

# Interpreting sloppy stick figures by graph rectification and constraint-based matching.

James V. Mahoney and Markus P. J. Fromherz

Xerox Palo Alto Research Center  
3333 Coyote Hill Road  
Palo Alto, CA 94304

{mahoney, fromherz}@parc.xerox.com

**Abstract.** Programs for understanding hand-drawn sketches and diagrams must interpret curvilinear configurations that are sloppily drawn and highly variable in form. We propose a two-stage subgraph matching framework for sketch recognition that can accommodate great variability in form and yet provide efficient matching and easy extensibility to new configurations. First, a rectification stage corrects the initial data graph for the common deviations of each kind of constituent local configuration from its ideal form. The model graph is then matched directly to the data by a constraint-based subgraph matching process, without the need for complex error-tolerance. We explore the approach in the domain of human stick figures in diverse poses.

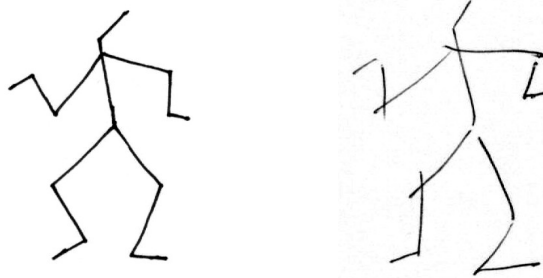
## 1. Introduction

Programs for understanding hand-drawn sketches and diagrams must interpret common curvilinear configurations, such as arrows, geometric shapes, and conventional signs and symbols. These figures are often sloppily drawn and highly variable from one instance to the next. A recognition system should accommodate considerable sloppiness and variability in form, but without sacrificing matching efficiency or easy extensibility to new configurations. This paper proposes a framework for meeting these requirements, exploring it in the domain of human stick figures in diverse poses.

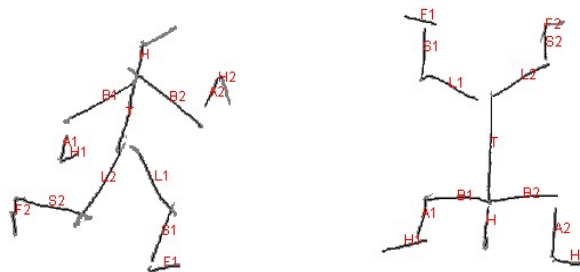
Figure 1 illustrates what we mean by “sloppiness”. In the neat example, the lines meet precisely at junctions and corners. In the sloppy case, the lines often fail to be co-terminal, due to drawing overshoot or undershoot. Basic image analysis techniques alone (thinning, tracing, and detection of junctions and corners) would not provide an adequate basis for reliable recognition in this case.

Sloppiness is pervasive; people cannot draw a figure exactly the same way twice. In addition, there are several other pervasive sources of the variability with which matching must contend. Noise in sensing what has been drawn introduces gaps and extraneous marks. The class of figure to be recognized may be articulated and/or flexible in shape (Fig. 3). Finally, there is interaction with context: the figure may overlap or intersect surrounding figures or markings (Fig. 4).

Because the configurations found in diagrams and sketches are often articulated or abstract, this recognition problem is well suited to a structural modeling approach. The configuration model and the input scene are represented as attributed graphs, with nodes representing figure parts (e.g., lines), and edges representing part relations (e.g., line connections). The attributes on nodes and edges represent geometric properties of the underlying lines and line relations. Recognition is cast as subgraph matching of the *model graph* to the *data graph*, wherein each line in the input is labeled with the name of the model part that it depicts. Matching subgraphs, rather than graphs, allows for background context in the input scene.



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



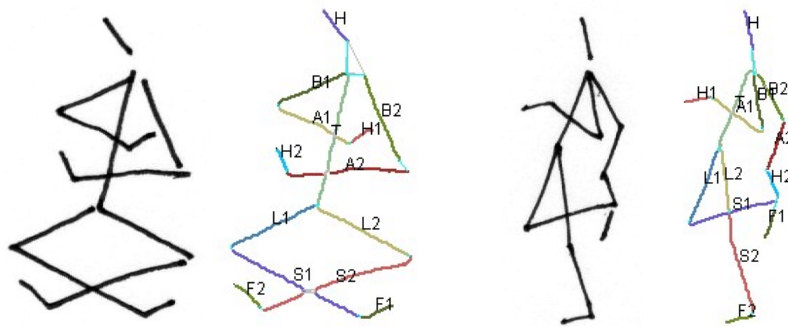
**Fig. 2.** Example matching results. Labels denote model parts: Head, Torso, BicepsL, ArmL, etc.

