

# Handling ambiguity in constraint-based recognition of stick figure sketches

James V. Mahoney and Markus P. J. Fromherz

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

## ABSTRACT

Even seemingly simple drawings, diagrams, and sketches are hard for computer programs to interpret, because these inputs can be highly variable in several respects. This variability corrupts the expected mapping between a prior model of a configuration and an instance of it in the scene. We propose a scheme for representing ambiguity explicitly, within a subgraph matching framework, that limits its impact on the computational and program complexity of matching. First, ambiguous alternative structures in the input are represented explicitly by coupled subgraphs of the data graph, using a class of segmentation post-processing operations termed *graph elaboration*. Second, the matching process enforces mutual exclusion constraints among these coupled alternatives, and preferences or rankings associated with them enable better matches to be found early on by a constrained optimization process. We describe several elaboration processes, and extend a straightforward constraint-based subgraph matching scheme to elaborated data graphs. The discussion focuses on the domain of human stick figures in diverse poses.

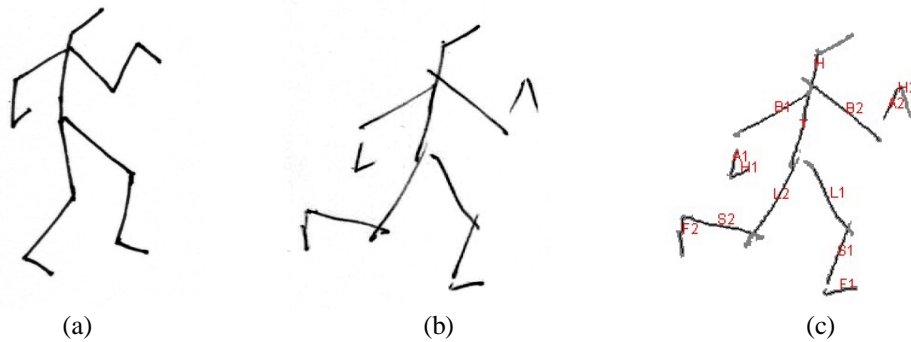
Keywords: Diagram understanding, sketch recognition, segmentation ambiguity, subgraph matching, graph elaboration.

## 1. INTRODUCTION

It is hard to write computer programs that recognize reliably even seemingly simple classes of drawings, diagrams, and sketches. 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: variability in the drawing process (Fig. 1); noise in the process that captures the drawing; variability in the shape or configuration of the figures themselves (Fig. 2); and variability arising from interaction of figures with background context (Fig. 3). By variability in the drawing process, we mean the inevitable discrepancies that arise even when someone attempts to redraw the same figure in the very same configuration or pose. Very often, for example, hand-drawn lines overshoot or undershoot rather than meeting at a point as intended. Noise in capturing the drawing may introduce gaps and extraneous marks. The class of figure to be recognized may have widely varying manifestations if it is articulated – i.e., if the angles between connected parts may vary – or if it is flexible or deformable in shape. Interaction with context refers to overlap, intersection, coincidence, or alignment of a figure's parts with neighboring marks (Fig. 4).

In all these forms, the negative effect of variability on matching is that it corrupts 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 or shape elements specified in the configuration model. Unfortunately, given straightforward segmentation schemes, the sources of variability discussed above often cause this mapping to become many-to-one or one-to-many. For example, if segmentation simply divides the input scene 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.

