

Perceptual organization as graph rectification in a constraint-based scheme for interpreting sloppy stick figures.

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

Extended Abstract

Machine systems for understanding hand-drawn sketches and diagrams must reliably 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. This paper argues that a perceptual organization process may impact a recognition process for such drawings in three major ways: by allowing great variability of form, by increasing matching efficiency, and by enabling easy extensibility to new configurations. Our initial focus is the domain of human stick figures in arbitrary pose. (See Figs. 1, 2.)

We adopt a structural modeling approach, suitable for highly articulated or abstract configurations. The configuration model and the input scene are represented as graphs, with nodes representing figure parts (e.g., lines), and edges representing part relations (e.g., line connections). Recognition is cast as subgraph matching of the *model graph* to the *data graph*, to allow for irrelevant parts or relations in the input. Matching is implemented using constraint propagation and branch-and-bound search in the constraint logic programming language CLP(fd) [Car97].

Consider an initial data graph, G_D , created from an image of a line drawing like Fig. 1. The nodes represent the curve segments that result from applying standard binarization, thinning, junction detection, corner detection, and curve tracing operations. Two nodes are linked by an edge 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. One solution is to use error-tolerant subgraph matching to explicitly allow and account for structural or attribute discrepancies. 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 [Mes95]. Typically, this added cost is incurred for every model matched.

In the alternative we propose, variability is characterized in terms of the possible ways that each constituent local relation of a configuration can be perturbed from its ideal form. E.g., two curve segments may fail to precisely coincide at their endpoints (an ideal corner) by either an undershoot, leaving a gap, or an overshoot, creating a spurious crossing (see Fig. 3). The data graph is explicitly corrected for these deviations in a prior perceptual organization stage, termed *graph rectification*, so as to greatly increase the chance that the model will find a direct match in the data (see Fig. 4). This process is an application of general principles such as good continuation and co-termination to the specific goal of producing a suitable data graph for matching.

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 the *proximity linking* operation, an edge may be added where two free ends just failed to meet. Candidates for proximity links are the edges of the Delaunay triangulation of the set of free ends, and they are selected for addition in a modification of Kruskal's minimum spanning tree algorithm. Another augmentation operation, *virtual junction detection*, establishes salient proximity relations between free ends and interior curve points.

Reduction operations remove elements from G_D . E.g., the *spurious segment elimination* operation removes nodes that correspond to short curve segments arising from other segments just failing to precisely coincide at a common junction. When such a segment is removed, associated junctions are merged, and appropriate new co-termination links are added to G_D . The decision criterion for this elimination process may be based on the global segment length distribution or on local relative length among co-terminal segments.

Elaboration operations add subgraphs to G_D that constitute alternative descriptions of substructures already in G_D . E.g., if two segments in G_D satisfy a smooth continuation constraint, the *continuity tracing* operation may add a new node to G_D that represents the alternative interpreta-

tion of them as a single smooth curve. Graph elaboration addresses the problem of local ambiguity (see Figs. 5, 6). For recognition by graph matching, there is local ambiguity whenever some part of the scene could be represented in multiple ways in the data graph; we distinguish weak and strong forms, as functions of the particular matching process used. An ambiguity is weak if passing only one of the possible descriptions to the matcher does not preclude a correct global interpretation. E.g., two segments meeting at a corner are always related by an edge in the data graph. The ambiguity here is that the edge suggests a structural association, whereas the segments might coincide only by accident. However, our matcher does not require every data edge to match a model edge, so this is a weak ambiguity. An ambiguity is strong if multiple descriptions must be passed to the matcher to ensure a correct interpretation; e.g., this is the case for crossing lines. The elaboration process can associate preference rankings with the alternative structures introduced, to direct the matching search toward preferred interpretations early on.

For the subgraph matching process itself, we employ a concurrent constraint programming formulation, wherein the matcher is decomposed into a number of processes or *observers*, one per node (figure part) in the model. The goal is to assign nodes (curves) in the data graph to parts in the model. Each observer waits until its curve label is determined and then constrains those parts that are linked to it in the model to sets of curves connected to its curve in the data graph. In a CLP(fd) language, this simple form of concurrency can be simulated using coroutines. As labeling proceeds, the available domains for remaining variables diminish rapidly, leading to narrow search trees. We use the branch-and-bound search in CLP(fd) to optimize a specified objective function; e.g., we may maximize number of parts matched and minimize the total proximity link length.

