Three main concerns in sketch recognition and an approach to addressing them

James V. Mahoney and Markus P. J. Fromherz

Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304
{mahoney,fromherz}@parc.xerox.com

Abstract

This paper discusses the problem of matching models of curvilinear configurations to hand-drawn sketches. It collects observations from our own recent research, which focused initially on the domain of sketched human stick figures in diverse postures, as well as related computer vision literature. Sketch recognition, i.e., labeling strokes in the input with the names of the model parts they depict, would be a key component of higher-level sketch understanding processes that reason about the recognized configurations. A sketch recognition technology must meet three main requirements. It must cope reliably with the pervasive variability of hand sketches, provide interactive performance, and be easily extensible to new configurations. We argue that useful sketch recognition may be within the grasp of current research, if these requirements are addressed systematically and in concert.

1. Introduction

This paper discusses the problem of matching models of curvilinear configurations to hand-drawn sketches. It collects observations from our own recent research, which focused initially on the domain of sketched human stick figures in diverse postures, as well as related computer vision literature. By sketch *recognition* or *matching* we mean labeling strokes in the input with the names of the model parts they depict. Such matching would be a key component of higher-level sketch *understanding* processes that reason about the recognized objects and configurations.

Our view is that a sketch recognition technology must meet three main requirements. First, and perhaps foremost, it must cope reliably with the variability and ambiguity that are so pervasive in sketches. Second, it must provide interactive performance or better. And third, it must offer easy or automatic extensibility to new shapes and configurations. These requirements conspire to make a problem that might at first appear to be child's play into a profound challenge. We believe, however, that useful sketch recognition is within the grasp of current research, if the three requirements are addressed systematically and in concert; this paper outlines an approach we are pursuing.

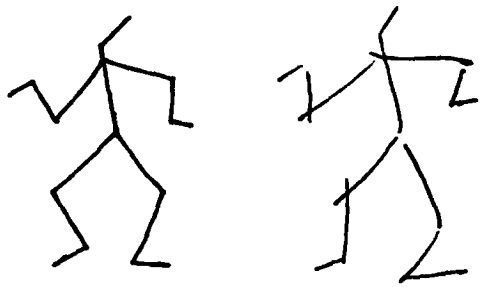


Fig. 1. Neat and sloppy stick figures. In sketching, failures of co-termination are a frequent variation from the ideal form.

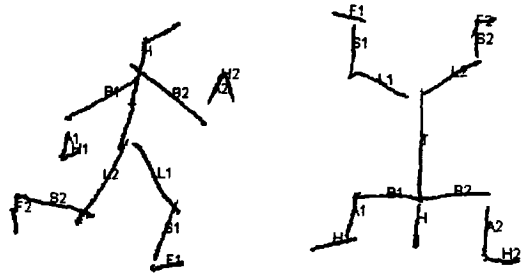


Fig. 2. Example matching results. Labels denote model parts: Head, Torso, Biceps1, Arm1, Leg 2, Shin2, Foot2 etc.

Section 2 describes a framework for addressing input variability, and the ambiguity that results from it, through dedicated pre-processing that is guided by human perceptual organization principles.

The need to handle articulated figures, and configurations that are defined by abstract spatial relations among their parts (e.g., connected, inside/outside, parallel, etc.), makes sketch recognition inherently a structural modeling and matching enterprise. Section 3 argues that although the worst-case complexity of structural matching in this domain is exponential, exploiting the natural

topological and geometric constraints of the problem to the fullest leads to runtimes that are in a useful range for problems of moderate size. Even so, because of the potential for exponential growth, Section 4 argues that various notions of focus ought to be brought to bear in matching where possible.

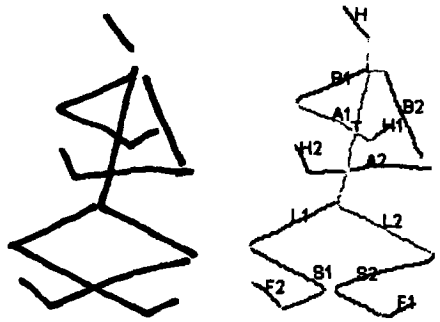


Fig. 3. Matching must cope with self-crossings, due to articulation.