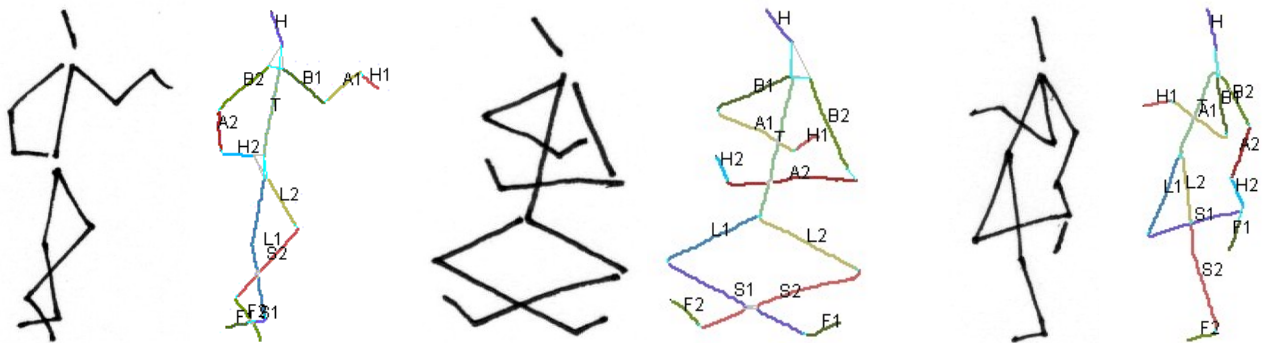
In this paper, the term *ambiguity* (or *segmentation ambiguity*) refers specifically to the absence of a one-to-one mapping between the elements produced by segmentation and the elements of the shape/configuration model, and the lack of any bottom-up (i.e., model-independent) basis for establishing such a one-to-one mapping. We propose a scheme for representing ambiguity explicitly, in the context of a subgraph matching recognition framework, so as to limit its impact on the computational and algorithmic complexity of the recognition process. Our initial exploration of this problem focuses on the domain of human stick figures in diverse poses.



**Fig. 1.** (a, b) Neat and sloppy stick figures. In sketching, failures of co-termination are a frequent variation from the ideal form. (c) Example matching result. Labels denote model parts: Head, Torso, Biceps<sub>1</sub>, Arm<sub>1</sub>, etc.

In a nutshell, there are two aspects to our proposal. First, ambiguous alternative structures in the input are explicitly represented by coupled subgraphs of the data graph, by a class of segmentation post-processing operations we call *graph elaboration*. Second, the matching process enforces mutual exclusion constraints among these coupled alternatives, and preferences or rankings associated with the alternatives enable better matches to be found early on by a constrained optimization process. We show how to implement several elaboration processes, and how a straightforward constraint-based subgraph matching scheme can be extended to accommodate elaborated data graphs.

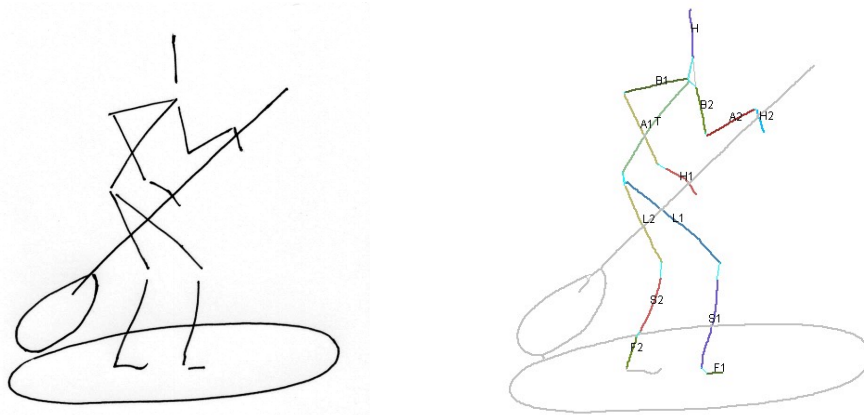
The next section introduces a graph-based framework for structural modeling of curvilinear configurations that are defined by end-to-end connectivity relations and length proportions. Section 3 outlines our proposal for addressing ambiguity in this recognition task. Section 4 presents details of the algorithms used to prepare the data graph. Section 5 develops a constraint-based scheme for subgraph matching. Section 6 presents experimental results.



**Fig. 2.** Matching must cope with self-crossings, due to articulation. Labels denote model parts: Head, Torso, Biceps<sub>1</sub>, Arm<sub>1</sub>, etc.

## 2. GRAPH-BASED STRUCTURAL MODELING FOR STICK FIGURE CONFIGURATIONS

Because the configurations found in diagrams and sketches are often articulated, or are defined by abstract spatial relations among their constituent lines, the task of recognizing them lends itself to a structural modeling and matching 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. 3.** Matching must cope with variations, e.g., crossings, due to background context.