A stick figure configuration *model* is expressed in a simple syntax, 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 optimal relative limb lengths.

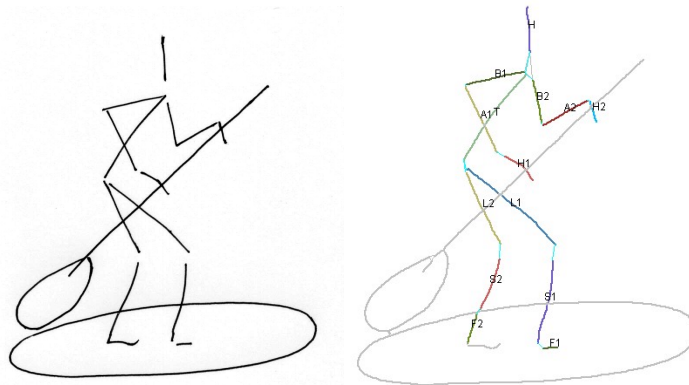
```

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

```



**Fig. 3.** Matching must cope with variations, e.g., self-crossings, due to articulation.



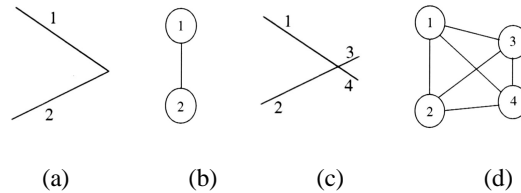
**Fig. 4.** Matching must cope with variations, e.g., crossings, due to background context.

Consider an initial data graph,  $G_D$ , created from an image of a line drawing like Fig. 1. Its nodes represent curve segments that result from applying standard operations of binarization, thinning, curve tracing, etc. Two nodes are linked if their curve segments terminate at the same junction or corner. Due to drawing variability

and noise, the resulting graph would rarely contain a verbatim instance of the model as a subgraph.

The prevalent solution in the literature is to use error-tolerant subgraph matching to explicitly account for discrepancies in structure or attributes [14,11]. The matching process searches for a mapping that minimizes the so-called *edit distance* between the model and any data subgraph, which reflects predefined costs associated with particular discrepancies. However, this increases matching complexity; e.g., 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 [11]. Typically, this added cost is incurred for every model matched.

As an alternative to error tolerant matching, we propose a two-stage approach to matching an ideal model graph to non-ideal data. The first stage, termed *graph rectification*, explicitly corrects data graph for each of the possible deviations of a constituent local relation from its ideal form, e.g., failures of co-termination (see Fig. 5). This process is an application of general *perceptual organization* 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.



**Fig. 5.** Two lines just meeting at a corner (a) give data graph (b), but overshoot (c) results in graph (d). Graph rectification operations applied to (c, d) produce a graph identical to (b).

We distinguish three classes of graph rectification operations: augmentation, reduction, and elaboration. *Augmentation* operations add new nodes or edges to  $G_D$ . E.g., in *proximity linking*, an edge is added where two free ends just failed to meet. *Reduction* operations remove elements from  $G_D$ . E.g., *spurious segment elimination* removes nodes representing short curve segments generated when other segments just fail to coincide at a common junction. *Elaboration* operations add subgraphs to  $G_D$  that constitute alternative descriptions of substructures already in  $G_D$ , to address locally ambiguous cases. E.g., if two segments satisfy a smooth continuation constraint, *continuity tracing* adds a new node to  $G_D$  that represents the alternative interpretation as a single curve.

The next section introduces methods for constructing and rectifying the data graph. Section 3 develops a constraint-based scheme for subgraph matching. Section 4 presents experimental results.

## 2. Data graph construction and rectification

**Graph construction.** The data graph,  $G_D$ , is constructed using standard image processing operations. The image is binarized and thinned; simple curve segments are extracted by tracing and then split into smooth segments at salient corner points. (Corners are detected based on a technique in [12]; this involves a free parameter  $P_1$ .) For each of the resulting segments, its graph representation is added to  $G_D$ .

