A Perceptually-Supported Sketch Editor

Eric Saund and *Thomas P. Moran*
Xerox Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304, USA
saund@parc.xerox.com

ABSTRACT
The human visual system makes a great deal more of images than the elemental marks on a surface. In the course of viewing, creating, or editing a picture, we actively construct a host of visual structures and relationships as components of sensible interpretations. This paper shows how some of these computational processes can be incorporated into *perceptually-supported* image editing tools, enabling machines to better engage users at the level of their own percepts. We focus on the domain of freehand sketch editors, such as an electronic whiteboard application for a pen-based computer. By using computer vision techniques to perform covert recognition of visual structure as it emerges during the course of a drawing/editing session, a perceptually supported image editor gives users access to visual objects as they are perceived by the human visual system. We present a flexible image interpretation architecture based on token grouping in a multiscale blackboard data structure. This organization supports multiple perceptual interpretations of line drawing data, domain-specific knowledge bases for interpretable visual structures, and gesture-based selection of visual objects. A system implementing these ideas, called *PerSketch*, begins to explore a new space of *WYPIWYG* (What Your *Perceive* Is What You Get) image editing tools.

KEYWORDS: image editing, graphics editing, drawing tools, sketch tools, interactive graphics, pen computing, gestures, machine vision, computer vision, perceptual grouping, perceptual organization, token grouping, scale space blackboard WYPIWYG, PerSketch.

INTRODUCTION

Drawing is an interactive process. Whether on paper or a computer screen, the physical marks appearing on a surface support the recognition and discovery of relationships and structures that had moments before been only latent in the imagination. After executing a few strokes, one takes note of new possibilities as well as problems. Then one draws some more, either by adding to or changing the existing marks, and so on. Thus, as perceived by the user, the structure of a drawing is emergent and dynamic. In order fully to participate in this process, an ideal drawing editor tool would be able to read the user's mind (his visual system in particular) and synchronize with whatever image entities the user happens to perceive as significant. While we cannot build such an ideal device, we can adopt methods from Computational Vision to construct and make available, within an image editing program, rich sets of visual objects that better reflect the coherent spatial structures that users are likely to perceive and want to manipulate [5]. We call the resulting class of perceptually supported tools *WYPIWYG* (What You *Perceive* Is What You Get) image editors.

We explore this idea within the context of freehand line drawing editors in which the user manipulates "digital-ink" in an Electronic Whiteboard or Electronic Sketchpad application. Underlying our approach are several goals for this class of systems [6, 8] The user interface must be transparent and immediately accessible, with increased functionality coming in layers that the user can either acquire or not: one should be able to walk up and just draw, oblivious to whatever the computer underneath is doing. The user shouldn't have to worry about whether the computer recognizes something correctly or not. Most work should be done directly on the drawing (e.g. without having to deal with menus).

Existing image editing programs are of two types. *Paint*-style programs let one create any possible image, but at the cost of working at the level of either individual pixels or, at best, crudely defined collections of pixels. *Structured graphics*-style programs let one create abstract objects, such as ellipses and rectangles; but once created these objects must be dealt with literally and cannot be dissembled or composed into new objects. In both cases the grain size of user-accessible objects rigidly constrains the set of image modifications easily available to the user at any given time. It is commonplace for users to experience frustration when they want to make apparently simple changes to an image that the editing tools just do not allow them to perform.

Existing digital-ink-based drawing systems most closely resemble structured graphics editors. The units of manipulation are *strokes*, defined by the path of the pen from the time it touches the surface of the display until it is lifted. Often, however, users wish to manipulate not the strokes as they were originally drawn, but objects emergent from the raw markings. See Figure 1.

The issue is: what happens when perceptually salient struc-

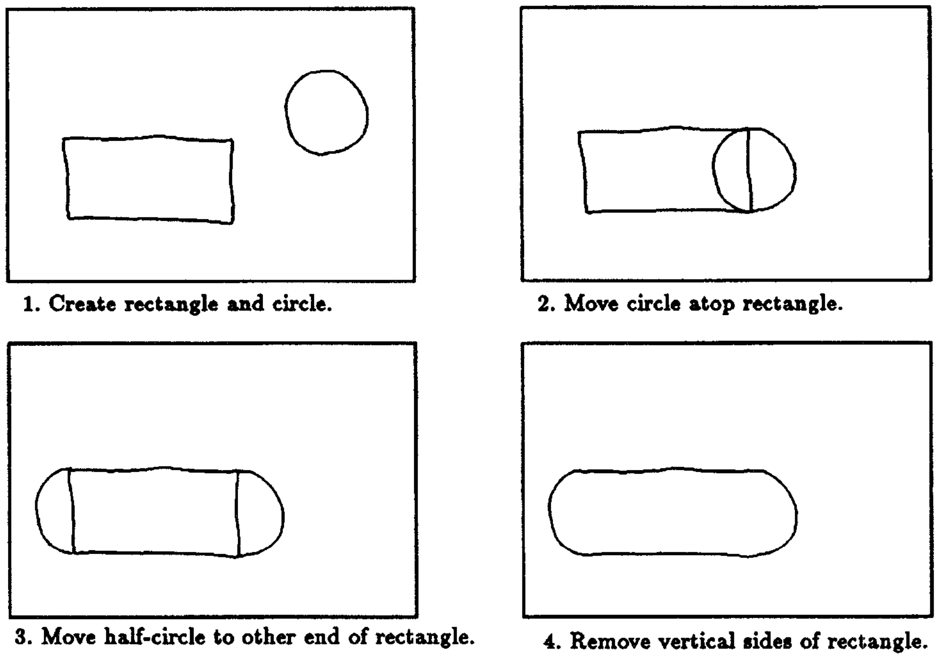

1. Create rectangle and circle.
2. Move circle atop rectangle.
3. Move half-circle to other end of rectangle.
4. Remove vertical sides of rectangle.