**The model graph.** 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 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

```

**The data graph.** 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 [12]; this involves a free parameter  $P_I$ .) 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).

$G_D$  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 several 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* in some areas: its segments may be in many-to-one correspondence with lines of the model, due to line intersections and

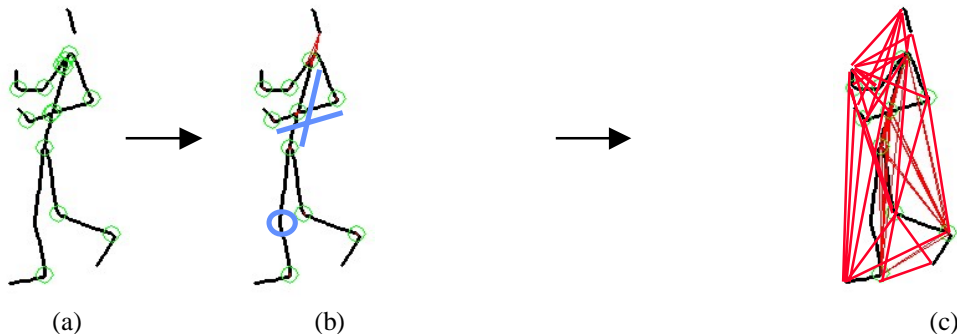
gaps. Fourth, it may be *under-segmented* in some areas: its segments may be in one-to-many correspondence with lines of the model, e.g., due to corner-detection failures.

### 3. GRAPH ELABORATION

Due to drawing variability and noise, the initial data graph,  $G_D$ , would rarely contain a verbatim instance of the model as a subgraph; the mapping from data nodes to model nodes is many-to-one in some areas, one-to-many in others. The prevalent approach to dealing with this ambiguity in the literature is to use *error-tolerant* subgraph matching to explicitly account for discrepancies in structure [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, the extension from straightforward subgraph matching to error-tolerant matching increases matching 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 [11]. This added cost is incurred for every model matched, unless the domain is such that a great deal of structure is shared among the models.

It would be possible to address the segmentation problems with  $G_D$  by brute force means. E.g., the connectivity issue could be dealt with by making  $G_D$  totally connected. Similarly, we could blindly add many of the possible concatenations of simple segments to  $G_D$ . The appeal of the brute force approach is that it offers all possible alternatives to the matching process – no interpretations are ruled out bottom-up. However, subgraph matching on the resulting graph would almost certainly entail the exponential worst-case complexity.

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 elaboration*, addresses ambiguity by adding subgraphs to  $G_D$  that constitute plausible alternative descriptions of substructures already in  $G_D$ . 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 (Fig. 4). 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 those solutions first whose elements have higher preferences or rankings. (This proposal generalizes one we made in [10] that emphasized *rectification* of the initial graph; that notion included in the idea of graph elaboration, but in a much more limited role than it is given here.)



**Fig. 4.** (a) Straightforward segmentation based on local processing is insufficiently connected, over-segmented, and under-segmented. (c) Naïve elaboration, e.g., making the graph totally connected, makes matching intractable. (b) Selective elaboration, guided by P.O. principles, produces a graph consisting of a manageably small superset of the model links and nodes.

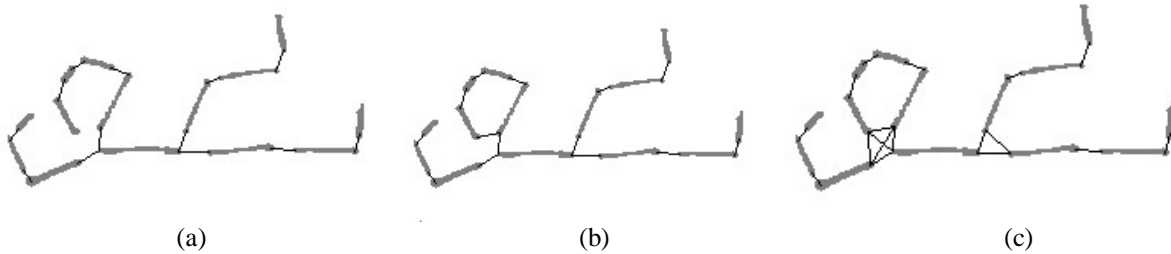
