer's accuracy was 63.4% and Bhat's entropy method reverted to a naive classifier which classified all strokes as non-text for an accuracy of 85.8%. For the family tree diagrams, our method achieved 90.2% accuracy, while InkAnalyzer achieved 75.3%, and the entropy method achieved 91.1%. A list of all results for two-way classification can be found in Table 1.

We also evaluated the performance of our single-stroke classifier on three-way classification. For the digital-circuit data the classifier achieved 94.0% accuracy, and for the family tree data it achieved 87.5% accuracy. The complete results are presented in Table 2.

We use two metrics to measure the accuracy of our grouping method. The first measure is the percentage of the cluster's ink (by arc length) that was correctly grouped, as well as the percentage of ink that was erroneously added to the cluster (i.e., extra strokes). These are computed on a per

Dig Circ Class	Ink: Avg / Shape Correct	Ink: Avg / Shape Extra	Shapes: X Errors or Less 0	1	2
Gate	90.6%	6.2%	72.9%	92.5%	97.1%
Text	94.2%	4.1%	86.4%	97.5%	99.5%
Wire	79.6%	10.1%	59.3%	81.8%	89.8%
Overall	86.3%	7.6%	69.2%	88.6%	94.2%
Fam Tree Class	**Ink: Avg / Shape Correct**	**Ink: Avg / Shape Extra**	**Shapes: X Errors or Less 0**	**1**	**2**
People	93.3%	2.1%	84.3%	94.2%	99.0%
Text	80.7%	0.2%	55.8%	75.8%	85.8%
Link	79.1%	18.0%	58.1%	86.7%	97.4%
Overall	84.5%	9.5%	67.4%	87.8%	96.2%

Table 3: Thresholded grouping classifier.

cluster basis, and then averaged across all shapes. These accuracy results are listed in the second and third columns of Tables 3 and 4. The second measure of accuracy is the percentage of clusters that have no more than a given number (X) of erroneous strokes; these results are listed in the last three columns of Tables 3 and 4. Consider, for example, a shape comprised of three strokes, A, B, and C, with arc lengths of 100, 200, and 300 pixels, respectively. If strokes B and C are grouped, but A is left out, and an additional stroke D (with length 150 pixels) is erroneously grouped with the cluster, the accuracies for this shape would be: 83.3% ink found, 25.0% extra ink, two errors – one missing and one extra stroke.

Table 3 presents the grouping results for our simple grouping classifier that uses the T_{JD} and T_{JT} thresholds. This classifier accurately groups shapes that have sufficient separation, however it performs poorly when the shapes overlap (e.g., wires). The overall accuracies for digital circuits with this approach are 86.3% ink found, 7.6% extra ink, and 69.8% of shapes grouped with zero errors. For family tree diagrams the accuracies are 84.5% ink found, 9.5% extra ink, 67.4% of shapes grouped with zero errors.

The results for our more sophisticated grouping classifier are presented in Table 4. Compared to the simple grouping classifier, this classifier gives slightly better results for gate and label shapes in digital-circuit sketches, and performs much better on wires. The overall accuracies for digital-circuits are 91.4% ink found, 5.3% extra ink, and 79.5% perfect clusters. The overall accuracy for family tree diagrams is 86.0% ink found, 7.7% extra ink, and 69.6% perfect clusters.

To test the effectiveness of the more sophisticated grouping classifier isolated from the errors of the single-stroke classifier, we tested the grouper using the correct class for each stroke. The accuracy for digital circuits increased to 96.7% ink found, 1.7% extra ink, and 93.3% of the shapes perfectly grouped. For the family tree domain the accuracies are 94.3% ink found, 4.3% extra ink, 88.1% of shapes perfectly grouped.

Discussion

Our single-stroke classification technique for text vs. non-text performed as well as or better than previous methods. In direct comparison to the entropy method described in (Bhat and Hammond 2009) and the commercial classifier

Dig Circ Class	Ink: Avg / Shape		Shapes: X Errors or Less		
	Correct	Extra	0	1	2
Gate	91.0%	4.5%	76.7%	93.2%	98.2%
Text	94.5%	6.7%	84.0%	96.7%	99.3%
Wire	90.5%	5.4%	79.9%	89.8%	94.2%
Overall	91.4%	5.3%	79.5%	92.3%	96.6%

Fam Tree Class	Ink: Avg / Shape		Shapes: X Errors or Less		
	Correct	Extra	0	1	2
People	89.8%	3.3%	80.9%	93.5%	97.6%
Text	80.9%	9.9%	50.5%	69.2%	78.5%
Link	85.2%	9.2%	68.5%	90.4%	99.0%
Overall	86.0%	7.7%	69.6%	88.5%	95.5%

Table 4: More sophisticated grouping classifier.

used by Microsoft's InkAnalyzer, our classifier performed significantly better in the digital-circuit domain, while the entropy method provided slightly better results in the family-tree domain. We also report higher accuracy than Patel *et al.*'s (2007) approximately 70% accuracy, and similar accuracy to Bishop *et al.*'s (2004) approximately 95% accuracy, and Qi *et al.*'s (2005) approximately 96% accuracy. Additionally, we report the first results for three-way classification of strokes, a task for which our approach achieved high accuracy.