We now move to a more structured view of the graph representation of a segment. It 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).

$G_D$  is a natural description of a curvilinear input scene for matching purposes in that it mirrors the form of our configuration models; i.e., it makes explicit the end-to-end connectivity relations among line segments. Given noise, sloppy drawing, and background context, however, it may have three shortcomings. First, it may be *insufficiently connected* due to failures of one line to terminate precisely at an end or interior point of another line by a drawing undershoot, leaving a gap. Second, it may contain *spurious segments* arising from: (i) failure of one line to terminate precisely at the end of another line by a drawing overshoot, creating a spurious crossing or T-junction; (ii) failure of two lines to terminate precisely at the same point of a third line; and (iii) thinning artifacts. Third, it may be *over-segmented*: its segments may be in many-to-one rather than one-to-one correspondence with lines of the model, due to line intersections. Graph rectification operations are introduced below to address each of these problems. Free parameters of the method are enumerated as they arise.

**Graph augmentation.** To deal with gaps, links are added to  $G_D$  based on proximity relations. One extreme possibility would be to simply make  $G_D$  totally connected, but this would make matching prohibitively costly. Ideally, the data graph would contain all links required by the model graph and as few extra ones as possible.

*End-to-end proximity linking.* A proximity link is an edge added to  $G_D$  based on proximity of two curve free ends. Links are added in increasing order of length until the graph becomes connected. As candidates for addition, we may use the set  $L$  of all  $n^2$  possible links between  $n$  free ends. Various possible optimizations on this candidate set produce similar results. One is the set of edges of the Delaunay triangulation of the set of free ends, which can be computed in  $O(n \log(n))$  time. Alternatively, we may use only those links in  $L$  between ends whose underlying image connected components are Voronoi neighbors. The results in this paper are based on this latter approach.

The process of adding links is a modification of Kruskal's algorithm for constructing the minimum spanning tree (MST) [4], resulting in a 'close-to-minimal spanning graph' of the base graph. (The MST itself would be insufficiently connected for matching: it would exclude some structurally important proximity relations, since it cannot contain a loop; see Fig. 6.). At each step a candidate link,  $l$ , is added either if (i) it connects two ends that are not yet connected (as in Kruskal's

algorithm) or (ii) at least one of  $l$ 's ends,  $e$  is, not associated with a previously added link, and the length of  $l$  is within a factor  $P_2$  (a free parameter), of the length of the curve segment containing  $e$ . For  $n$  candidate links the complexity of this step is  $O(n \log(n))$ .



**Fig. 6.** Minimum spanning tree (MST) and augmented MST of a configuration.



**Fig. 7.** Link closure counteracts variability in relative link lengths.

*End-to-interior proximity linking (virtual junction detection).* A virtual junction is a non-end point of a segment,  $s$ , that is sufficiently near a free end,  $e$ , and whose distance to  $e$  is a local minimum among points of  $s$ . The graph is modified to reflect a virtual junction  $j$  by splitting  $s$  into two segments  $s1$ ,  $s2$ , and adding co-termination links relating  $s1$ ,  $s2$  and the segment containing  $e$ . Virtual junctions may be detected using the Delaunay triangulation of the union of the free ends and evenly sampled non-end curve points, in a manner similar to the detection of proximity links.

*Link closure.* Even after proximity linking,  $G_D$  may still be insufficiently connected, due to differences in relative link lengths between the input instance and the model. E.g., the head and arms are each linked to the torso in the model, but in Fig. 6 (b) the head-torso link is missing. We address this problem by computing the transitive closure of links in  $G_D$  and then augmenting  $G_D$  with those links in the closure that it does not already contain (Fig. 7). (This operation was previously proposed in [13], in a somewhat different context.)

### Graph reduction.

*Spurious segment elimination.* A spurious segment,  $s$ , results from two segments just failing to coincide at a common junction. This creates two nearby junctions in the skeleton, bridged by  $s$ . When  $s$  is removed from  $G_D$ , the junctions are merged into one, and appropriate new co-termination links are added. Deciding which segments are spurious is a classic scale selection or noise estimation problem that we have not

addressed in a definitive way. Our current implementation is an *ad hoc local elimination* technique, in which a segment is removed if it is shorter than some factor (parameter  $P_4$ ) of the length of the longest segment terminating at a common junction.

