EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2004)
John F. Hughes and Joaquim A. Jorge (Editors)

Spatial Recognition and Grouping of Text and Graphics

Michael Shilman Paul Viola
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{shilman,viola}@microsoft.com

ABSTRACT

We present a framework for simultaneous grouping and recognition of shapes and symbols in free-form ink diagrams. The approach is completely spatial, that is it does not require any ordering on the strokes. It also does not place any constraint on the relative placement of the shapes or symbols. Initially each of the strokes on the page is linked in a proximity graph. A discriminative classifier is used to classify connected subgraphs as either making up one of the known symbols or perhaps as an invalid combination of strokes (e.g. including strokes from two different symbols). This classifier combines the rendered image of the strokes with stroke features such as curvature and endpoints. A small subset of very efficient features is selected, yielding an extremely fast classifier. An A-star search algorithm over connected subsets of the proximity graph is used to simultaneously find the optimal segmentation and recognition of all the strokes on the page. Experiments demonstrate that the system can achieve 97% segmentation/recognition accuracy on a cross-validated shape dataset from 19 different writers.

Categories and Subject Descriptors (according to ACM CCS): I.5.4 [Document Capture]: Graphics recognition and interepretation).

1. Introduction

Sketched shape recognition is a classic problem in pen user interfaces. Augmenting a sketched shape with its symbolic meaning can enable numerous features including smart editing, beautification, and interactive simulation of visual languages [Gross94, LM95, AD01, KS04].

In this paper we present an integrated, accurate, and efficient method for recognizing and grouping sketched symbols. Our approach applies to both hand-sketched shapes and printed handwritten text, and even heterogeneous mixtures of the two.

1.1 Previous Work

The problem of recognizing sketched drawings can be divided into two parts: grouping strokes into sets, and recognizing what symbol a set of stroke represents.

Previous research has proposed numerous shape recognition strategies including a wide variety of different features and classifiers. Some strategies emphasize invariance to changes in scale and rotation [HN04]. Others require few examples to train [Rub92, KS04, VD04]. Others are able to cope with dashed sketches and overstrikes [FPJ02].

There are also many approaches to grouping ink strokes for recognition. Some systems are designed with the constraint that the user must draw shapes with a single stroke [Rub92]. Some systems use a timeout: when the user does not sketch for a pre-specified time, the system will group the last set of strokes into a shape to be recognized. Some systems use hand-tuned heuristics to group shapes [KS04]. Many handwriting systems require the users to finish writing one shape before beginning on the next one, and then perform an optimization over the sequence of strokes to find the grouping that maximizes some heuristic or statistical score [TSW90].

In work that is most closely related to ours, Mahoney and Fromherz [MF01] have constructed a system that uses finds subgraphs of strokes that satisfy heuristically-specified constraints. They suggest that their approach should work well for sketches that are defined by the structural relationships between strokes, but may not be well-suited for sketches that are defined by the curve shape of the strokes.

We have presented an initial version of this work for the purpose of recognizing and grouping handwritten character strokes in mathematical equations and diagrams [SVC04]. This paper extends the work to flowcharts and mixtures of text and graphics. For this work we have developed a more powerful classification scheme and an improved search strategy for discovering the optimal grouping.

© The Eurographics Association 2004.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

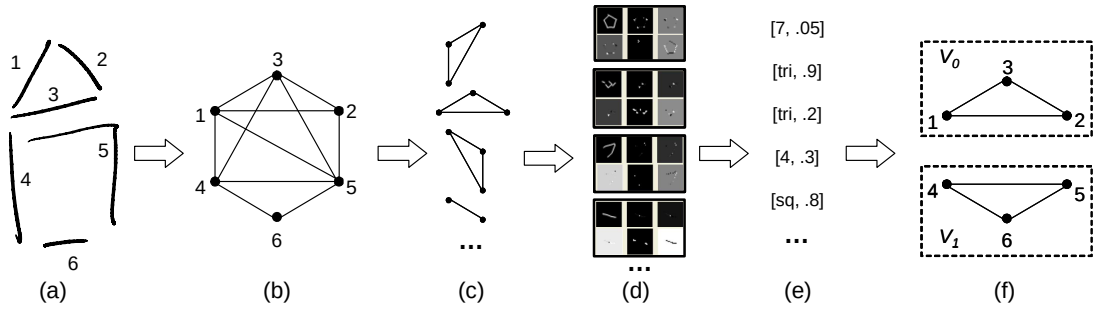

Figure 1. *An overview of our approach. (a) A user sketch containing several strokes. (b) A neighborhood graph of the strokes in the sketch. (c) Connected subsets of the neighborhood graph of up to a fixed size K. (d) Rendered images of the subsets that are passed to an AdaBoost recognizer. (e) Results from the classifier include a symbol hypothesis and a score. (f) An optimization partitions the graph to jointly maximize the classifier scores.*