Figure 1: A sequence of drawing creation and transformation steps easy to visualize and describe, but not supported under conventional structured graphics or "ink" drawing editors.

ture occurs at the level of (1) fragments of ink strokes, or (2) collections of several strokes? Few existing systems address these possibilities, although some high-end commercial graphics editors (not ink-based systems per se) do permit such structure to be made explicit through a cumbersome process of converting curves to a spline representation, selecting breakpoints, fragmenting the curves, selecting fragments, then reconstructing new curves.

Our work borrows from computer vision in two ways to support user access and manipulation of visually apparent structure in the course of creating and editing drawings. First, we employ techniques of perceptual organization by token grouping first to decompose ink-based strokes into primary units, and then reassemble these into coherent composite objects in the fashion of Marr's *Primal Sketch* [3]. To reflect the sometimes ambiguous and often goal-directed nature of human perception, our methods support multiple overlapping interpretations of the primitive image data. Second, we employ shape modeling, shape recognition, and curve tracing techniques in support of a straightforward gesture-based method for the user to select which visual object or objects he or she intends to edit.

The image editing paradigm follows conventional draw/select/modify user interactions. The pen is in one of two primary modes which are toggled by button press. In DRAW mode the pen simply lays down ink on the surface, which results the creation of stroke objects. In EDIT mode existing ink objects are deleted, moved, copied, rotated, and so forth, in a two-step process: (1) Select the object(s) to be edited; (2) Perform the actual deletion, copy, or transformation operation on the selected object(s).

Our system, called *PerSketch*, is currently implemented as a research prototype in CommonLisp running on Symbolics Lisp Machines and on Sun workstations under Lucid Lisp. The system is not optimized for speed, and currently presents a delay ranging from one quarter of a second to a few seconds to process each stroke as it is input, depending on drawing complexity in the vicinity of the newly added stroke. This is fast enough to support many graphical editing tasks but is too slow for unimpeded handwriting. The system has not been used by a large number of people. Its intent however is not for production use but as a vehicle to explore extensions to existing electronic whiteboard systems such as the Tivoli drawing program [8] which does have a substantial user community.

MAINTAINING PERCEPTUAL INTERPRETATIONS

Motivation

Out of the array of light and dark elements comprising an image, the human visual system constructs a richly articulated description across multiple spatial scales and multiple levels of abstraction. Our objective is to mimic these processes to some significant degree in data structures and procedures operating covertly to the user, but reflecting the salient spatial structures his visual system is likely to be constructing. These structures serve as a resource for evaluating users' later object selection commands, which often make reference to abstract entities in the image. The early, middle, and later stages of human visual processing each exploit a wealth of prior assumptions and knowledge about the visual world. These sources of constraint and their counterparts in our system are summarized in Table 1.

To date we have concentrated on image analysis support for sketch editing at the intermediate level of *perceptual organization*, that is, groupings of ink fragments that form coher-

Table 1: Knowledge used at different stages of the human visual system, and its counterparts in the PerSketch drawing editor.

Processing Stage	Human Visual Processing	PerSketch
Early Vision: Sensing	Assumptions about object cohesion and the laws of optics lead to hard-wired edge, line and motion sensitive analyzers.	Premise of line drawing editing reflected in a chaincode "ink" data structure for curve primitives.
Middle Vision: Perceptual Organization	Gestalt rules of perception guide the segmentation and articulation of coherent collections of image components likely to reflect common underlying objects or processes in the world.	Gestalt-like rules operate to assemble coherent groupings of "tokens" which represent individual strokes or collections of strokes.
Later Vision: Object Recognition, other tasks	Domain specific knowledge of particular visual environments supports tagging of task and goal-specific visual objects.	Open-ended sets of domain-specific rules construct semantically significant drawing entities such as specific shapes and drawn objects.

ent chunks as a result of curvilinear alignment, parallelism, cotermination, closure, or other straightforward, but spatially significant properties. Higher-level object recognition according with the semantics of particular drawing domains relies on knowledge of domain specific grouping rules, e.g. for schematic diagrams [2, 10], mechanical drawings [1], or chemical illustrations [7]. These recognition techniques fit into our architecture in principle but have not been incorporated as yet.

System Organization

The PerSketch system design is grounded in the fundamental representational elements of symbolic *tokens*, which make explicit the presence of and properties of visual structure in the image. Tokens possess attributes of:

- *type* of structure denoted
- spatial *location*
- *orientation*
- *scale* or size
- pointers to *supporting tokens* or data
- pointers to *supported tokens*
- additional type-specific properties such as curvature, aspect ratio, stroke width, and so forth, as applicable

The general principle of operation is that, as a sketch is created and modified, image analysis routines are constantly working behind the scenes to dynamically maintain an up-to-date multilevel description of visual structure present in the current image, represented in terms of tokens. A number of data structures and computational resources are employed to support this process; Figure 2 portrays the major components.