Subgraph matching is exponential in the worst case, but the above design gives good performance for inputs within a useful size range. We have explored recognition runtime and accuracy for a variety of input configurations: an isolated target figure; two to five targets together; targets mingled with non-targets; and targets mingled with varying numbers of random distractor lines. E.g., for isolated figures and figure pairs, the optimal matches are reliably correct and are returned in under 250 msec on a 600MHz Pentium III. With fifteen random distractor lines (doubling the size of the data graph) search times typically range between one and five seconds.

References

[Car97] M. Carlsson, G. Ottosson, B. Carlson. An open-ended finite domain constraint solver. Proc. Programming Languages: Implementations, Logics, and Programs, 1997.

[Mes95] B. Messmer. Efficient graph matching algorithms for preprocessed model graphs. PhD thesis. Bern Univ., Switzerland, 1995.

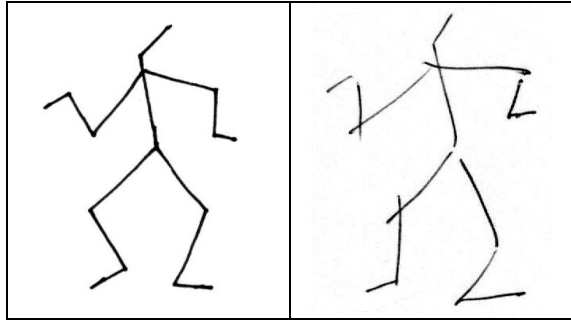


Figure 1. Neat and sloppy stick figures. Figure on left is close to ideal form described by model: curve segments are precisely co-terminal at junctions and corners.

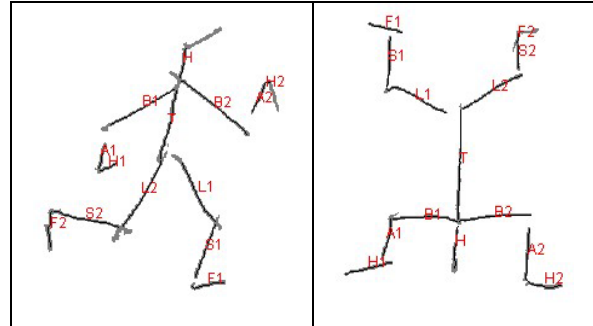


Figure 2. Example matching results. Each curve segment is labeled with its interpretation as a model part: H: head, B1: biceps1, T: torso, etc.

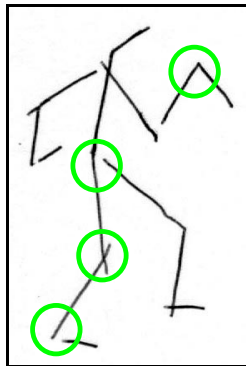


Figure 3. This figure requires (circles, top to bottom) corner detection, virtual junction detection, junction detection and spurious segment elimination, and proximity linking.

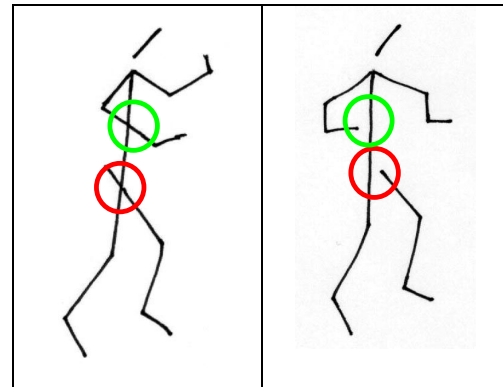


Figure 5. Strong local ambiguity: within the lower circles the curves should be broken at a junction or virtual junction. Within the upper circles they should not.

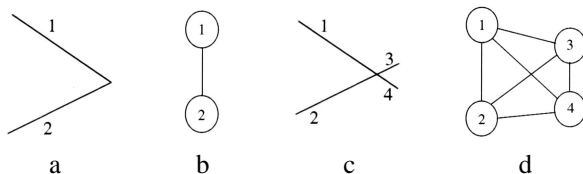


Figure 4. Two lines just meeting at a corner (a) give data graph (b), but overshoot (c) results in graph (d). Graph rectification operation applied to (c, d) produce a graph identical to (b).

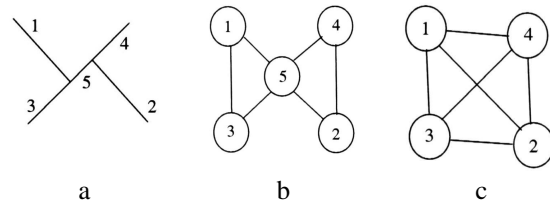


Figure 6. Input (a) gives initial data graph (b). If segment 5 is unambiguously spurious, it is eliminated to produce graph (c). Otherwise (c) is passed on to the matcher as an alternate description in addition to (b).