1.2 Spatial Grouping and Recognition

In this paper, we present an efficient, purely spatial approach to simultaneously group and recognize handwritten symbols on a page. Our approach is an optimization over a large space of possible groupings in which each grouping is evaluated by a recognizer (Figure 1). This is in contrast to approaches where grouping and recognition are carried out as separate steps (e.g. systems with a separate layout analysis step).

In this approach the recognizer carries the burden of distinguishing good groupings from bad groupings and also must assign correct labels to good groupings. This sort of recognizer must evaluate quickly in order to process the large number of possible stroke groupings for a page of ink in a reasonable time.

Given such a recognizer, there are several benefits to this factoring of the problem. Improving the accuracy or performance of the system is simply a function of improving the accuracy or performance of the recognizer. Introducing new features to the system, such as rotation- or scale-invariance is simply a matter of changing the recognizer, rather than changing both the recognizer and the layout analysis.

Perhaps most significantly, it enables our system to be nearly entirely learned from examples rather than relying on hand-coded heuristics. This last point bears repeating: ours is a monolithic system which once developed, requires no hand constructed geometric features. All thresholds and parameters are learned automatically from a training set of examples.

Our system operates in the following manner. As a preprocessing step, it first builds a neighborhood graph of the ink in which nodes correspond to strokes, and edges are added when strokes are in close proximity to one another. Given this graph, we iterate efficiently over connected sets of nodes in the graph using dynamic programming and fast hashing on collections of nodes. For each set of nodes of up to size *K*, we perform a discriminative recognition on the set. This allows us to incorporate non-local information that rules out spurious answers that might result from a generative model. We use dynamic programming to optimize over the space of possible explanations. The resulting system achieves high accuracy rates without any language model, places no stroke ordering requirements on the user, and places no constraints on the way in which symbols must be laid out on the page.

We first describe the search-based optimization that we perform over different groupings of strokes and recognition alternatives. We then describe the classifier (AdaBoost) and its features (the Viola-Jones image filters) that we use to evaluate each stroke group. We then evaluate our work on a publicly-available database of sketched shapes and a heterogeneous mixture of shapes, arrows, and text in the form of a flowchart.

2. Search-Based Optimization

We approach shape recognition and grouping as an optimization problem. In other words, in the space of all possible groupings of strokes on the page all possible labelings of those groupings, there is a best grouping and labeling according to a cost function.

The cost of a grouping and labeling is a function of the costs of each of its constituents.

$$C(\{V_i\}) = \Phi(R(V_1), R(V_2), \ldots, R(V_N)) \quad [1]$$

In Equation 1, each V_i is a subset of the vertices which form a partition of the page (we use the terms *strokes* and *vertices* interchangeably), R is the best recognition result for that set of vertices, the function Φ is a combination cost (such as *sum*, *max*, or *average*), and C represents the overall cost of a particular grouping V_i.

Of course, the number of possible groupings is combinatorial in the number of vertices, so it would be prohibitively expensive to compute all of the combinations. Therefore, we constrain the possible groupings in the following ways:

1. We construct a neighborhood graph (Figure 1(b)) in which vertices that are close to each other on the page are connected. We say that a grouping V_i is only valid if its vertices are connected in the neighborhood graph.

© The Eurographics Association 2004.