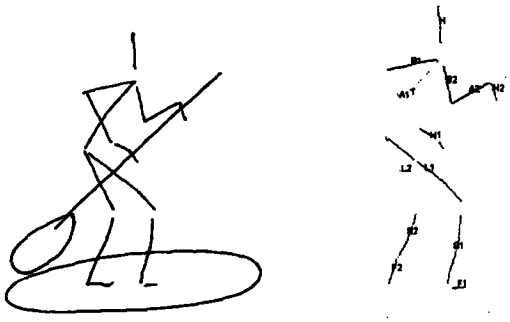


Fig. 4. Matching must cope with interactions with background context, e.g., crossings and accidental alignments.

Finally, Section 4 examines the issue of model acquisition, and argues that even in the relatively simple realm of hand sketches, manual entry of configuration models is likely to be prohibitively tedious. At the same time, machine learning-based schemes that require the user to present many training examples will probably also be too inconvenient in most cases. What is needed, therefore, are methods that will build a useful model given one or very few examples, possibly with incremental refinement as further instances are provided at the user's convenience.

2. Handling variability and ambiguity through dedicated pre-processing

It is hard to write reliable computer programs for recognizing drawings, diagrams, and sketches, even seemingly simple ones. The main obstacle is that these kinds of inputs are highly variable in several respects, and effective, general techniques for coping with this variability have not been developed. The variability stems from several distinct sources. To address this overall concern, it is important to identify these sources and to establish detailed generative models for each of them. Our work on stick figure matching has focused on three such sources: sloppy drawing, articulation, and interaction with background context [13,14].

Sloppy drawing leads to *failures of co-termination*, where strokes overshoot or undershoot one another, rather than meeting at their end points (see Fig. 1). The problem resulting from articulation is that the simple preprocessing techniques that are normally employed may over-segment or under-segment the data if the strokes of articulated figures intersect or accidentally align with one another (Fig. 3). This is also the case when target figures interact with background markings, through adjacency, intersection, or overlap (Fig. 4). (There are several further sources of variability that we will not address, including sensor noise, which leads to dropouts and speckle, and shape deformations of the figure lines.)

For matching purposes, all these forms of variability corrupt the expected mapping between a prior model of the figure and an instance of it in an input scene. The simplest situation for a matching task occurs when applying a segmentation process to the input produces primitives that are in one-to-one correspondence with the parts specified in the configuration model. Using straightforward segmentation schemes, however, the above types of variability often lead to a many-to-one or one-to-many mapping. For example, if the input scene is simply segmented into simple curves meeting at junctions and corners, then gaps in figure lines, crossings of figure lines with one another, or crossings of figure lines with background lines would each entail a many-to-one mapping from these simple curves to the parts in a curvilinear configuration model. Conversely, accidental alignments among figure lines or between figure lines and background lines would entail a one-to-many mapping.

In this paper, the term *ambiguity* (or, more precisely, *segmentation ambiguity*) refers specifically to this lack of a one-to-one mapping between the primitives produced by segmentation and the elements of the model, together with the lack of any bottom-up (i.e., model-independent) basis for establishing such a one-to-one mapping. In the computer vision literature, the prevalent approach to structural matching in the presence of segmentation ambiguity is *error-tolerant* subgraph matching [26,23,15], wherein the matching process explicitly accounts for discrepancies in structure, by searching for a mapping that minimizes the structural difference between the model and any data subgraph. (The computation of this difference, or *edit distance*, reflects a predefined way of associating costs with particular discrepancies.)

The drawback of this approach is that generalizing from exact subgraph matching to error-tolerant matching increases complexity: from $O(mn)$ to $O(mn^2)$ in the best case; from $O(m^n n^2)$ to $O(m^{n+1} n^2)$ in the worst case; m and n being the node counts of the data and model graphs respectively [15]. This added cost is typically incurred for every model matched. (Hierarchical modeling and matching may allow some of the added cost to be shared if there is a lot of common structure among the modeled configurations.) Error tolerant matching is also algorithmically more complicated, and it rules out the use of off-the-shelf subgraph matching implementations.