**Graph elaboration.** Graph elaboration deals with local ambiguity in the input. We have implemented one elaboration process, *continuity tracing*, which addresses the ambiguity associated with smooth continuation of lines at crossings and T-junctions.

*Continuity tracing.* Suppose a local colinearity relation is established between co-terminal curve segments  $s_1, s_2$  in  $G_D$ . (We do this by applying a threshold, free parameter  $P_5$ , to the angle between their co-terminal ends). A new segment,  $s_3$ , is formed by merging  $s_1$  and  $s_2$ , and it is added to  $G_D$ , along with links connecting  $s_3$  to any other nodes to which  $s_1, s_2$  are connected. Specifically, the graph representation of  $s_3$  consists of copies of the two non-co-terminal nodes of  $s_1, s_2$ , connected by a new bond. The copy  $n_b$  of a node  $n_a$  is in turn assigned copies of all the links of  $n_a$  (but not of  $n_a$ 's bond). Each link of  $n_a$  and its copy in  $n_b$  are related by mutual exclusion; i.e., a valid match of a model graph to  $G_D$  may not involve more than one of these edges. This fact is made explicit by assigning these links a common label: the unique identifier of the mutual exclusion set to which they both belong. This mutual exclusion relation applies transitively if  $n_b$  is itself copied later.

A step of *continuity tracing* involves applying the preceding operation to all pairs of collinear segments at every junction in the scene. By iterating this tracing step to convergence, all sequences of pair-wise collinear segments are added to  $G_D$ .

The idea of adding alternate subgraphs to the data graph, with a mutual exclusion relation holding among links that the alternates have to common nodes, is general. It may be applied to various other local relations where they are ambiguous, including co-termination at corners and colinearity at gaps.

### 3. Constraint-based subgraph matching

**Formulation as a CSP.** Our subgraph matching problem may be formulated as a constraint satisfaction problem (CSP) as follows. Mirroring the data graph, the graph representation of a limb in the model is 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  $d_1, d_2$ . This requires some spelling out because of the possibility of missing parts. Specifically, if  $l$  is a bond, then the constraint is satisfied if (i)  $m_1$  and  $m_2$  are both null; or (ii)  $d_1$  and  $d_2$  are connected by a bond. If  $l$  is link, then the constraint is satisfied if one of four conditions hold: either (i)  $m_1$  and  $m_2$  are both null; (ii)  $m_1$  is null and  $d_2$  has no links; (iii)  $m_2$  is null and  $d_1$  has no links;

or (iv)  $d_1$  and  $d_2$  are connected by a link. (The reason for conditions (ii) and (iii) is that a model part should not be assigned null when there is support for it in the data.)

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 an isolated instance, but spurious matches arise if the data contains noise, self-intersections, or background clutter. To find reliable matches in such cases, other 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. (The effect of progressively enabling these terms is illustrated in Fig. 12.)

To handle ambiguity, two global mutual exclusion constraints and an objective function term 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 objective function term (whose weight is free parameter  $P_6$ ) in effect preference ranks curve segment chains according to the number of segments they contain. This emulates the fact that long smooth curves are normally perceived as more salient than their constituent segments.

**Implementation.** One of our implementations for solving the above CSP employs a standard branch-and-bound state-space search framework. A state in the search space 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 function* of a search process is given a state taken from the search queue and returns a set of new states to be added to the queue. The successor function selects an unassigned CSP variable and creates a new state for each possible assignment of this variable, applying the specified constraints in the process to effect possible reductions in the set of new states generated.

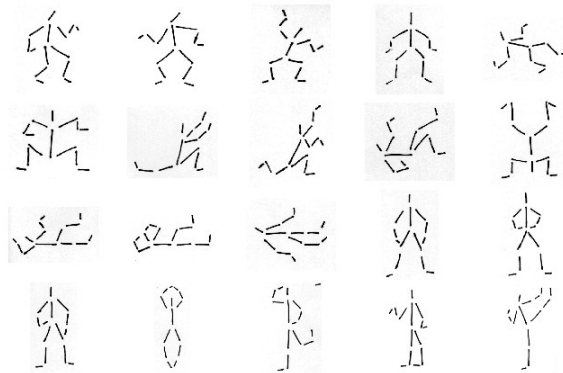
In this design, the additional criteria may be applied by incorporating them into the objective function that is optimized by the search process. The three optimization criteria are combined into a single function as a weighted sum (the weights being parameters  $P_7$ ,  $P_8$ ,  $P_9$ ). The link length and part proportion criteria have equal weights while the part count criterion has a much higher weight; i.e., the first two criteria only come into play among solutions with equal part counts.