We have explored five graph elaboration operations in particular. *Proximity linking* in effect fills gaps in the data; an edge is added to  $G_D$  between two free ends if they satisfy a proximity constraint. *Virtual junction splitting* adds proximity links between free ends and neighboring non-end points. *Spurious segment jumping* enables the matcher to ignore nodes representing short curve segments generated when other segments just miss coinciding at a common junction. 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. *Link closure* adds redundant co-termination links to reduce sensitivity to small distance variations. Details of these techniques are given in the following section.

#### 4. SOME ELABORATION OPERATIONS

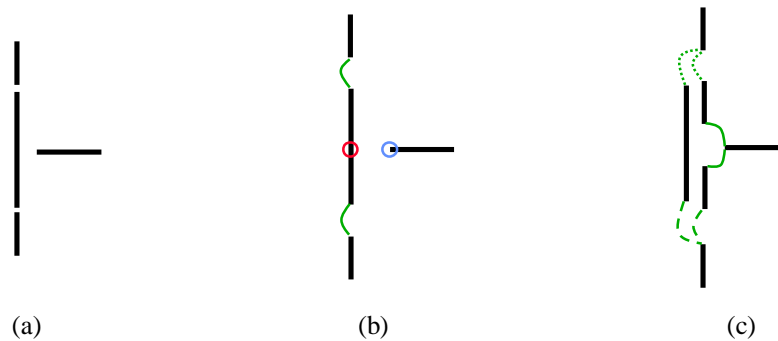
**Proximity linking.** Proximity linking deals with the problem of missing links in  $G_D$  due to gaps or line undershoot in the data. A link between two nodes is added to  $G_D$  if the associated free ends satisfy a proximity constraint. Note that adding the link obliterates the original structure, in which the two nodes are *not* linked. However, the fact that the matcher does *subgraph* matching means that it can ignore data links, so the unlinked structure is implicitly available to it. In this respect, proximity linking is unlike other elaboration operations introduced below, which, in adding alternative structures to  $G_D$ , leave the original structure intact and establish explicit mutual exclusion relations between the original and alternate structures.

Proximity linking adds edges to  $G_D$  based on proximity of pairs of 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 connected components in the image 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]. The MST itself would be insufficiently connected for matching: it would exclude some structurally important proximity relations, since it cannot contain a loop (Fig. 5); the modification allows some redundant links to be added, resulting in a 'close-to-minimal spanning graph' (CSMG) of the initial graph. 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 process is  $O(n \log(n))$ .



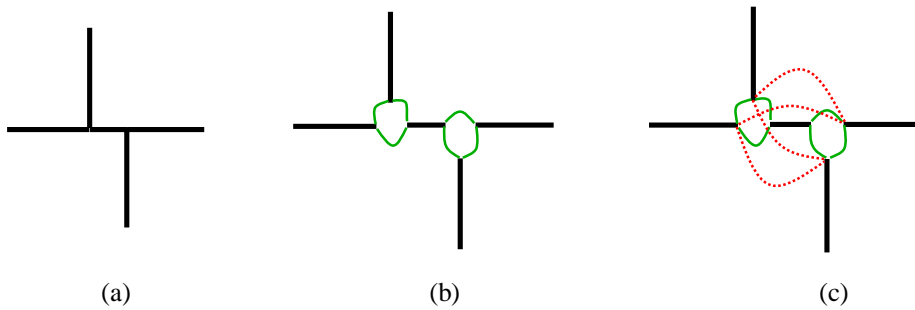
**Fig. 5.** Proximity linking: MST (a) and CSMG (b) of a disconnected figure. Link closure (c) counteracts variability in relative link lengths.



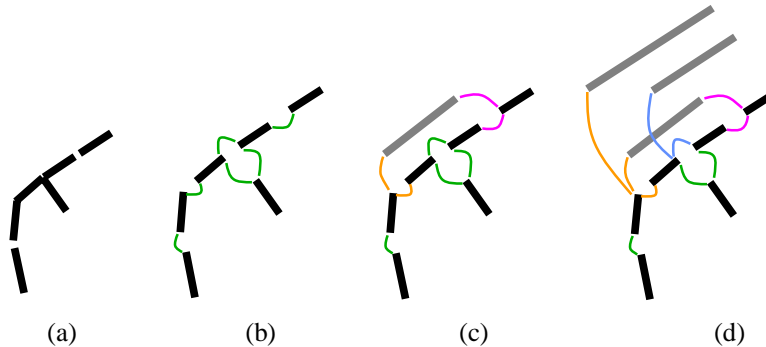