In [14], we proposed an alternative way of handling ambiguity in a subgraph matching recognition framework that avoids the added computational and algorithmic complexity of error tolerant matching. Our method is a two-stage process. The first stage, termed *graph elaboration,* explicitly addresses ambiguity by adding subgraphs to the data graph, G_D, that constitute plausible alternative descriptions of substructures already in the graph. This stage applies general *perceptual organization* (P.O.) principles, such as good continuation and proximity, to the specific goal of producing a data graph in which the model is much more likely to find a direct match, i.e., one containing a subset of nodes and links that map to the model in a one-to-one manner. P.O. principles are also used to rank the alternatives or to associate preferences with them. The subsequent matching phase is a constrained optimization process that enforces mutual exclusion constraints among the related alternatives, and finds solutions in descending order of the aggregate preferences or rankings of their components. When one alternative is strongly preferable to all others on perceptual grounds, the initial stage may instead simply rule out the other alternatives in a related process we call *rectification.*

We have explored five graph elaboration/rectification operations in particular. *Proximity linking* in effect fills gaps in the data; an edge is added to G_D between two free ends if they satisfy a proximity constraint (Fig. 5). *Link closure* adds redundant co-termination links to reduce sensitivity to small distance variations. *Virtual junction splitting* introduces proximity links between free ends and neighboring non-end points (Fig. 6). *Spurious segment jumping* enables the matcher to ignore curve segments generated when other segments just miss coinciding at a common junction (Fig. 7). If two segments satisfy a smooth continuation constraint, *continuity tracing* adds a new subgraph to G_D that represents the alternative interpretation of them as a single curve (Fig 8).

Each of these operations is controlled by a single parameter, with the exception of link closure, which has none. In our current implementation these parameters have been tuned manually to give roughly equal matching accuracy across several example sets that each emphasize a distinct sort of local variability. One topic for future investigation is automatic estimation of these parameters from training data.

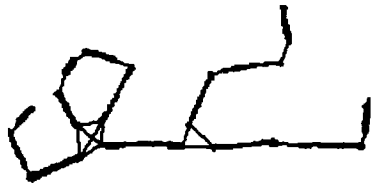


Fig. 5. Proximity linking: Links are inserted by an extension of Kruskal's Minimum Spanning Tree algorithm that builds a "close-to-minimal spanning graph".

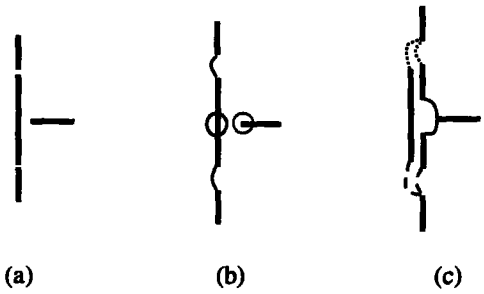


Fig. 6. Virtual junction splitting. (a) Input configuration. (b) Exploded view showing links defined before junction splitting. Small circle on left indicates virtual junction point detected in relation to free end circled on right. (c) Two new segments are added with links to the free end and also to the segments previously linked to the original segment.

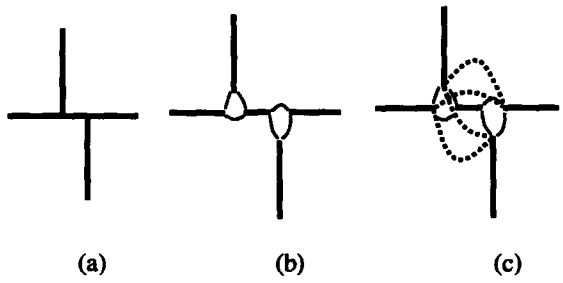


Fig. 7. Spurious segment jumping. (a) A spurious segment created by the two vertical segments failing to coincide. (b) Exploded view of (a), showing co-termination links defined prior to segment jumping. (c) Links added by segment jumping (dashed lines) enable the matcher to ignore the spurious segment.

3. Efficient structural matching by constrained optimization

Structural matching is an inherently complex search task. We see two main ways of maximizing matching efficiency. First, we may attempt to exploit to the fullest whatever constraint is intrinsic to the matching problem

itself. This inherent constraint includes general perceptual organization constraints and the topological and geometric constraints implicit in the configuration models being matched. Second, we may apply any available extrinsic constraints, such as knowledge about the user's current goal, to focus and limit this search to what makes the most sense. The first aspect is the topic of this section. Section 4 makes brief suggestions for future work on ways of applying extrinsic constraints through focused processing.