Object Lattice: In general, human users are capable of attending to any of several alternative interpretations or parsings of a given image depending upon their immediate tasks goals, surrounding context, and other cues. For example, in Figure 1 (Panel 3), one can readily focus on either the sausage or the rectangle. To which does the top stroke fragment belong? We maintain that the set of identifiably plausible intermediate level objects maintained by a perceptually supported image editor should reflect the rich and overlapping set of coherent perceptual chunks discovered or discoverable by the human visual system. To this end, tokens are conceived as forming an *Object Lattice* that relates perceptual objects across levels of abstraction. Figure 3 illustrates.

At the lowest levels in the hierarchy, tokens represent elemental curve fragments, which constitute PRIME objects. At the higher levels, each COMPOSITE object reflects a collection of PRIME or COMPOSITE objects that forms a sensible chunk according to some rule of perceptual organization. The lattice nature of this organization provides for alternative interpretations of the primary data, that is, a given PRIME object may participate in the support of more than one COMPOSITE object.

Token Grouping Procedures: COMPOSITE objects are placed in the Object Lattice one by one as coherent structure is identified by an open-ended and extensible set of token grouping procedures. Thus far we have found that substantial power derives from a rather modest set of rules underlying the grouping procedures, consisting mainly of analysis of cotermination relations and alignment relations among tokens representing curve fragments. Notice in Figure 3 how COMPOSITE objects emerge and are obliterated as the result of simple edit steps. We have also designed rules for identifying closure, parallelism, corners, and T-junctions; attempts at building rules for these and other structures can be found in the computer vision literature e.g. [4, 9].

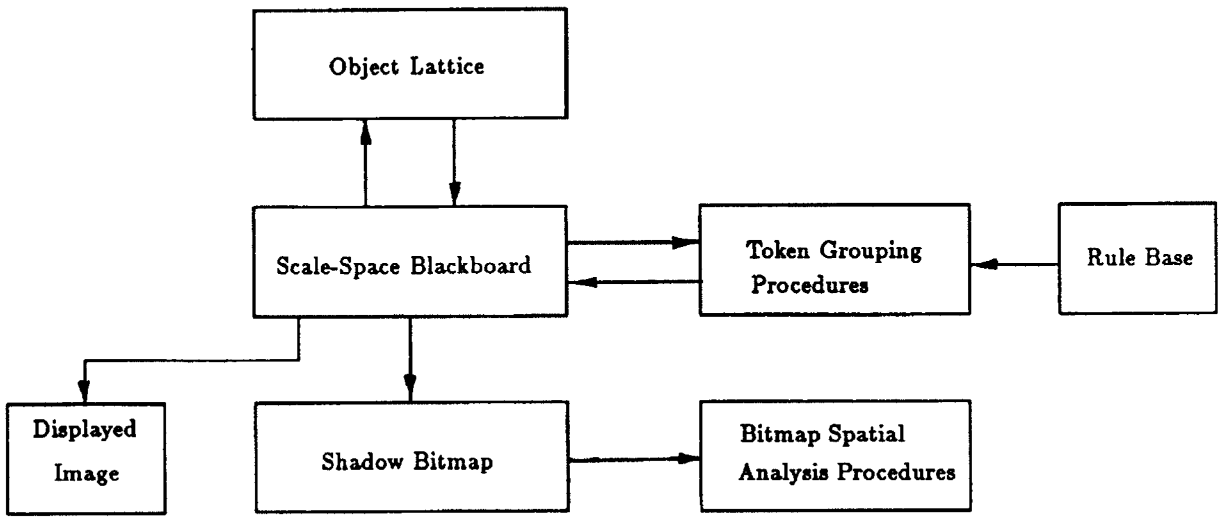


Figure 2: The major functional components of the PerSketch perceptually supported sketch editor.

Scale-Space Blackboard: Perceptually coherent objects are identified by virtue of qualifying spatial configurations of constituent tokens. The token grouping rules spend a great deal of effort searching for and testing for pairs and tuples of PRIME and COMPOSITE objects that satisfy respective conditions on their spatial arrangements. In general a combinatorial explosion could result from testing all combinations of tokens against all rules. However, by and large, meaningful collections of tokens will be specified as lying in a common spatial neighborhood, and the grouping rules may be applied locally. The combinatorics is managed by the use of a spatially indexed data structure called the *Scale-Space Blackboard*. This permits grouping procedures to perform enquiries of the form, "Return all tokens of type T within distance D of location (x, y)." The Scale-Space Blackboard also indexes tokens by size so that large scale structure is segregated from small scale detail. Spatial neighborhoods are defined not in terms of absolute pixels, but instead in terms of a *scale-normalized distance* which assesses spatial proximity with respect to the sizes of the objects involved. This ensures that like visual structure can be identified consistently across all magnifications of any given image. For details see [11].

Shadow Bitmap: For a sketch editing application, PRIME objects consist of the smallest curve fragments not broken by corners or junctions with other curves. Thus it routinely becomes necessary, during the course of a drawing session, for the system to break up an existing PRIME object when a new stroke is drawn to cross it. To support the efficient discovery of stroke intersections, a SHADOW BITMAP is maintained that depicts explicitly the paths of all strokes in the sketch. Whereas the image displayed to the user will show strokes in their proper thicknesses as well as ancillary user interface elements, the shadow bitmap maintains only single pixel wide "spines" of the curve elements. Whenever an intersection of a newly drawn stroke with an existing stroke is detected in the shadow bitmap, it becomes an easy matter to check nearby PRIME objects in the Scale-Space Blackboard to discover which token represents the existing stroke, so that it can be removed from the Blackboard and replaced with two smaller PRIME objects bounded by the newly formed junction. The "Bitmap Spatial Analysis Procedures" in Figure 2 support this and other similar functions related to analyzing the proximities of curves on the bitmap level.