**Fig. 6.** Virtual junction splitting. (a) Input configuration. (b) Exploded view showing links defined before junction splitting. Circle on left indicates virtual junction point detected in relation to free end circled on right. (c) Two new segments are added with links to the free end and also to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

**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. 5 (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 5(c)). (This operation was previously proposed in [13], in a somewhat different context.)

**Virtual junction splitting.** A virtual junction is a non-end point  $j$  of a segment,  $s$ , that is sufficiently near a free end,  $e$ , and whose distance to  $e$  is a local minimum among points of  $s$ . (Parameter  $P_2$  is again used in this proximity determination.) The alternative graph structure associated with  $j$  consists of two new segments  $s_1, s_2$ , which may be thought of as the result of splitting a copy of  $s$  at  $j$ , along with some new co-termination links (see Fig. 6).  $e$  is linked to the end of  $s_1$  ( $s_2$ ) that is coincident with  $j$ . Also, the end of  $s_1$  ( $s_2$ ) not coincident with  $j$  is linked to any other segment(s) to which  $s$  is linked at that end.  $s_1$  and  $s_2$  are mutually exclusive with  $s$ ; i.e., if  $s$  participates in a match  $s_1$  ( $s_2$ ) may not, and vice versa. 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 process similar to that used to add proximity links.



**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.



**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.

**Spurious segment jumping.** A spurious segment,  $s$ , is generated when two segments just miss meeting at a common junction point. This creates two nearby junctions,  $j_1$  and  $j_2$ , in the skeleton, bridged by  $s$ . To enable the matching process to ignore  $s$ , for each segment  $s_1$  (excluding  $s$ ) terminating at  $j_1$ , co-termination links are added to  $G_D$  between the end of  $s_1$  coincident with  $j_1$  and the end coincident with  $j_2$  of each segment  $s_2$  (excluding  $s$ ) terminating at  $j_2$ . Each added link  $l_a$  is mutual exclusive with the pre-existing link  $l_p$  connecting the ends of  $s_1$  and  $s$  at  $j_1$ ; i.e., only one of them may participate in a match. 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 technique in

which a segment is jumped if it is shorter than some factor (parameter  $P_4$ ) of the length of the longest segment terminating at a common junction.

**Continuity tracing.** *Continuity tracing* addresses the ambiguity associated with smooth continuation of lines at crossings and T-junctions, i.e., possible over-segmentation (Fig. 8). Suppose a local colinearity relation is established between co-terminal curve segments  $s_1, s_2$  in  $G_D$ . (This is done by applying a threshold, free parameter  $P_5$ , to the angle between their co-terminal ends). A new segment,  $s_3$ , which may be thought of as the result of merging copies of  $s_1$  and  $s_2$ , 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 end 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 of  $n_a$ 's links (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., only one of them may participate in a valid match of a model graph to  $G_D$ . 