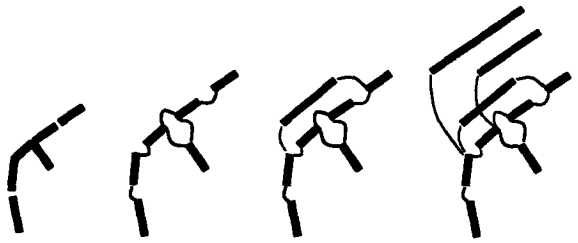


Fig. 8. Continuity tracing. (a) Input figure. (b) Exploded view showing links defined before continuity tracing. (c) A new segment added by continuity tracing, along with associated links. (d) All segments added by repeating continuity tracing to convergence.

Algorithmic formulations of structural matching, such as subgraph matching [6], graph rewriting [3], parsing of graph grammars [1], etc., have exponential worst-case complexity. Structural matching is feasible, however, because most practical problems do not entail the worst case. For example, in our subgraph matching approach to sketch recognition, the preprocessing stage by design produces a data graph that is not totally connected, but rather whose links reflect perceptually salient spatial relations in the input. Thus, the preprocessing discussed in Section 2 is a key determiner of matching performance; it attempts to produce data graphs that are as concise as possible while still providing the needed alternative interpretations in ambiguous circumstances. Similarly, we design configuration models to make explicit only those relations required for matching and discrimination. Having met these conciseness requirements on the data and model graphs, it remains to specify a matching scheme that can take full advantage of the implicit structural and geometric constraint. The rest of this section outlines a constrained optimization approach to subgraph matching that we developed for the sketching recognition application [14,6].

The data graph, G_D, is constructed using standard image processing operations. The image is binarized and thinned; the simple curve segments, extracted by tracing, are then split into smooth segments at salient corner points. (Corners are detected based on a technique in [22].) For each of the resulting segments, its graph representation is added to G_D. The graph representation of a segment consists of *two* nodes, one for each end point, connected by a type of edge we call a *bond*. For each pair of curves terminating at the same junction or corner, another type of edge, called a (*co-termination*) *link*, connecting their co-terminal end nodes, is added to the graph. The data graph then, consists of a single type of node, representing a curve segment end point, and two types of edges (links and bonds).

A simple syntax for expressing stick figure configuration models is illustrated below. Each **limb** statement defines a part of the figure. The **optional** modifier allows a part to be missing in the data. The **linked** statement asserts an end-to-end connection between two curvilinear parts. Two optional integer arguments allow the modeler to specify with which end of each part the link is associated. For example, the (default) values (2,1) indicate that the link goes from the second end of the first named part to the first end of the second, where "first" and "second" are assigned in an arbitrary but consistent manner. The syntax also allows constraints on part attributes to be specified. For example, the **minimize** statement in this example specifies ideal relative limb lengths.

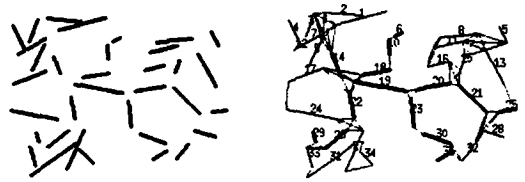
```
model stick_figure {
  limb head, torso, biceps1, ...;
  optional limb hand1, hand2, ...;
  link(head, torso);
  link(torso, biceps1, 1, 1);
  ...
  minimize (torso.len-2*head.len)^2
         + (2*torso.len-3*biceps1.len)^2
         + ...;
  ...
} // end model stick_figure
```

Subgraph matching may be formulated as a constraint satisfaction problem (CSP) as follows. Mirroring the data graph, a limb in the model is represented as a pair of nodes, representing the limb's ends, connected by a bond. A link statement in the model specification gives rise to links between the specified nodes. A CSP variable is defined for each node in the model graph. The initial domain of each variable is the set of data graph nodes plus the label *null* for a missing part, since our models may specify some parts as optional.

The primary constraint of the problem, termed *link support*, is that a link/bond, l, between two model nodes, m_1, m_2, requires the existence of a corresponding link/bond between the associated data nodes d_1, d_2. (Details of this constraint that allow specified parts to be missing in the data are given in [14].) The *unique interpretation* constraint requires that each data node may be assigned to at most one model node. This is

implemented as a global cardinality constraint. These two constraints are sufficient to establish a purely topological match between the model and data graphs. This is often adequate for matching to isolated instances, but spurious matches arise if the data contains spurious strokes, gaps, or self-crossings. To find reliable matches in such cases, further quantitative criteria that rank topologically equivalent solutions, e.g., based on geometry, must also be applied. We have explored three such criteria.