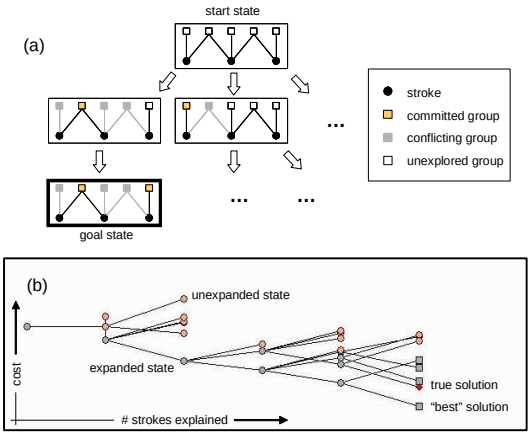

Figure 3. *A* Search. (a) A toy search example showing the hypotheses in each state. (b) Visualization of the search for a small example which yielded the incorrect "best" solution. For purposes of example we let the search run until it hit the true solution.*

2. We restrict the size of each subset V_i in the graph to be less than a constant K (Figure 1(c)). For instance, in our data we have not seen symbols of greater than 8 strokes, so we assert that it is reasonable to place an upper bound on the size of a subset.

Given these restrictions, we have used two optimization methods to solve for the best solution. The first is dynamic programming, as we describe in [SVC04]. In this paper we present an improvement based on A* search.

2.1 A* Search

A typical search problem can be defined as a state space in which each state has a cost, operators to transition between states, and a test to see whether a state is a goal state. A* is a search technique that uses a heuristic underestimate to the goal from each state to prune away parts of the search space that cannot possibly result in an optimal solution [RN95]. The quality of the estimate impacts the efficiency of the search: a weak underestimate can result in a slow search and an aggressive underestimate that is not a true underestimate can result in a suboptimal solution (also known as an *inadmissible* heuristic).

Our search space is a set of partial groupings of strokes (Figure 2). Our initial states are all individual groups of strokes of up to size K. Each group of strokes can combine with another group to form an aggregate partial grouping provided that the two groups do not share any common strokes. Such combinations are the operators of the search. Finally, a state is a goal state if it explains all of the strokes on the page.

As in Equation 1 and [SVC04], the cost of a grouping is the combination cost of its sub-groups. The underestimate to the goal from a partial grouping is a function of the *best* explanations of the parents of the strokes unexplained by that grouping. In particular, if a partial grouping explains the first N strokes of a drawing, the underestimate cost for each unexplained stroke is $R(V^*)/|V^*|$ where V^* is the best partial explanation that explains that stroke (note this partial explanation may explain multiple strokes, so we divide the cost across the strokes). This is a true *underestimate* because in the best case those best interpretations can all be taken. It is not a true *estimate* because some of those interpretations may conflict, in which case they cannot all be taken.

Figure 2. *Classifier input. The candidate stroke (square) is shown in red and its context strokes is shown in blue. Both candidate and context are rendered into 29x29 images. The first image shows the original ink, the other images depict stroke features such as endpoints, curvature, and self-intersections.*

3. Recognition

The recognizer utilized in the optimization described above is based is a novel application of AdaBoost. The primary input to the classifier is a rendered image of the strokes that comprise the hypothetical shape. Since we do not yet know the segmentation of the strokes, the strokes passed to the classifier may not make up a shape at all (we call these groupings garbage).

The framework used is most closely related to the work of Viola and Jones, who constructed a real-time face detection system using a boosted collection of simple and efficient features [VJ01]. The Viola-Jones approach is distinguished because it classifies images extremely rapidly and reliably. We chose this approach both because of its speed and because it is easily extensible to include additional feature information.

We have generalized Viola-Jones in two ways. First, our classification problem is multi-class. Second, we have added additional input features to the 29x29 input images. These additional features are computed directly from the on-line stroke information and include curvature, orientation, and end-point information (Figure 3). While we believe that this information could be computed directly from the image, this information is only currently available from on-line systems.

The observations that are sent to the classifier are sums over the pixel values in rectangular regions of the image.

While in practice we do not generate all possible rectangles at all possible locations in a 29x29 image, we have generated 5280 rectangles per image. Because there are 12 input images example, the classifier rceives 63,360 observations per training example! Over the course of its training, the classifier automatically determines which of these observations are relevant to the classification problem and selects a small subset of these observations which should actually be made in practice. The mechanics of this process is outside the scope of this paper, but is discussed in [SS98].

In this paper we have further extended the learning framework to include boosted decision trees. In our previous work we boosted "stumps" or depth one decision trees. In other words, each boosted classifier previously reasoned about a single threshold (i.e. a depth 1 decision tree), whereas our current boosted classifiers reason about small conjunctions of thresholds on different rectangles.