Control Structure

In order to maintain a consistent internal representation of emergent perceptual structure during an image creation/editing session, the PerSketch line drawing editor obeys the control structure shown in Figure 4. The rounded boxes reflect the draw/select/modify loop apparent to the user.

The body of the image analysis work falls within the modules, "Remove Objects From Image" and "Add Objects to Image." Figure 3 illustrates the internal representations underlying the scenario of Figure 1. When objects are removed from the image, their constituent PRIME curve fragments are removed from the Scale-Space Blackboard and the Shadow Bitmap, and all COMPOSITE objects in the Blackboard that had been supported by any of these PRIME fragments are removed as well. Furthermore, PRIME objects remaining in the vicinity of newly deleted PRIME objects are tested to see if they can be merged. When objects are added to the image, the Shadow Bitmap is checked for the creation of new junctions. Existing PRIME objects are fragmented and replaced where necessary. Then, the token grouping rules are applied to label newly emergent COMPOSITE objects.

In the current implementation all perceptual organization rules are applied at each pass through the cycle. Computational expense increases with the sophistication and scope of the object recognition procedures, leading to a potential computational bottleneck as more domain-specific knowledge is brought to bear to recognize more abstract objects. However, the control structure easily extends to one in which processing resources are allocated among the primary func-

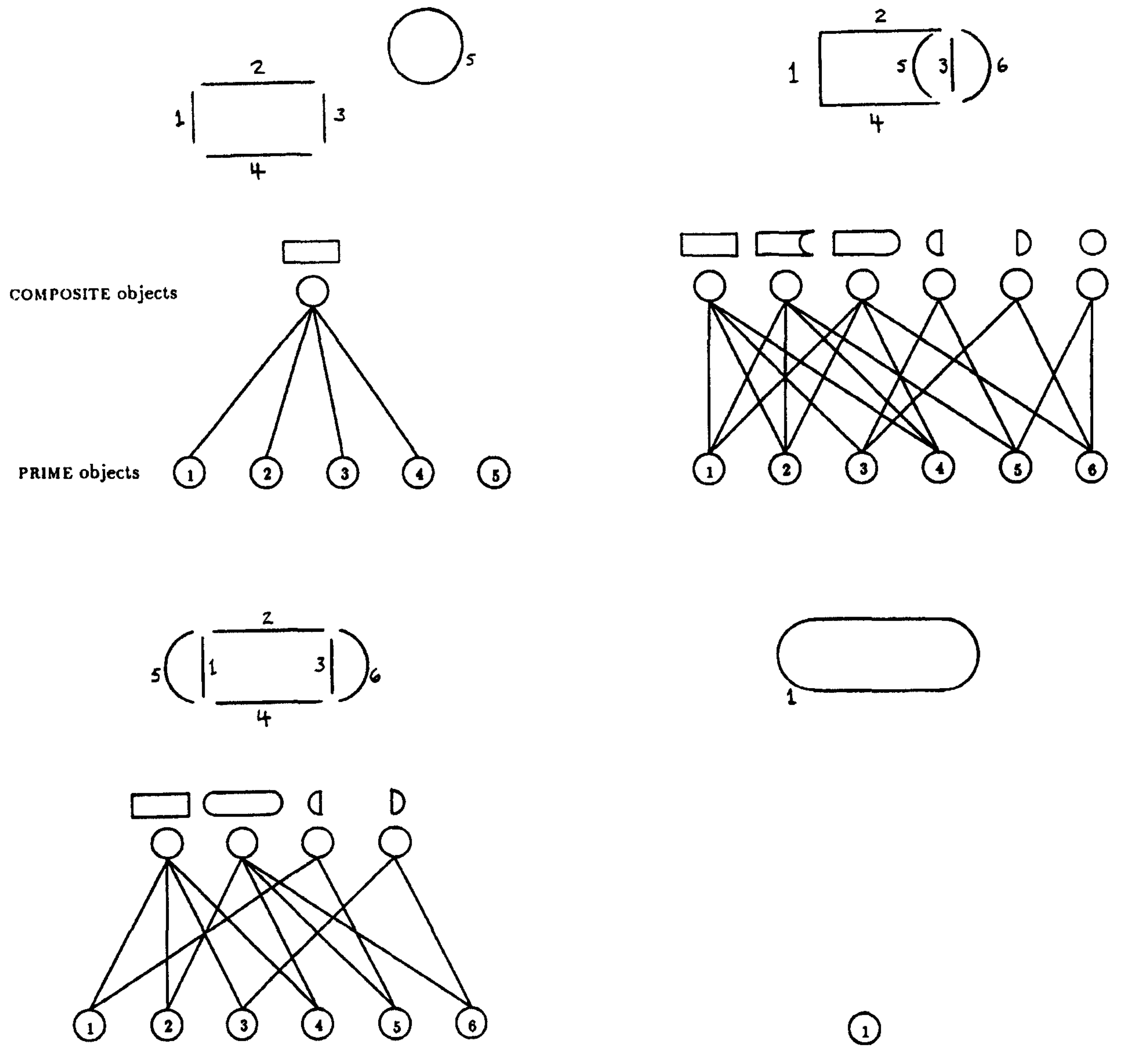


Figure 3: The Object Lattice of PRIME objects (elemental curve fragments) and COMPOSITE objects (emergent figures) underlying the sketch creation and editing steps of figure 1.