The *minimal total link length* criterion prefers interpretations with smaller total length of links participating in the match. (A link participates in the match if it corresponds to a link in the model.) The *optimal part proportions* criterion prefers interpretations in which the length ratios of the segments closely approximate those specified in the model. The *maximal part count* criterion prefers matches that leave fewer model parts unassigned; it is needed because the constraints above allow optional model parts to be missing and make no distinction between solutions with differing numbers of missing optional parts.



(a) (b)

Fig. 9. (a) A stick figure with 20 distractor lines. (b) The corresponding graph with labels and links produced by the image analysis stage. The strokes that matched the model are shown as bold lines.

To handle ambiguity, as represented in an elaborated data graph, two global mutual exclusion constraints and a quantitative criterion are added. The first constraint enforces mutual exclusion among links that are in the same mutual exclusion set. The second constraint enforces mutual exclusion among data subgraphs that have primitive segments in common; if a given segment is assigned to a model part, no other segment built from any of the same primitive segments may be assigned to any model part. The added quantitative criterion prefers curve segment chains that contain more segments. This emulates the fact that long smooth curves are normally perceived as more salient than their constituent segments.

One of our implementations solves the above CSP using a branch-and-bound state-space search framework. A state consists of an assignment to the set of CSP variables. (In the initial state, all variables are unassigned. In a goal state, all variables are assigned such that all constraints are satisfied.) The *successor state function*, which is given a state taken from the search queue and returns a set of new states to be added to the queue, selects an unassigned CSP variable and creates a new state for each possible assignment of this variable. It applies the specified constraints to effect possible reductions in the set of new states generated.

The quantitative criteria are incorporated into the objective function that is optimized by the search process. All four optimization criteria are combined into a single function as a weighted sum. The link length and part proportion criteria have equal weights while the part count criterion is given a much higher weight; i.e., the first two criteria only come into play among solutions with equal part counts. The segment count criterion is assigned a weight in between these other two values.

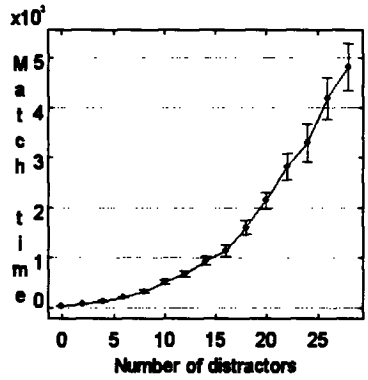


Fig. 10. Mean match time (in milliseconds) for a figure with 0 through 28 random distractor lines. The error bars indicate the standard error over 100 runs (10 test figures, at 10 runs each).

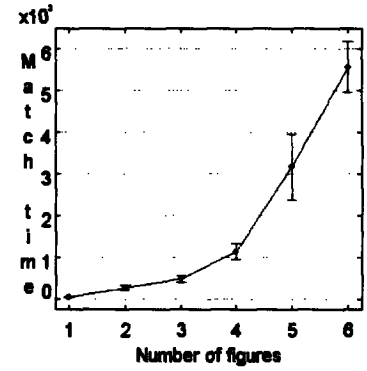


Fig. 11. Mean match time (in milliseconds) for composite images containing 1 to 6 figures. The error bars indicate the standard error over 20 different composites.

Our evaluations of the approach have focused so far on how runtime and accuracy scale with data graph complexity. Input complexity was varied in two ways: by adding random distractor lines to images of isolated figures, and by composing multiple figure scenes at

random from individual example figures. Figures 10 and 11 show runtime results for these experiments, for a Java implementation running on a 600 MHz Pentium III processor. Figure 9 shows an example figure with distractor lines, its data graph, and the associated matching result. Figures 12 and 13 show an example composite input and its matching result.

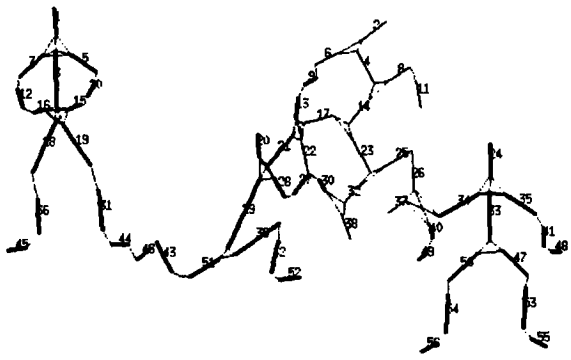


Fig. 12. Sketch with three stick figures and a distractor figure, with labels and links produced by the image analysis stage.