This scheme will find only one instance of a given model in  $G_D$ . To find multiple instances, or to match multiple models, individual matches are done sequentially, removing the matched nodes and any associated links/bonds from  $G_D$  at each step.

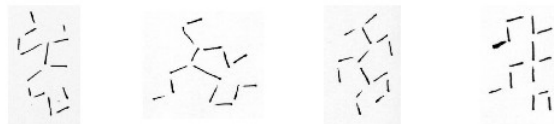
We also implemented this basic matching scheme in a concurrent constraint programming formulation, expressed using the `clp(FD)` library and corouting in SICStus Prolog [2]. A detailed description of this implementation is given in [7]. The two implementations give similar performance. The performance results given below are for the Prolog implementation running on a 600MHz Pentium III processor.

#### 4. Experiments

We developed the matching scheme using five parallel sets of 24 example images each. Each set contains 20 images that contained an isolated stick figure in a distinct pose, as well as four images of a non-figure (see Figs. 8, 9). Non-figures are visually similar to figures with respect to local relations and number of lines. The example sets are parallel in the sense that the figure images are in one-to-one correspondence across them, with respect to posture, in the case of figures, or configuration, in the case of the non-figures. Three of the sets each emphasize a particular local relation: neat co-termination, line undershoot, and line overshoot, respectively. The other two sets combine all three relations at two degrees of sloppiness.



**Fig. 8.** The example set emphasizing line undershoot.

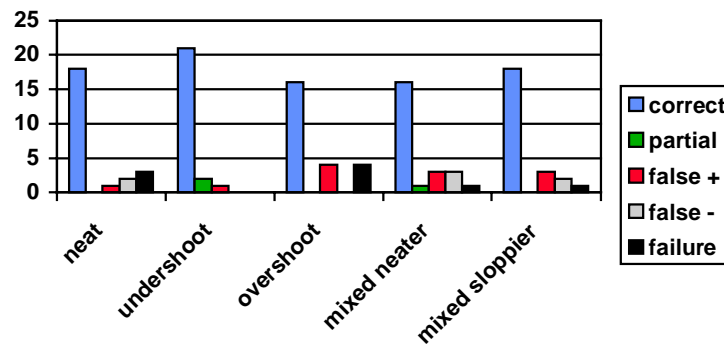


**Fig. 9.** Non-figure examples from the example set of Fig. 8.

The free parameters of the matcher were tuned by hand to give roughly equal matching accuracy across all five example sets. Manually scored accuracy results,

shown in Fig. 10, indicate that, with its parameters set favorably, the matching scheme can cope with highly variable configurations of a given model. Using this tuned performance as a baseline, our formal evaluations of the approach have focused so far on how runtime and accuracy scale with data graph complexity. (We plan to also compare matching accuracy on figures drawn by human subjects against these baseline results.)

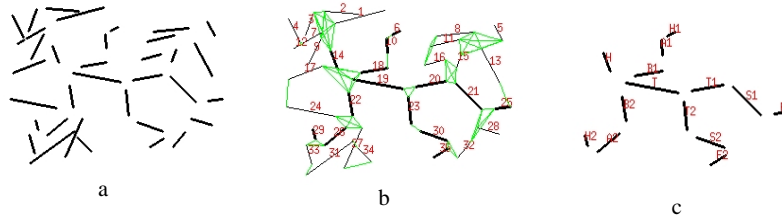
Input complexity was varied in two ways: by adding distractor lines to images of isolated figures, and by composing multiple figure scenes from the figures in the example sets. In both cases, errors were counted automatically, in terms of the agreement between the altered and unaltered cases of the line labels assigned by the matcher. The figures used for these experiments were those of the undershoot example set, because its baseline accuracy results were nearly perfect.



