HMM-Based Efficient Sketch Recognition[1]

Tevfik Metin Sezgin and Randall Davis
MIT Computer Science and Artificial Intelligence Laboratory
77 Massachusetts Ave, 32-235
Cambridge MA 02139
{mtsezgin,davis}@csail.mit.edu

ABSTRACT

Current sketch recognition systems treat sketches as images or a collection of strokes, rather than viewing sketching as an interactive and incremental process. We show how viewing sketching as an interactive process allows us to recognize sketches using Hidden Markov Models. We report results of a user study indicating that in certain domains people draw objects using consistent stroke orderings. We show how this consistency, when present, can be used to perform sketch recognition efficiently. This novel approach enables us to have polynomial time algorithms for sketch recognition and segmentation, unlike conventional methods with exponential complexity. [1]

Categories and Subject Descriptors: I.5.5 Pattern Recognition: Interactive Systems

General Terms: Algorithms.

Keywords: Sketch recognition, Enabling input technologies, Interpretation of user input, Intelligent user interfaces

1. INTRODUCTION

Sketching is a natural input modality of increasing interest [5]. Our goal is to build systems that can recognize complex objects, and as a step towards this goal, we propose a novel approach to symbolic sketch recognition that takes advantage of the incremental and interactive nature of sketching.

By a sketch, we mean messy, informal, hand-done drawings (e.g., Fig. 1). Specifically we are interested in recognizing sketches of objects that can be described and recognized using structural methods. This is the class of sketches that has been the focus of the sketch recognition community [2, 3, 8, 7]. Sketch recognition involves grouping strokes that constitute the same object (segmentation) and determining the object class for each group (recognition).

We view sketching as an incremental process, defining a sketch as a sequence of strokes. Strokes are captured using a digitizer, preserving the drawing order. We use the term ***sketching style*** to refer to a user's preferred – although not necessarily conscious – stroke ordering when drawing an object.

[1]More details on this work is available at:
http://www.rationale.csail.mit.edu/publications.shtml

IUI'05, January 10–13, 2005, San Diego, California, USA.
ACM 1-58113-894-6/05/0001.

1.1 The problem

Current sketching systems are indifferent to who is using the system, employing the same recognition routines for all users. The framework we introduce in this paper provides a mechanism for capturing an individual's preferred stroke ordering during sketching, and uses it for efficient segmentation and recognition of hand-sketched objects in polynomial time.

2. APPROACH

Our approach is differentiated from work on images by the need to deal with the dynamic character of sketches (i.e., sketches are constructed one stroke at a time). We believe stroke ordering information can be a powerful aid to recognition because we believe that people have strong biases in the way they sketch. We ran a user study to assess the degree to which people have sketching styles.

2.1 User study

In our study, users were asked to sketch various icons, diagrams and scenes such as emoticons expressing happy, sad, surprised, angry faces, scenes with stick-figures, military Course of Action Diagram symbols, finite state machines, Unified Modeling Language (UML) diagrams, and digital circuit diagrams.

We asked 10 subjects to sketch three sketches from each of the six domains, collecting a total of 180 sketches. Our analysis of the collected data revealed that in these domains people indeed sketch objects in a highly stylized fashion using predictable stroke orderings. These orderings may be different for each individual, but persist across sketches for each individual. In order to capitalize on this structure we used Hidden Markov Models (HMMs) to model different sketching styles.

3. HMM-BASED RECOGNITION

After each stroke is added, we encode the new scene as a *sequence of observations*. Recognition and segmentation is achieved by aligning to this observation sequence a series of HMMs, while maximizing the likelihood of the whole scene. Each HMM models the drawing order of a single class of objects.

Using HMMs gives us the ability to learn compact models of how single objects are drawn and allows us to have a probabilistic score for computing how well individual object models match a particular subsequence of an observation sequence encoding a complete scene. Recognition and segmentation is achieved by combining matching scores from individual HMMs in a mathematically sound way. Our framework supports multiple object classes, multiple drawing orders, and can handle variations in the lengths of encodings for individual objects.

A review of HMMs is outside the scope of this paper and we refer the reader to a comprehensive tutorial by Rabiner [1]. We adopt the same notation in this paper.

3.1 Modeling with HMMs

3.1.1 Encoding

Sketches must be encoded to generate observation sequences for recognition. We encode strokes using the Early Sketch Processing Toolkit described in [2], which converts strokes into geometric primitives. We encode the output of the toolkit to convert sketches into discrete observation sequences using a codebook of 13 symbols; four to encode lines (positively/negatively sloped, horizontal/vertical), three to encode ovals (circles, horizontal/vertical ovals), four to encode polylines (with 2, 3, 4, and 5+ edges) one to encode complex approximations (i.e., mixture of curves and lines), and one symbol to denote two consecutive intersecting strokes.