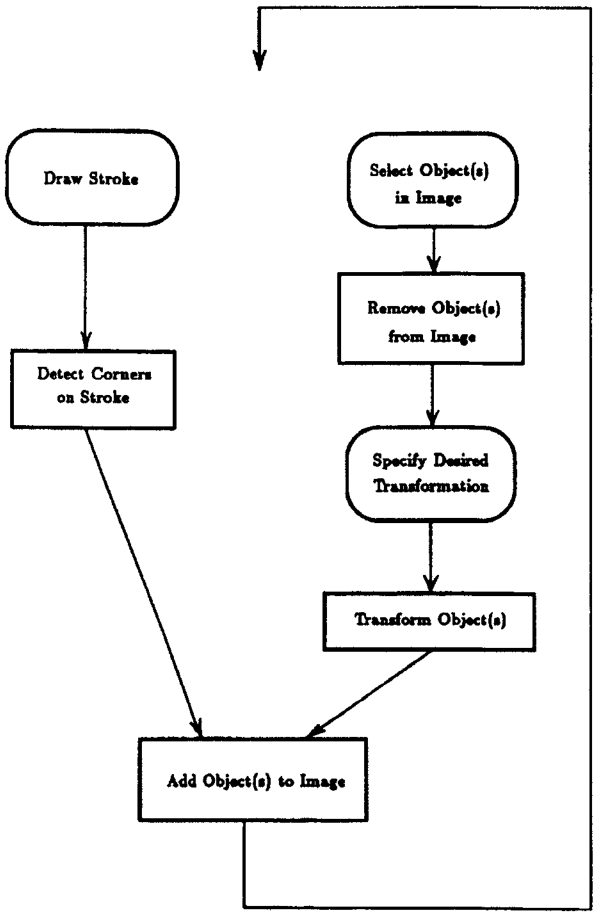

Draw Stroke
Detect Corners on Stroke
Select Object(s) in Image
Remove Object(s) from Image
Specify Desired Transformation
Transform Object(s)
Add Object(s) to Image

Figure 4: Program control structure underlying the Draw/Select/Modify interaction loop apparent to the user.

tion of supporting real-time user interaction, and a secondary function of identifying emergent spatial structure. In other words, as techniques for sophisticated drawing recognition are refined, they can be performed on an opportunistic basis in a real-time interactive environment.

GESTURE-BASED OBJECT SELECTION

Motivation

The collection of marks comprising an image may give rise to numerous overlapping plausible parsings and interpretations. PerSketch's image analysis procedures and internal representations attempt to make the most salient emergent structure explicitly available, but this raises the issue of how the user is to specify to the system which particular interpretation he has in mind at the moment. Gestures are a natural communication means for pen-based systems. The signal analysis problem that arises by adopting gesture-based selection is one of inferring the user's intent in terms of the collection of identified primitive and abstract objects. Machine vision techniques are useful because they provide mechanisms for generating hypotheses reflecting structured models for signal data, and for matching these hypotheses to observations.

Existing graphical object selection methods include pointing and clicking/tapping at or near image objects, and encircling. One problem with these techniques is that they lead to ambiguity when there are multiple overlapping interpretations of the visible marks. We wish to employ these techniques, but also to augment them to leverage the multiple levels of visual structure made explicit by token-based perceptual organization and domain specific object recognition procedures. We offer two additional gesture selection techniques, plus a framework for deploying a multiplicity of gesture selection methods simultaneously and in cooperation with one another.

Pose Matching

Although the various abstract objects identifiable in a collection of curvilinear lines may overlap and share support at a primitive level, each is characterized by its own unique combination of location and shape in the image. A technique called *pose matching* enables users to select among objects by exploiting the dual properties of gesture location and gesture shape. All the user has to do is to make a quick gesture that indicates the approximate location, orientation, and elongation of the intended object.

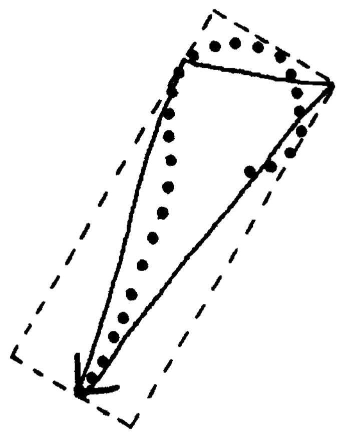
To each COMPOSITE object in the Scale-Space Blackboard we assign a parametric model based on its location and shape. At present we use a five degree of freedom *pose* model possessing the parameters, *x-location*, *y-location*, *orientation*, *length*, and *width*. These parameters are assigned equivalently to fitting an oriented bounding box to the object, using the first moment of inertia about the centroid to estimate orientation. See Figure 5a. Similarly, any curve comprising the path of a selection gesture can be modeled by pose parameters in the same way.

To compare object and gesture poses it is necessary to use a nonlinear similarity measure that trades off distance with congruence in shape. It is insufficient to use a linear similarity measure such as Euclidian distance because, for example, difference in orientation parameters of two poses is significant only when the aspect ratio of each is relatively high, but becomes insignificant when either object displays low aspect ratio. See Figure 5b. For any given selection gesture we rank order abstract objects residing in the Scale-Space Blackboard according to the similarity measure and offer the most similar as the best guess of the object the user intends to select. Figure 5c illustrates that pose matching permits perceptually coherent objects to be selected with a single gesture despite the presence of overlapping objects and clutter.