The separation created by our single-stroke classifier allows even simple grouping methods to effectively cluster shapes. Using our more sophisticated grouping algorithm improves grouping accuracy in most cases, particularly for shapes with complex interactions, such as wires which can intersect one another. While most previous grouping techniques rely on search and recognition, we use classification of pairs of strokes along with chaining of pairs to form complete shapes.

Our method is domain-flexible, and can be applied to a new domain without additional coding. Applying our method to a new domain simply requires examples of each class of pen stroke to train the single-stroke classifier, and examples of each kind of grouping pair (don't join, near join, and far join) to train the grouping classifier. We have demonstrated our method in two distinct domains, digital circuits and family trees. The approach did worse for the latter domain, but this is likely due to a lack of training data: there are only 27 family tree diagrams compared with 192 circuit sketches. Another possibility is the difference in complexity between domains. Many of the family-tree sketches had significantly more strokes and looked "messier" than the digital circuits; see Figure 3 for examples.

While we have a total of 27 features for single-stroke classification, we can achieve accuracy that is only a few percentage points worse by using just four features: *Closed Path*, *Bounding Box Width*, *Distance to Left/Right*, and *Sum of (signed) Curvature*. These features were identified using WEKA's attribute selection functionality on the data for three-way classification of digital-circuit sketches. This indicates that much of the information necessary for classification is contained in a small subset of the features, but all of the features are needed to achieve maximum accuracy.

Conclusion

Grouping strokes in freely-drawn sketches is so challenging that few recognition systems attempt it. Our work is a significant step toward solving this important problem. We have shown that separating pen strokes into different classes can make the grouping process much easier and more effective. The separation is achieved by an accurate three-way single-stroke classifier, the first of its kind reported. Further we have demonstrated a new and efficient method for grouping. The method is novel in that it is based on classification rather than search.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. 0729422 & 0735695.

References

Alvarado, C., and Davis, R. 2005. Dynamically constructed Bayes nets for multi-domain sketch understanding. In *Proc. IJCAI*.

Alvarado, C., and Lazzareschi, M. 2007. Properties of real-world digital logic diagrams. In *Proc. of PLT*, 12.

Bhat, A., and Hammond, T. 2009. Using entropy to identify shape and text in hand-drawn diagrams. In *IJCAI*.

Bishop, C. M.; Svensen, M.; and Hinton, G. E. 2004. Distinguishing text from graphics in on-line handwritten ink. In *Proc. of the Int. Workshop on FHR*, 142–147.

Gennari, L.; Kara, L. B.; and Stahovich, T. F. 2005. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers and Graphics* 29(4).

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.* 11(1).

Hse, H., and Newton, A. R. 2005. Recognition and beautification of multi-stroke symbols in digital ink. *C&G* 29(4).

Kara, L. B., and Stahovich, T. F. 2004. Hierarchical parsing and recognition of hand-sketched diagrams. In *UIST*.

LaViola, J. J. 2005. *Mathematical sketching: A new approach to creating and exploring dynamic illustrations*. Ph.D. Dissertation, Brown University.

Patel, R.; Plimmer, B.; Grundy, J.; and Ihaka, R. 2007. Ink features for diagram recognition. In *Proc. of SBIM*.

Qi, Y.; Szummer, M.; and Minka, T. P. 2005. Diagram structure recognition by bayesian conditional random fields. In *Proc. of CVPR*.

Rubine, D. 1991. Specifying gestures by example. *Computer Graphics* 25:329–337.

Saund, E., and Lank, E. 2003. Stylus input and editing without prior selection of mode. In *Proc. of UIST*.

Shilman, M., and Viola, P. 2004. Spatial recognition and grouping of text and graphics. In *Proc. of SBIM*.

Shilman, M.; Pasula, H.; Russell, S.; and Newton, R. 2002. Statistical visual language models for ink parsing. In *AAAI Spring Symposium on Sketch Understanding*.

Wang, X.; Biswas, M.; and Raghupathy, S. 2007. Addressing class distribution issues of the drawing vs. writing classification in an ink stroke sequence. In *Proc. of SBIM*.