3.1.2 Segmentation and recognition

Assume we have n object classes. Encodings of training data for class i may have varying lengths, so let $L_i = \{l_{i1}, l_{i2}, ...l_{ik}\}$ be the distinct encoding lengths for class i. We train one HMM per object class using encodings of individual objects as examples.

For isolated object recognition, we compute $P(O|\lambda_i)$ for each model λ_i using the Forward procedure with the observation sequence O generated by encoding the isolated object. λ_i with the highest likelihood gives us the object class. Unfortunately, isolated object recognition requires the input sketch to be presegmented, which is usually not the case, and segmentation is itself a hard problem.

Interpretation of a complex scene requires generating hypotheses for the whole scene. That is, it requires assigning models to subsequences of the entire observation sequence that encodes the scene, such that the interpretations assigned to individual groups don't conflict with each other and also they maximize the likelihood of the whole scene (thus generating a *globally coherent interpretation*).

Hypothesis generation should be efficient, so combinatoric approaches are ruled out. The fact that individual HMMs return probability values makes it easy to define the objective for this stage, namely, choose interpretations that maximize the probability corresponding to the entire scene. This is an optimization problem that we solve using dynamic programming implemented in the form of a shortest path problem.

The shortest path in a graph $G(V, E)$ that we generate gives us the segmentation. We then perform classification as described above. Segmentation and recognition begins by building the graph $G(V, E)$ such that V consists of $|O|$ vertices, one per observation, and a special vertex v_f denoting the end of observations.

Starting with O_1, for each observation symbol O_s, we take a substring $O_{s,s+k}$ for $k \in L_i$. Next we compute the loglikelihood for the observation given the current model, $log(P(O_{s,s+k}|\lambda_i))$, and add a directed edge from vertex v_s to vertex v_{s+k} in the graph with an associated cost of $|log(P(O_{s,s+k}|\lambda_i))|$. We augment each weight in the graph with a term that accounts for the probability that $O_{s,s+k}$ is the encoding of a complete object. This is achieved by penalizing edges corresponding to incomplete objects, by testing whether the observation used for that edge puts λ_i in one of its final states using the ending probabilities estimated offline. If the destination index $s + k$ exceeds the index of v_f, instead of trying to link v_s to v_{s+k}, we put a directed edge from v_s to the final node v_f. We set the weight of the edge to $|log(P(O_{s,|O|}|\lambda_i))|$. Here $O_{s,|O|}$ is the suffix of O starting at index s. This allows us to do recognition when the scene is not yet complete and is a major strength of our approach. We complete the construction of G by repeating this operation for all models.

In the constructed graph, having a directed edge from vertex v_i to v_j with cost c means that it is possible to account for the observation sequence $O_{i,j}$ with some model with a loglikelihood of $-c$. The constructed graph may have multiple edges connecting two vertices, each with different costs. By computing the shortest path from v_1 to v_f in G, we minimize sum of negative loglikelihoods, equivalent to maximizing the likelihood of the observation O. The indices of the shortest path gives us the segmentation. Classification is achieved by finding the models that account for each computed segment.

A nice feature of the graph-based approach is that while the shortest path in G gives us the most likely segmentation of the input, we can also compute the next k-best segmentations using a k-shortest path algorithm. This information can be used by another algorithm for dealing with ambiguities or by the user, as done in speech recognition systems with n-best lists.

4. EVALUATION

4.1 Evaluation of the HMM-based recognition

Our first experiment was aimed at measuring the suitability of our approach for sketch recognition and observing its behavior with test data containing clutter in the form of spurious strokes or unknown objects.

We trained models with 10 states for 10 objects from the domains of geometric objects, military course of action diagrams, stick-figure diagrams, and mechanical engineering drawings. Training data was sketched using up to 6 styles with 10 examples per style. For a separate test set of 88 objects, we achieved a 96.5% recognition rate (see Fig. 1 for an example scene).

We also tested our method with sketches that include negative examples to measure its robustness in presence of unknown objects and spurious strokes. Two classes of negative examples were obtained by randomly inserting strokes selected from other sketches, and by simulating the effects of common low level recognition errors. We observed that 69% of the time, spurious strokes caused misrecognitions that were bounded to objects immediately preceding or following the source of error. For low level errors, this rate was 78%. This implies that effects of errors usually remain local.

4.2 Running time comparison to a baseline method

To serve as a baseline, we implemented a feature-based recognizer of the sort used in [7, 3, 8]. The baseline method

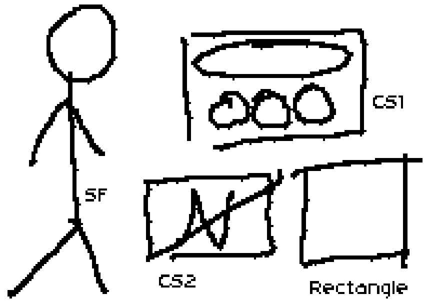

Figure 1: An example input for our stem, and the system's recognition indicated by labels. The scene contains with two course of action diagram symbols, a rectangle and a stick-figure.