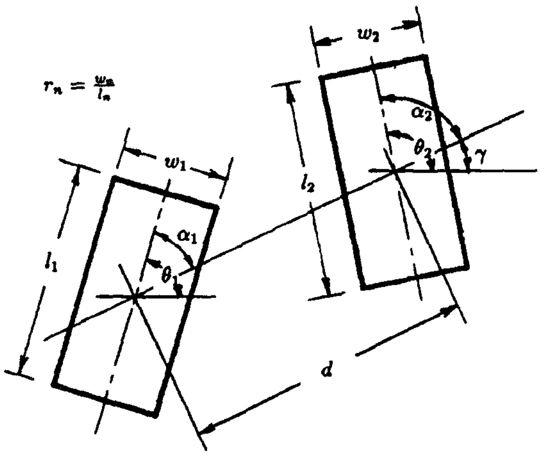
Path Tracing

A second method for gesture-based object selection allows the user to select an arbitrarily composed curve by tracing an approximate path over it. The algorithm identifies the path of curve fragments connected end-to-end that best matches the path of the selection gesture. For any given chain of curve fragments, a quality measure, or score, can be assigned assessing the degree to which a given selection gesture path resembles that defined by the curves. This is based on the fraction of the selection gesture spanned by the chain, and the fit of each constituent curve fragment to the portion of the gesture path it projects onto. See Figure 6a.

We use a dynamic programming algorithm to grow, one PRIME



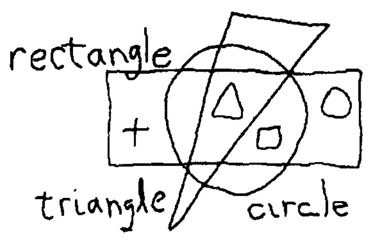
a



$$dissimilarity = 1 - (1 - f_1)(1 - f_2)(1 - f_r)(1 - f_\theta)(1 - f_s)$$

$$f_n = \frac{\rho_1 f_{n'}}{1 + f_{n'}}$$
$$f_{n'} = \frac{\rho_2 d / l_n}{1 - (1 - r_n)|\sin \alpha_n|}$$
$$f_r = |r_1 - r_2|$$
$$f_\theta = \frac{2}{\pi}|\theta_1 - \theta_2|(1 - \max(r_1, r_2))$$
$$f_s = \frac{|s_1 - s_2|}{1 + |s_1 - s_2|}$$

b



c

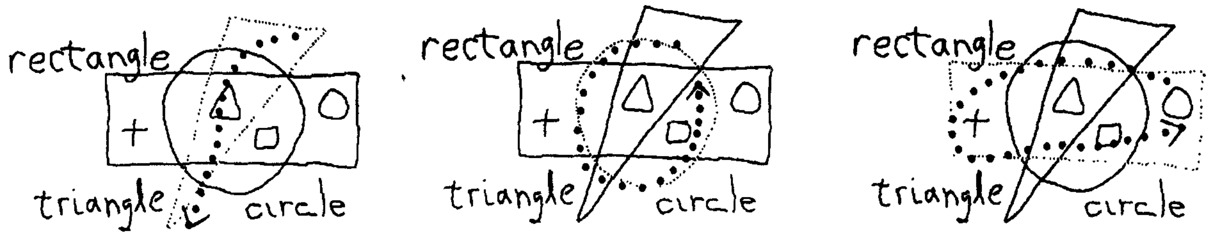
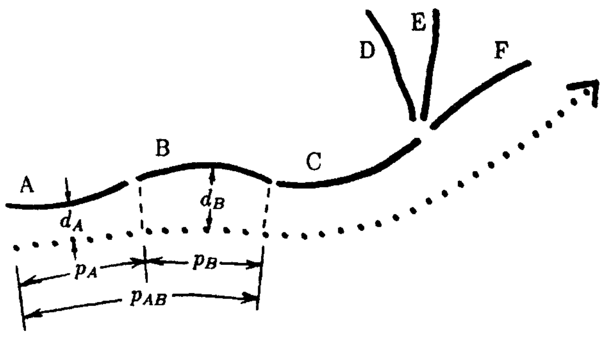


Figure 5: a. A pose model capturing the location and rough shape of emergent objects is equivalent to fitting an oriented bounding box around the object. The triangle and the selection gesture path (depicted with circular dots) share the same pose parameters. b. Nonlinear pose similarity measure (actually a dissimilarity measure whose minimum value of 0 occurs for identical poses). Dissimilarity is expressed as a soft OR function over differences in location, aspect ratio, orientation, and scale. Satisfactory values for the free parameters are $\rho_1 = .75$ and $\rho_2 = .4$. c. Examples of selection by pose matching. Selection gesture paths are depicted in circular dots, resulting selected objects are shown with dotted lines.