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 number of distinct sequences that can be made from a collinear chain of  $n$  segments is  $n(n+1)/2$ . (There is one sequence of length  $n$ ; there are two of length  $n-1$ ; three of length  $n-2$ ; etc.)

## 5. CONSTRAINT-BASED SUBGRAPH MATCHING

**Formulation as a CSP.** 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  $d_1, d_2$ . 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, 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. (The effect of progressively enabling these terms is illustrated in Fig. 13.)

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 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.

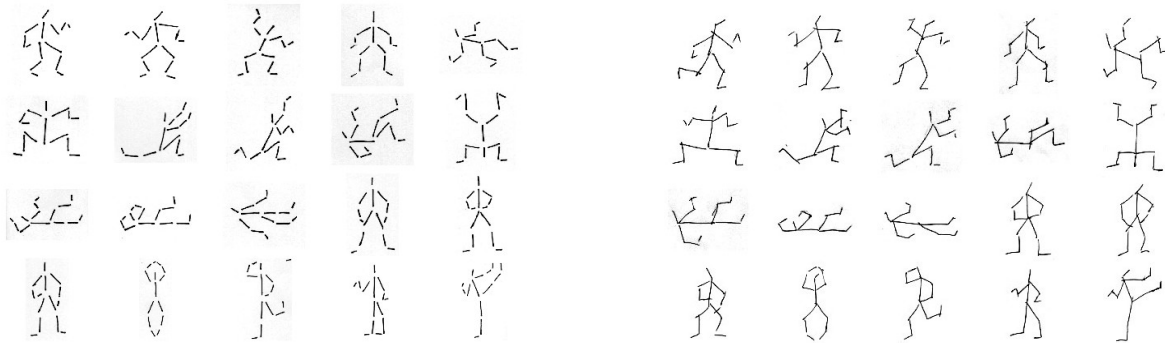
**Implementation.** One of our implementations solves the above CSP using a branch-and-bound state-space search framework written in Java. A state consists of an assignment to the set of CSP variables. (In the initial state, all

variables are unassigned. In a goal state, all variables are assigned such that all constraints are satisfied.) The *successor 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.

The quantitative criteria are incorporated into the objective function that is optimized by the search process. The four optimization criteria are combined into a single function as a weighted sum (the weights being parameters  $P_6, P_7, P_8, P_9$ ). 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. The weight of the segment count criterion is in between these other two values.

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 approach in a concurrent constraint programming formulation, expressed using the `clp(FD)` library and coroutines 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 Java implementation running on a 600 MHz Pentium III processor.



**Fig. 9.** Two example sets, emphasizing line undershoot (left) and mixture of undershoot and overshoot (right).



**Fig. 10.** Non-figure examples from the example sets of Fig. 9.

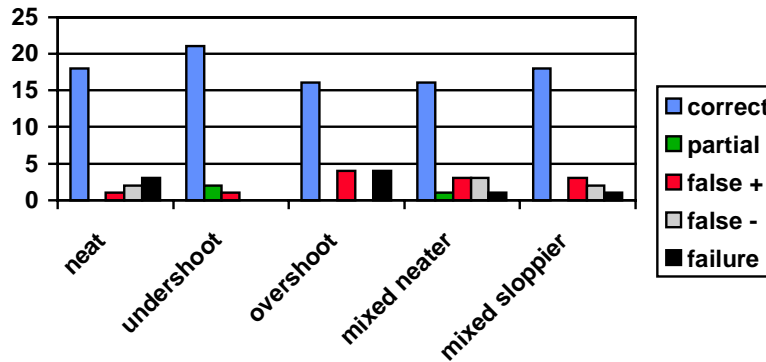
## 6. EXPERIMENTS

We developed the matching scheme using five parallel sets of 24 example images each. Each set includes 20 figure images each containing an isolated stick figure in a distinct pose, as well as four images each containing a non-figure (see Figs. 9, 10). The non-figures are visually similar to the 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.

The elaboration operations were applied in the order in which they were presented in Section 4, except that virtual junction splitting was not enabled for (or relevant to) these experiments. The free parameters of the method were tuned by hand to give roughly equal matching accuracy across all five example sets. Manually scored accuracy results, shown in Fig. 11, indicate that, with its parameters set favorably, the approach 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.