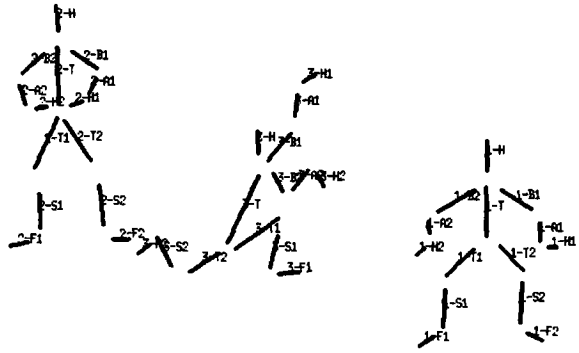


Fig. 13. Interpretation of the data of Fig. Z, with part labels preceded by the index of the instance.

4. Reducing matching complexity through focused processing

The approach of Section 3 gives runtime performance in a useful range for interactive applications, when applied to problems of moderate size. But as inputs and model sets grow large, performance degrades substantially, and the approach does not rule out the possibility of exponential growth. Additional criteria must be applied to focus and limit the search for a match to the most promising data and models. We can distinguish four main notions of focus applicable to sketch recognition, to be explored in future work: anchors, salience, grouping, and model bias.

We have an *anchor* when part of the target match is given *a priori*. E.g., one or more strokes in the data may be known *a priori* to belong to the target figure, perhaps based on an interactive indication by the user. During matching an anchor may be treated as including one or more strokes of the target figure, or as including all strokes of the target figure. Typically, the latter interpretation of an anchor would give more restriction on search but it requires more careful user selection. In an interactive context, the user would somehow specify in which of these ways an anchor should be interpreted.

Salience refers to a prior labeling of data elements that reflects how promising they are for matching. Salience criteria may be bottom-up or top-down. Bottom-up, salience criteria reflect general perceptual principles or phenomena. An example of such a phenomenon is pop-out, in which an element is singled out for focused processing based on how different it is from neighboring elements, or the global distribution of elements, with respect to properties such as length or orientation. Top-down criteria involve domain or task-specific knowledge. For example, the set of configuration models currently selected for matching by the user can be used to rank scene elements with respect to their goodness of fit to that set of models. The fit of an element is measured in terms of its local relations and relative attributes.

Salience information may be exploited in two ways. First it can be applied directly to controlling the search order; i.e., more salient elements may be tried first. Second, it may be used to define anchors: i.e., a threshold may be applied to the salience values, and search may be restricted to the above-threshold elements.

Grouping defines coherent subsets of the data. Groups may be treated as independent matching sub-problems that are not only more manageable in size that the entire input, but being visually coherent, also more sensible. Criteria for defining groups include the Gestalt principles of similarity, proximity, good continuation, closure, and symmetry. There is a significant research challenge in developing computational formulations of these grouping criteria that are useful in the sketch interpretation domain, but there is some promising work to suggest that this research direction will be fruitful. E.g., Saund presents techniques for grouping sketched curves by good continuation [20] and closure [21]. Thorisson [25] presents techniques for similarity and proximity grouping that are applicable to sketched data.

Groups may also influence which models are selected as promising candidates for matching, and in what order they are tried. For example, we may rule out models that have too many parts compared to the number of strokes in a given group.

Model bias is the *a priori* specification of which sets of models should be applied in a given matching operation. E.g., matching may be restricted to a given domain-

specific family of models, or a set of models a user has interactively selected.

5. Automating model acquisition

Part of the appeal of a structural modeling approach is that the matcher may be manually extended to a new configuration in an intuitive way, by specifying by name its constituent parts and relations. In practice, however, for all but the simplest configurations, manual entry of models is not very convenient. A structural model for a curvilinear configuration of even moderate complexity may involve dozens of statements specifying parts, attributes, and relations. Moreover, the statements that specify geometric attributes, such as length proportions and angles, involve numerical parameters that are usually not readily accessible to introspection. (Also, as was noted in Section 3, a number of additional parameters are used to specify the relative importance of the various matching constraints. These are not associated with individual models, but they may depend on the domain being modeled, or the sketching style of particular users.) Add to this the fact that an effective recognition system might incorporate dozens or hundreds of models, and new configurations are continually being designed or improvised by users.