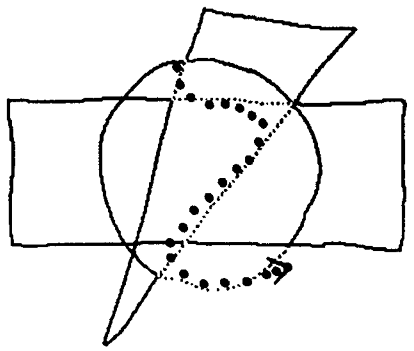


Figure 6: a. Any partial path consisting of a chain of PRIME curve fragments, e.g. path A-B, is assigned a score assessing how well it accounts for the selection gesture path (circular dots). The score is based on the fraction p of the selection gesture spanned, and on the maximum distance d between the selection path and each constituent curve fragment. b. Dotted lines show the path of end-to-end-linked PRIME curve fragments chosen by the path tracing algorithm for a sample selection gesture (circular dots).

curve fragment at a time, the best scoring chain of curve fragments that accounts for the selection gesture. At each step of the algorithm a list of partial-chains is maintained along with their associated gesture matching scores, each chain beginning with a PRIME object residing near the starting end of the selection gesture. For each partial chain we also note the *best-possible-score* that could be obtained if the chain were continued by a segment congruent with the remaining portion of the selection gesture. Each step of the path growing algorithm adds one link to the partial chain possessing the greatest *best-possible-score*. Partial chains whose *best-possible-scores* fall below the chain with the best actual score are pruned from further consideration. The Scale-Space Blackboard is used to efficiently find PRIME objects linked end-to-end with the endmost PRIME curve segment of each partial chain. Figure 6b presents the chain of PRIME curve fragments best matching a selection path in a complex scene.

Choosing Among Selection Methods

To give the user several methods for mapping a selection gesture to one or more objects in an image would lead to awkwardness if the user had to perform an extra step to specify among the selection methods available. Instead, we have implemented a simple means for the system to infer which selection method—point-and-tap, encircling, pose matching, or path tracing—is the most apt interpretation of the current selection gesture: Each time the user executes a selection gesture, each object selection algorithm is runs independently. Furthermore, each algorithm returns not only its best guess as to which object(s) the user intends to select, but also a confidence score indicating its "belief" that the user is indeed selecting by tapping, encircling, mimicing a pose, or tracing a path, respectively. For example, if the two ends of the selection gesture meet or form a small pigtail, then an encircling can be asserted with fairly high confidence. The confidence scores are compared to decide which selection method to choose. Parameters for the confidence score estimation algorithms are tuned by observing users and allowing for the kinds of slop they tend to exhibit in pointing, circling, pose matching, curve tracing, and so forth.

Example Use Scenarios

Figure 7 presents two scenarios for the kinds of situations in which perceptually supported sketch editing leads to faster and easier image modifications than are possible with conventional ink editing tools. Existing marks may be rearranged and reused in all or in part, and as they are combined with each other and with new strokes, the curvilinear units available for manipulation by the user reflect many of the perceptually salient structures in which his visual system conceives the image.

These examples happen to have been drawn originally not on a stylus-based computer, but on paper. In other words, we are able to import static bitmaps of line drawings and apply all of the image analysis procedures enabling perceptually-supported editing from scanned images as well as sketches created online.

CONCLUSION

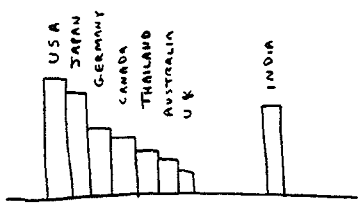
We view this as a first step into an emerging space of *WYPIWYG* (What You *Perceive* Is What You Get) image editors. Computer vision technology is at this moment in its infancy. We have in this paper applied only the simplest techniques of curvilinear token grouping. As the scientific study of perceptual organization and object recognition mature, more powerful image interpretation methods will become increasingly available to permit image editors to take advantage of additional kinds of image structure computed by the human visual system, including representations for image regions, region textures, three dimensional spatial structure, character and text recognition, and a host of methods for recognizing objects at a domain-specific, semantic level. By applying these forms of perception covertly as a user interacts with an image, we envision machines that come closer to giving one the image one wants by reading one's mind.

ACKNOWLEDGMENTS

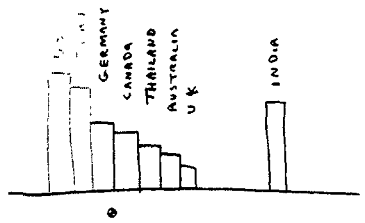
Craig Becker implemented an early version of the system and helped in developing the representations. We also thank the Tivoli group and the members of the PARC Image Understanding Area for helpful feedback and discussions.

REFERENCES

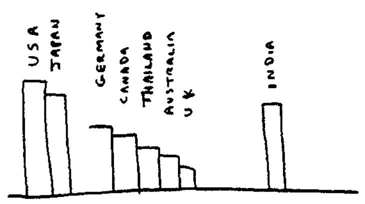
1. Joseph, S., and Pridmore, T. Knowledge-Directed Interpretation of Mechanical Engineering Drawings. *IEEE TPAMI*, **14**:9 (1992), 928-940.
2. Lee, S. Recognizing Hand-Drawn Electrical Circuit Symbols with Attributed Graph Matching. In H. S. Baird, H. Bunke, and K. Yamamoto (eds.), *Structured Document Image Analysis*. Springer-Verlag, New York, 1992.
3. Marr, D. Early Processing of Visual Information. *Phil. Trans. R. Soc. Lond.*, B **275** (1976), 483-519.
4. Mohan, R., and Nevatia, R. Using Perceptual Organization to Extract 3-D Structures. *IEEE TPAMI*, **11**:11 (1989), 1121-1139
5. Montalvo, F. Diagram Understanding: The Symbolic Descriptions Behind the Scenes. In T. Ichikawa, E. Jungert, and R. Korfhage (eds.), *Visual Languages and Applications*. Plenum Press, New York, 1990.
6. Moran, T. Deformalizing Computer and Communication Systems. Position Paper for the InterCHI93 Research Symposium. 1993.
7. Okazaki, S., and Tsuji, Y. An Adaptive Recognition Method for Line Drawings Using Construction Rules. *NEC Research and Development Journal*, **92** (1989).
8. Pedersen, E., McCall, K., Moran, T., and Halasz, F. Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. *Proceedings of the InterCHI93 Conference on Human Factors in Computer Systems*. ACM, New York, 1993.
9. Sarkar, S., and Boyer, K. Integration, Inference, and Management of Spatial Information Using Bayesian Networks: Perceptual Organization. *IEEE TPAMI*, **15**:3 (1993), 256-274.
10. Sato, T., and Tojo, A. Recognition and Understanding of Hand-Drawn Diagrams. *Proc. 6th International Conference on Pattern Recognition*. IEEE Computer Society Press, New Jersey, 1982.
11. Saund, E. Symbolic Construction of a 2-D Scale-Space Image. *IEEE TPAMI*, **12**:8 (1990), 817-830.
12. Saund, E. Identifying Salient Circular Arcs on Curves. *CVGIP: Image Understanding*, **58**:3 (1993), 327-337.