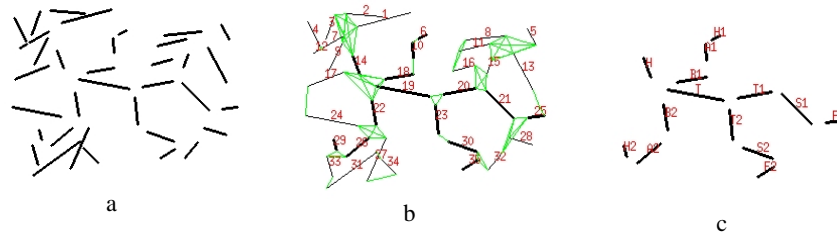


**Fig. 11.** Tuned performance on development example sets. Vertical axis is number of examples with a given score. *Correct*: every line label correct; *partial*: at most 4 line labels incorrect; *false +*: matcher gave a solution for a non-figure; *false -*: matcher gave no solution for a figure; *failure*: more than 4 lines labels incorrect.

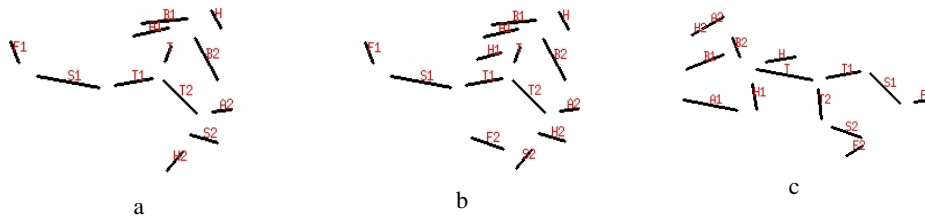
**Effect of distractor lines.** Randomly generated distractor lines were added to the undershoot example images subject to two constraints: they were not allowed to cross the figure lines, since the figures were themselves disconnected; and their lengths were in a similar range as the figure lines. (Fig. 12 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 error of runtime, in milliseconds, and match error, in number of mislabeled lines, were measured across ten example images at ten repetitions each (Fig. 14). 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 the case of 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. Note that the error rate appears to increase only linearly with the number of distractors.

Fig. 15 shows the effect of forcing the distractor lines to cross the figure lines exactly once or exactly twice. The other parameters of this experiment were as above, except that the examples were taken from the neatly drawn set, and the number of distractors ranged only from zero to nine. Although there is clearly overhead associated with the presence of crossings, the preference ranking of segment chains, made by continuity tracing, according to the number of primitive segments they contain appears to be reflected in an insensitivity to the crossing rate.

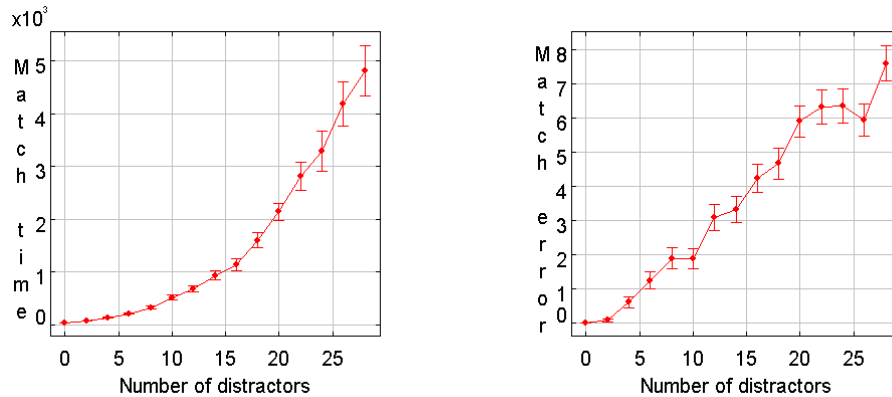
**Performance on multiple instances.** Multi-figure scenes were made by selecting undershoot example images at random and composing them into a single new image subject to the constraint that figure lines should not cross. Fig. 17 (a) shows an example composite image containing three stick figures, with the links and node identifiers of its data graph superimposed; Fig 17 (b) shows the matching result. A condition of this experiment was the number of constituent figures, which ranged from one to six. The mean and standard error of runtime, in milliseconds, and match error, in number of mislabeled lines, were measured over twenty repetitions (Fig. 16). 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. 12.** 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. 13.** 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.



**Fig. 14.** Mean match time (in milliseconds) and match error for 0 through 28 distractor lines. For each distractor-count, the standard error over 100 runs is shown as an error bar.