While stumps yielded good results when the number of classes is small, it didn't work well for problems with a larger number of similar symbols/characters. In these experiments we have used depth 3 decision trees. These more general decision trees are more powerful classifiers, capable of modeling complex dependencies between features. The main risk in using a decision tree is that it may overfit the training data. We have found that be limiting the depth of the tree to 3 there has been no tendency to overfit in our experiments.

4. Evaluation

We have evaluated this work on the publicly-available HHReco sketched shape database [HN04], containing 7791 multi-stroke examples over 13 shape classes, collected from 19 different users. On randomly-generated shape scenes in which shapes are placed near one another in random patterns (Figure 4(a)), we achieved 97% accuracy for both grouping and recognition simultaneously, and over 99% accuracy for grouping alone (using 80/20 train/test split).

We also evaluated our approach on a more complex set of randomly synthesized flowcharts. Each flowchart was generated from the shapes { square, ellipse, diamond, hexagon, pentagon }, the connectors { ↔, →, − }, and the digits { 0 – 9 }, in which four nodes were synthesized in random non-overlapping locations with randomly sampled edges between them, and four digits were contained in each node (Figure 4(b)). On this problem we achieved an 85% grouping/recognition accuracy, and a 90% grouping accuracy.

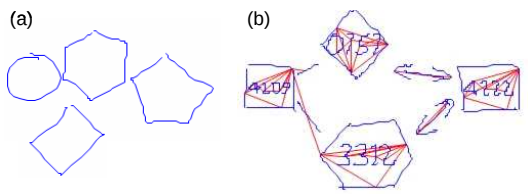

Figure 4. *Test examples. (a) A randomly generated collection of shapes. (b) A synthetic flowchart composed of shapes, arrows, and characters collected from real users. We superimpose its neighborhood graph to give a feel for the size of the problem.*

Data set	DT Depth	False Pos	False Neg	FP Grp	FN Grp	Truth Count
shape	1	16	30	6	11	600
shape	3	25	18	9	7	600
flow	1	80	78	18	14	1200
flow	3	76	58	21	8	1200
flow2	1	459	430	107	134	2400
flow2	3	330	348	107	134	2400

Table 1. *Grouping and recognition results. Data sets include shapes (shape), flowcharts (flow), and flowcharts with digit labels (flow2). False pos/neg refer to errors in grouping and recognition. FP/FN Grp refer to grouping errors only. Truth refers to the number of symbols in the true grouping.*

Our initial results are shown in Table 1. They indicate that small-depth boosted decision trees are roughly equivalent to stumps for sketch recognition problems with a small number of classes. However, as the number of classes grow, the decision trees show a modest improvement over the stumps. These results also show that, at least for our system and data set, digit recognition is substantially more difficult than shape recognition–the error rate on flowcharts without digits was much lower than the flwochart with digits. Furthermore, the errors that did occur in the flowcharts with digits were mostly errors on the digits.

We are able to process files such as the one shown in Figure 4(a) in appromimately .1s on a 1.7GHz Tablet PC. Larger examples such as Figure 4(b) currently take approximately 8s to process. Nearly 90% of the time is spent rendering the examples to bitmaps (Figure 1(d)) since there are an order of magnitude more symbol candidates than there are symbols in the file. We have spent almost no time optimizing our implementation and believe that there is plenty of room for improvement.

The experiment had several limitations. As in the shape experiment, both the training and test data were synthesized. However, we used the HHReco shape data and synthesized it with arrows and digits collected from users in our lab. We were careful to keep the test and training users separate to show that our approach is able to generalize, and to keep a one-to-one mapping between shape writers and digit/arrow writers.

5. Discussion

We have presented a method for grouping and recognized sketched shapes that is efficient and accurate, yet relies solely on spatial information and is completely example-based. We were not surprised that the approach appears to be equally applicable to sketched shapes, arrows, and printed handwritten characters.

This work has several implications to the field of sketch recognition and sketch-based user interfaces in general.