It seems clear that for sketch recognition to be useful, the burden on users of entering models must be kept to a minimum. One approach is to develop techniques that would automatically or interactively learn models from user specified examples. Our exploration of this problem is just beginning, but it is possible to state some broad requirements at the outset. The advantage of learning over manual entry is greatly diminished if the user must initially draw or collect many examples and non-examples. Therefore, the learning scheme should require as few training examples as possible, and preferably only positive examples. Ideally, learning should be incremental, generating a useable model from a single positive example, and then gradually refining that model as new examples are presented at the user's convenience.

Several distinct literatures appear to be pertinent to this specification of the problem. It may be possible to ground future work on learning curvilinear configuration models in a combination of methods and insights from all of them. One key division is between schemes for inductive learning of the part-relation (graph) structure of the model (i.e., learning structural descriptions), and methods for estimating ideal values or ranges for the numerical parameters of a pre-defined structure (i.e., parameter estimation).

AI techniques for learning structural descriptions [27,16] ought to be explored in the sketch recognition domain. As originally presented, these methods may often require negative examples and/or a more careful selection of examples by the teacher that we would like. However, it may be possible to adapt these techniques to meet our requirements. For example, an interesting application of version spaces to learning from small numbers of examples, in the programming-by-demonstration domain, is given in [11,12].

The literature on grammatical induction contains various incremental techniques for learning regular grammars [3,17,18,19]. For use in sketch recognition, these methods may need to be extended to the case of tree, graph, or plex grammars [7,8]. Structural models of the type introduced in Section 3 can be derived from grammars. Alternatively, one may attempt to adapt insights from the grammar induction work to the problem of learning model graphs directly.

These structure learning and grammatical induction methods are not applicable to the problem of learning the model's numerical attributes. A computational model -- applicable to both parameter estimation and structure learning -- of how humans are able to learn concepts from small numbers of positive examples is presented in [24].

Note that the graph elaboration ideas of Section 2 will likely play a key role in enabling learning. Without them, as in the case of matching, appropriate error tolerance must be built into the learning process itself, to cope with variability and noise. Most learning techniques in the literature make no provision for such error tolerance. Prior elaboration should mitigate the complexity of learning no less than that of matching.

6. Conclusion

This paper proposed a framework for the design of effective sketch recognition systems. The three pillars of that framework are (i) dedicated pre-processing to address the effects of drawing variability; (ii) a constrained optimization, subgraph matching approach to recognition, possibly applied in a focused manner to mitigate the inherent complexity of the problem; and (iii) automated model acquisition through incremental and highly effective learning techniques.

Our research so far has realized only parts of this program; we are turning to the rest in ongoing work. There are certain segmentation ambiguities, such as corners, that we have identified but for which we have not developed elaboration operations. We plan to explore ways of estimating the parameters of the elaboration operations automatically, possibly subject to the influence of given sets of configuration models. We intend to experiment with a wider range of configuration models, with control schemes for matching multiple models, and with input data drawn from diverse users and domains.

The approach in this paper deals mainly with curvilinear configurations, and the models we have tested are defined primarily by end-to-end connectivity relations. We believe the matching approach will extend naturally to

other abstract relations among strokes, such as inside/outside, left/right, perpendicularity, parallelism, etc. However, it must be noted that some recognition problems in the sketch domain, such as recognition based primarily on curve shape, may fall outside the scope of a structural modeling and matching approach altogether.

Acknowledgements

We are grateful to David Fleet, Eric Saund, and Dan Larner, of the Perceptual Document Analysis Area at Xerox PARC, and Allan Jepson, of the University of Toronto, for helpful discussions over the course of this research.