## 7. DISCUSSION

A drawing recognition technology must cope reliably with the variability and ambiguity that are so pervasive in drawings and sketches, and yet provide interactive performance or better. This paper presented an approach to handling various kinds of drawing variability and the segmentation ambiguities that result from them. Ambiguity was represented explicitly, by adding reasonable alternative descriptions to the base representation used for matching. Mutual exclusion relations and preference rankings among these alternatives enabled a constrained optimization process to find best matches efficiently, the potentially exponential complexity of structural matching notwithstanding.

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 parts of our analysis of local variability, but applies it within the

framework of error-tolerant subgraph matching. The work described in [6] also uses alternate subgraphs to represent alternative interpretations in a discrete relaxation framework.

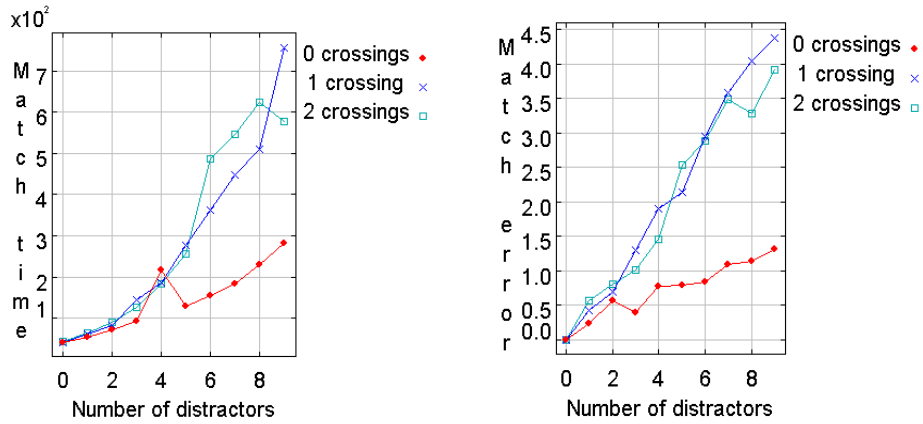


Fig. 15. The effect of crossings of distractor lines with figure lines, on runtime (in milliseconds) and error rate.

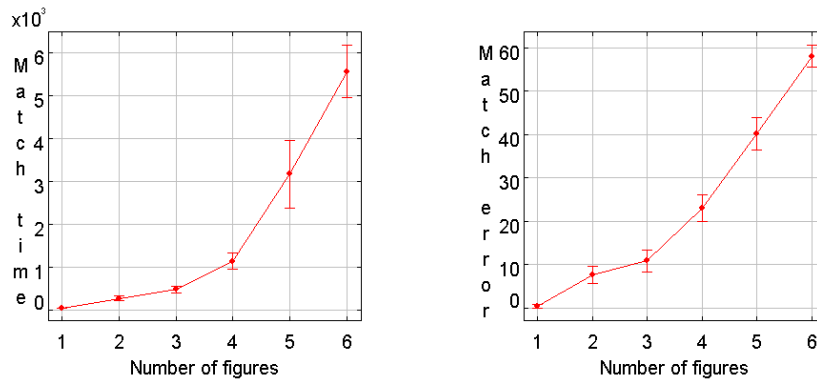
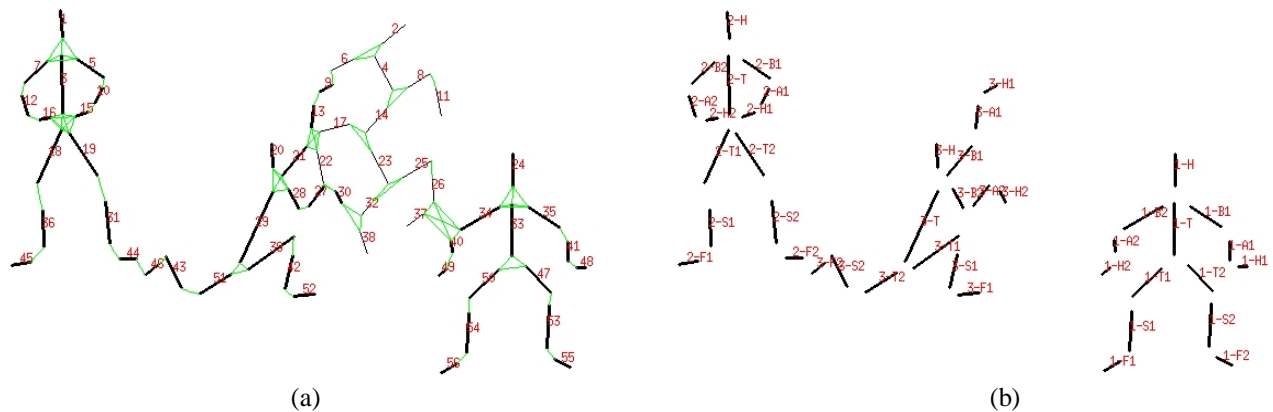


Fig. 16. Mean match time (in milliseconds) and match error for images containing 1 to 6 model instances. For each instance-count, the standard error over 20 runs is shown as an error bar.

Constraint-based pattern recognition approaches have previously been used mainly in domains with strong (visual) grammars, such as musical notation ([5],[1]) and state-transition diagrams [3]. The former two references extend Definite Clause Grammars to allow for the nonlinear composition of the graphical elements; the latter uses Constraint Multiset Grammars to similar effect. Other work has proposed dedicated forward checking and full lookahead search algorithms for subgraph matching [14][9]; these involved special-purpose algorithms that cannot be extended easily to user-defined constraints and objective functions.

There are four main avenues to be explored in our ongoing work. First, our assessment of the approach in this paper is 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 very extensive domain. Putting multiple models simultaneously in play raises issues of control and model indexing that we have not touched so far. Third, several segmentation ambiguities remain to be addressed from a graph elaboration perspective, including corners and collinear groupings. 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.



**Fig. 17.** (a) Sketch with three stick figures and a distractor figure, with labels and links produced by the image analysis stage. (b) Interpretation of the data of (a), with part labels preceded by the index of the instance.

### ACKNOWLEDGEMENTS

We are grateful to David Fleet, Eric Saund, and Dan Lerner, 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. 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.