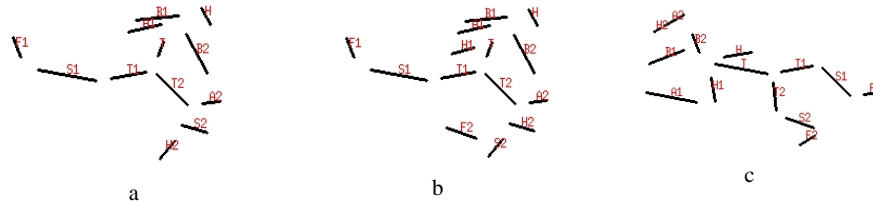
**Fig. 10.** Tuned performance on development example sets. Vertical axis is number of examples with a given score. *Correct*: every line label correct; *partial*: at most 4 line labels incorrect; *false +*: matcher gave a solution for a non-figure; *false -*: matcher gave no solution for a figure; *failure*: more than 4 lines labels incorrect.

**Effect of distractor lines.** Randomly generated distractor lines were added to the undershoot example images subject to two constraints: they were not allowed to cross the figure lines, since the figures were themselves disconnected; and their lengths were in a similar range as the figure lines. (Fig. 11 shows an example figure with distractor lines, its data graph, and the associated matching result.) A condition of the experiment was the number of distractors, which ranged from zero to 28. For each condition, the mean and standard deviation of runtime, in milliseconds, and match error, in number of mislabeled lines, were measured across all 24 example images at ten repetitions each (Fig. 13). A different set of distractors was generated for each image at each repetition. (The maximum of 28 distractors triples the number of nodes in the data graph, compared to an isolated stick figure.) For up to about ten neighboring distractors --a reasonable limit for realistic scenes -- the runtimes are clearly in a useful range. Overall growth in runtime is acceptable, considering the potentially exponential nature of this problem. The error rate appears to increase only linearly with the number of distractors.

**Performance on multiple instances.** Multi-figure scenes were made by selecting undershoot example images at random and composing them into a single new image subject to the constraint that figure lines should not cross. Fig. 15 shows an example composite image containing three stick figures, with the links and node identifiers of its data graph superimposed; Fig 16 shows the matching result. A condition of this experiment was the number of constituent figures, which ranged from one to five. The mean and standard deviation of runtime, in milliseconds, and match error, in number of mislabeled lines, were measured over twenty repetitions (Fig. 14). Runtimes are comparable to those in the previous experiment; these results suggest that performance should be in a useful range for realistic scenes containing a small number of target figures. For large data graphs, however, there is a trend toward rapid growth, and the error rate soon becomes unacceptably large. This indicates a clear need for additional computations to focus the matching process on appropriate subsets of a complex scene.



**Fig. 11.** A stick figure with 20 distractor lines (a), the corresponding graph with labels and links produced by the image analysis stage (b), and the interpretation found by the matching process (c).



**Fig. 12.** Three stick figure matches found in the data of Fig. 11 with progressively more objective terms: a) no optimization, b) maximal part count term only, c) maximal part count and optimal part proportion terms only.

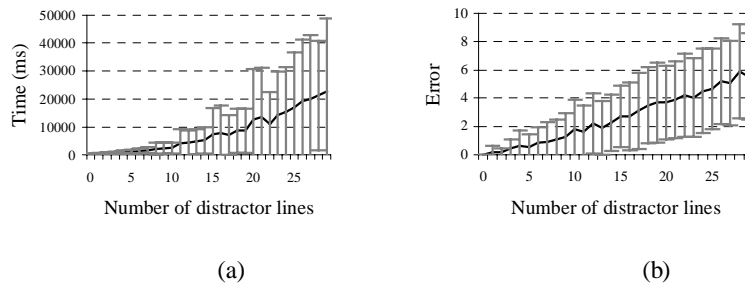
## 5. Discussion

Our research goal is a recognition system for simple, general curvilinear configurations of the kind routinely found in everyday graphic communications. Sloppiness and other types of variability characteristic of human drawing

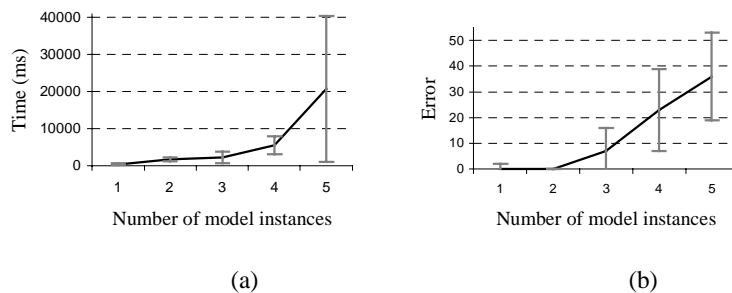
performance constitute a wide gulf between how people express themselves graphically and what current machine interpretation techniques can make sense of.