References

1. S. Chok, K. Marriot. Parsing visual languages. *Proc. 18th Australasian Computer Science Conf.*, 27(1), 1995: 90–98.
2. T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*. Cambridge, MA: M.I.T. Press, 1990.
3. F. Denis, *Learning regular languages from simple positive examples*, Machine Learning, vol. 44 (1/2):37-66, July 2001.
4. P. Dupont. Incremental regular inference. In *Grammatical Inference: Learning Syntax from Sentences, ICGI'96*, number 1147 in Lecture Notes in Artificial Intelligence, pages 222-237. Springer Verlag, 1996.
5. H. Fahmy and D. Blostein. A graph-rewriting paradigm for discrete relaxation: application to sheet music recognition. *Int. Journal of Pattern Recognition and Artificial Intelligence, vol. 12*, no. 6, Sept. 1998,pp. 763-799.
6. M. Fromherz and J. Mahoney. Interpreting sloppy stick figures with constraint-based subgraph matching. *7th Int. Conf. on Principles and Practice of Constraint Programming*. Paphos, Cyprus: 2001.
7. K. Fu, *Syntactic pattern recognition and applications*, Prentice-Hall, NJ, 1982.
8. R. Gonzalez and M. Thomason, *Syntactical Pattern Recognition*. Addison-Wesley, 1978.
9. D. Isenor and S. Zaky. Fingerprint identification using graph matching. *Pattern Recognition, vol. 19*, no. 2, 1986, pp. 113-122.
10. J. Larrosa and G. Valiente, Constraint satisfaction algorithms for graph pattern matching. Under consideration for publication in *J. Math. Struct. in Computer Science*, 2001.
11. T. Lau, P. Domingos, and D. Weld, Version Space Algebra and its Application to Programming by Demonstration, ICML 2000, Stanford, CA, June 2000, pp. 527-534
12. T. Lau, S. Wolfman, P. Domingos, and Daniel S. Weld, Programming by Demonstration using Version Space Algebra, to appear in *Machine Learning*.
13. J. Mahoney and M. Fromherz. Interpreting sloppy stick figures by graph rectification and constraint-based matching. *4th IAPR Int. Workshop on Graphics Recognition*: Kingston, Ontario: Sept., 2001
14. J. Mahoney and M. Fromherz. Handling Ambiguity in constraint-based recognition of stick figure sketches. *Document Recognition and Retrieval IX*: San Jose, California: Jan., 2002
15. B. Messmer. Efficient graph matching algorithms for preprocessed model graphs. PhD thesis. Bern Univ., Switzerland, 1995.
16. T. Mitchell. *Version spaces: an approach to concept learning*. PhD dissertation, Stanford University, Dept. of Electrical Engineering, 1978.
17. Parekh, R. and Honavar V., An Incremental Interactive Algorithm for Regular Grammar Inference. *3rd Int. Colloquium on Grammatical Inference* (ICGI'96), Montpellier, France. September 24-27, 1996. *Lecture Notes in Computer Science* vol. 1147 pp. 238-250
18. Parekh, R. and Honavar, V., Learning DFA from Simple Examples. *8th Int. Workshop on Algorithmic Learning Theory* (ALT'97), Sendai, Japan. Oct. 6-8, 1997. *Lecture Notes in Computer Science* vol. 1316 pp. 116-131.
19. Parekh, R., Nichitiu, C. and Honavar, V., A Polynomial Time Incremental Algorithm for Learning DFA. *4th Int. Colloquium on Grammatical Inference* (ICGI'98), Ames, IA. 1998. *Lecture Notes in Computer Science* vol. 1433 pp. 37-49.
20. E. Saund. Labeling of curvilinear structure across scales by token grouping. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1992: 257—263.
21. E. Saund. Perceptually closed paths in sketches and drawings. *3rd Workshop on Perceptual Organization in Computer Vision*. Vancouver, Canada, July, 2001.
22. E. Saund. Perceptual organization in an interactive sketch editor. *5th Int. Conf. on Computer Vision*. Cambridge, MA: 1995: 597–604.
23. L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. In *IEEE Trans. on Pattern Analysis and Machine Intelligencl, vol. 3*, no. 5, Sept. 1981, pp. 504-519.
24. J. Tenenbaum. Bayesian modeling of human concept learning. *Advances in Neural Information Processing Systems 11*, 1999.
25. K. Thorisson. "Simulated perceptual grouping: an application to human-computer interaction. " *Proc. 6th Annual Conf. of the Cognitive Science Society*. Atlanta, Georgia,Aug 13-16, 1994: 876—881.
26. W.H. Tsai and K.S. Fu. Subgraph error-correcting isomorphisms for syntactic pattern recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 13(1):48--62, Jan-Feb 1983
27. P. Winston. Learning structural descriptions from examples. In *The Psychology of Computer Vision*. McGraw-Hill, 1975.