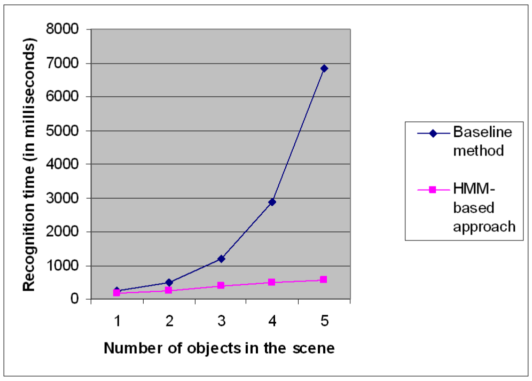

Figure 2: Running times for the baseline feature-based recognition system and our HMM-based recognition system.

is symbolic and feature-based. We ran the baseline system on scenes containing up to 5 objects (rectangles or stick-figures in this case). To avoid stroke orderings that are unusually ambiguous for the baseline method, we measured the mean recognition times for different drawing orders. We started the experiment with a scene containing a single rectangle, and in an alternating manner, we added either a stick-figure or a rectangle to get a new scene. We repeated the experiment with up to 5 objects (three rectangles, two stick-figures). Fig. 2 shows the average times for increasing number of objects for the baseline and our system on a Pentium III 933 MHZ Windows XP machine with 512M of RAM. As seen in Fig. 2, the HMM-based method scales very well with increasing number of objects. These results show how valuable drawing order information can be when users sketch in predictable orders.

5. RELATED WORK

Work in [3] and [8] describe grammar-based statistical approaches to sketch recognition, while [4] presents a technique for generating bottom-up recognizers from object descriptions that perform exhaustive search. Our work naturally complements these systems.

Work in [6] describes an HMM based sketch interpretation and refinement system, though this system requires objects to be drawn using single strokes. Scenes are recognized by looking at spatial relationships of consecutive strokes.

Our system also differs from numerous sketch recognition systems by its ability to do recognition with only polynomial time and space complexity, and by its utilization of drawing order for capturing and modeling user sketching styles.

6. FUTURE WORK

We are investigating alternative HMM setups (number of states, topology, encoding schemes) and exploring ways of inducing HMM properties by applying Bayesian model merging techniques. We are also interested in exploring hierarchical HMMs to capture inter-object correlations.

We are currently developing a new HMM architecture that integrates our system with a model based sketch recognition architecture, which will allow two approaches to share hypotheses and help each other.

We believe the HMM framework is also appropriate for learning editing operations (e.g., deletion, selection) if corresponding observations are supplied during training.

7. SUMMARY

We have showed how viewing sketching as an interactive process allows us to model and recognize sketches using Hidden Markov Models. We presented results of a user study indicating that in certain domains people have preferred ways of drawing objects. We illustrated how the consistent ordering of strokes naturally preferred by users can be exploited to build models for individual users and perform sketch recognition efficiently, without restricting the users to sketch in a certain way.

8. ACKNOWLEDGMENTS

This research was supported by the Project Oxygen and the Microsoft iCampus Project at MIT.

9. REFERENCES

[1] Rabiner L. R. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE, Vol. 77, no 2, February 1989.*

[2] Sezgin T. M., Stahovich T. and Davis R. Sketch Based Interfaces: Early Processing for Sketch Understanding. *PUI'01.*

[3] Alvarado C., Oltmans M. and Davis R. *A Framework for Multi-Domain Sketch Recognition. AAAI Sketch Symposium 2002..*

[4] Sezgin T. M. Generating Domain Specific Sketch Recognizers from Object Descriptions. *Student Oxygen Workshop.* July 17, Gloucester MA, 2002.

[5] Stahovich T., Landay J. and Davis R., Chairs. *AAAI Sketch Understanding Symposium.* March 25-27, Stanford CA, 2002.

[6] Saul Simhon and Gregory Dudek Sketch interpretation and refinement using statistical models *Eurographics Symposium on Rendering.* Norrkoping, Sweden, 2004.

[7] Mahoney V. J., and Fromherz M. P. J. Three main concerns in sketch recognition and an approach to addressing them. *AAAI Spring Symposium: Sketch Understanding.* March 25-27, Stanford CA, 2002.

[8] Shilman M., Pasula H., Russel S. and Newton R. Statistical Visual Language Models for Ink Parsing. *AAAI Spring Symposium: Sketch Understanding.* March 25-27, Stanford CA, 2002.