Firstly, we envision a recognizer that achieves high accuracy for shapes, symbols, arrows, and so on, and places no constraints on the user in terms of order or specific page layout of those symbols. With such a recognizer available off the shelf, the designer of a sketch-based user interface would not have to make compromises on which symbols to include or how the use should enter those symbols, and could instead focus on defining the right symbol set for the problem, an appropriate correction user interface, and so on.

The approach we present in this paper is the basis for such a solution. Because the approach is entirely based on machine learning and requires no hand-coded heuristics, it can be easily retargeted to different domains, as we have done originally for mathematics and now for flowcharts. There has been work on recognizers that can operate on small sets of training examples, and recognizers that adapt to user correction. The approach described in this paper does not achieve these goals. However, we have noted that our graph optimization ultimately takes the form of the underlying recognizer, so a variant of this approach may eventually achieve this. In the meantime, we are happy with a combination of efficiency and high accuracy.

From a recognition standpoint, we note that the system relies on very few parameters. The only significant ones are the maximum number of strokes in each character (in our case 6), and a proximity threshold for building a neighborhood graph. Furthermore, it relies entirely on the concise cost function for its answer and so improvements in accuracy can be achieved through improvements of the cost function and the underlying recognizer, without needing to modify any of the rest of the algorithm.

The biggest limitation of the approach is on symbols that are actually deformable templates. Real arrows in diagrams need not be straight and can snake arbitrarily. We have not evaluated our approach on deformable arrows for flow charts, but we expect that it will be a challenge. We are therefore interested in establishing a more formal relationship between this work and ongoing work in so-called structural recognition of sketches.

References

[HN04] HSE, H., AND NEWTON, A. R: Sketched Symbol Recognition using Zernike Moments. *International Conference on Pattern Recognition*, Aug. 2004, Cambridge, UK.

[SVC04] SHILMAN, M., VIOLA, P., AND CHELLAPILLA, K.: Recognition and Grouping of Handwritten Text in Diagrams and Equations. *IWFHR 2004*, September. 2004, Tokyo, Japan

[CSK02] CALHOUN, C., STAHOVICH, T.F., KURTOGLU, T. AND KARA, L.B., Recognizing Multi-Stroke Symbols. *2002 AAAI Spring Symposium - Sketch Understanding*, (Palo Alto CA, 2002), AAAI Press, 15-23.

[LM95] LANDAY, J., AND MYERS, B. Interactive Sketching for the Early Stages of User Interface Design. *Proc. of CHI '95: Human Factors in Computing Systems*, Denver, CO, May 1995, pp. 43-50.

[TSW90] TAPPERT, C., SUEN, C. , WAKAHARA, T. The State of the Art in Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(8): 787-808 (1990)

[MF01] MAHONEY, J., FROMHERZ, M. Interpreting Sloppy Stick Figures by Graph Rectification and Constraint-based Matching. *Fourth IAPR Int. Workshop on Graphics Recognition*, Kingston, Ontario, Canada, Sept. 2001

[RN95] RUSSELL, S. AND NORVIG, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[SS98] R. SCHAPIRE, Y. SINGER. Improved Boosting Algorithms using Confidence-Rated Predictions. *COLT 1998*: 80-91

[Gross94] M. GROSS. Stretch-A-Sketch, a dynamic diagrammer. *IEEE Symposium on Visual Languages* (VL '94), 1994.

[VJ01] P. VIOLA, M. JONES. Robust Real-Time Face Detection. *ICCV 2001*: 747

[AD01] ALVARADO, C., DAVIS, R. Preserving the freedom of paper in a computer-based sketch tool. P*roceedings of HCI International*, 2001

[VD04] VESELOVA, O. DAVIS, R. Perceptually Based Learning of Shape Descriptions for Sketch Recognition. *The Nineteenth National Conference on Artifical Intelligence* (AAAI-04), July 2004.

[KS04] KARA, L., STAHOVICH, T. Sim-U-Sketch: A Sketch-Based Interface for Simulink, *AVI 2004*, pp 354-357.

[FPJ02] FONSECA, M.J., PIMENTEL, C. AND JORGE, J.A., CALI: An Online Scribble Recognizer for Calligraphic Interfaces. *2002 AAAI Spring Symposium on Sketch Understanding*, (Palo Alto CA, 2002), AAAI Press, 51-58.

[Rub92] RUBINE, D.: Specifying Gestures by Example. SIGGRAPH '91, 25 (4). 329-337