Where should the knowledge and computation needed to counter variability reside? The answer should not be the model, for that would hinder extending the system to new configurations. It should also not be the matcher, we feel, because that increases matching cost for a problem that is already dangerously complex computationally, and because it requires specialized matching machinery.

Instead, we located the knowledge about common local geometric variations and the processing to account for them in a dedicated prior process – graph rectification -- within a structural matching formulation. A constraint-based approach to matching suggests itself because it mates well with declarative modeling, and because constraint propagation and branch-and-bound search make effective use of the natural structure of this domain. But matching could be accomplished using various other technologies as well.



**Fig. 13.** Mean match times and errors for 0 through 28 distractor lines. For each distractor-count, the standard deviation over 200 runs is shown as an error bar.



**Fig. 14.** Mean match times and errors for 1 to 5 model instances. For each instance-count, the standard deviation over 10 runs is shown as an error bar.

Graph matching is a well-established approach to recognition of structural models and there is previous work in which explicit correction of an initial graph has been employed prior to matching [8]. We are not aware of a previous application of this technique to the domain of hand-drawn sketches or diagrams. Messmer [11] gives an analysis of possible “distortions” in input drawings similar to our analysis of local



There are three main avenues to be explored in our ongoing work. First, our assessment of the approach in this paper has been preliminary, limited to how it scales under varying scene complexity. Assessing the robustness of the scheme to variations in drawings collected from human subjects is a key next step. Second, the approach must be exercised on scenes containing instances of a variety of configuration models; the stick figure model was simply an entry point to this domain. Putting multiple models simultaneously in play raises issues of control and model indexing that we have not touched at all so far. Finally, the method involves a number of free parameters that would not in general be easy to tune by hand. It will be important to develop techniques for automatically setting them based on example data.

**Acknowledgements.** We are grateful to David Fleet, Eric Saund, and Dan Lerner, of the Perceptual Document Analysis Area at Xerox PARC, for helpful discussions over the course of this research.

## References

1. D. Bainbridge and T. Bell. An extensible optical music recognition system. In Proc. Nineteenth Australasian Computer Science Conf., 1996.
2. M. Carlsson, G. Ottosson, B. Carlson. An open-ended finite domain constraint solver. Proc. Programming Languages: Implementations, Logics, and Programs, 1997.
3. S. Chok, K. Marriot. Parsing visual languages. Proc. 18th Australasian Computer Science Conf., 27(1), 1995: 90–98.
4. T. Cormen, C. Leiserson, R. Rivest. Introduction to Algorithms. Cambridge, MA: M.I.T. Press, 1990.
5. B. Couasnon, P. Brisset, I. Stephan. Using logic programming languages for optical music recognition. Proc. 3rd Int. Conf. on Practical Application of Prolog, Paris, 1995.
6. 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.
7. 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.
8. D. Isenor and S. Zaky. Fingerprint identification using graph matching. Pattern Recognition, vol. 19, no. 2, 1986, pp. 113-122.
9. J. Larrosa and G. Valiente. Constraint satisfaction algorithms for graph pattern matching. Under consideration for publication in J. Math. Struct. in Computer Science, 2001.
10. 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
11. B. Messmer. Efficient graph matching algorithms for preprocessed model graphs. PhD thesis. Bern Univ., Switzerland, 1995.
12. E. Saund. Perceptual organization in an interactive sketch editor. 5th Int. Conf. on Computer Vision. Cambridge, MA: 1995: 597–604.
13. E. Saund. Perceptually closed paths in sketches and drawings. Submitted for publication.
14. L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. In IEEE PAMI, vol. 3, no. 5, Sept. 1981, pp. 504-519.