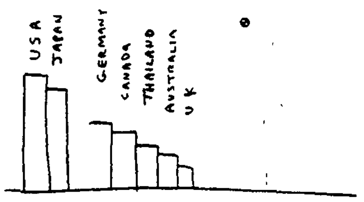
i. Original drawing.



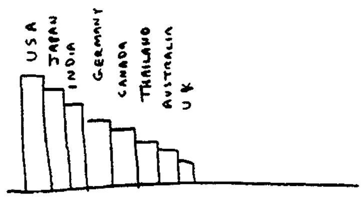
ii. User selects "USA" and "Japan" bars.



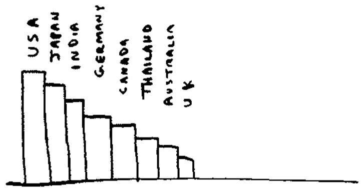
iii. "USA" and "Japan" bars moved over to make room for "India" bar.



iv. User selects "India" bar.

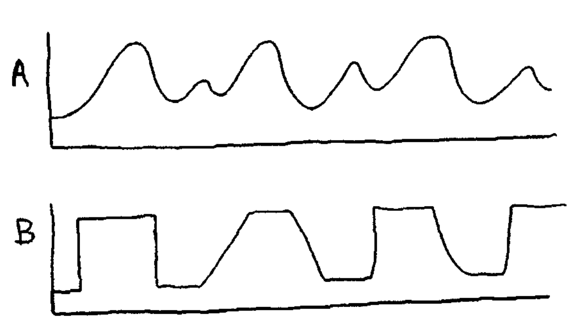


v. "India" bar moved to proper location in graph.

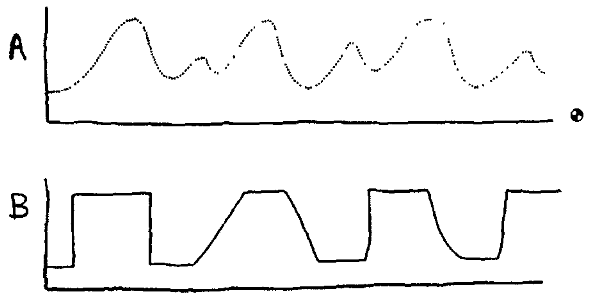


vi. Graph cleaned up by extending "Germany" bar.

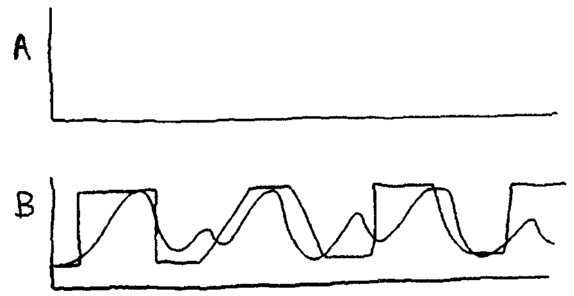
Figure 7: PerSketch use scenarios. In each case the main intermediate steps are shown. a. Inserting a missing item in a bar chart.



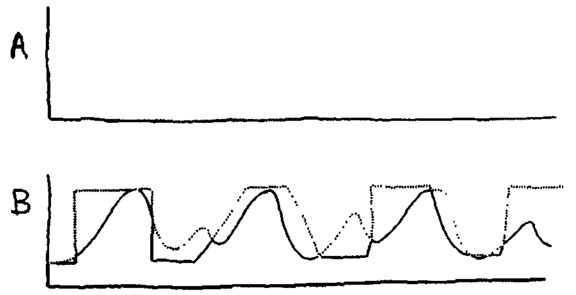
i. Original drawing.



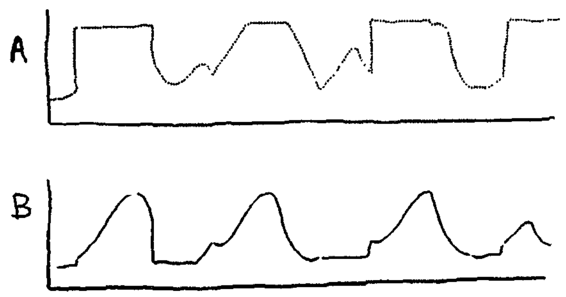
ii. User selects "analog" signal.



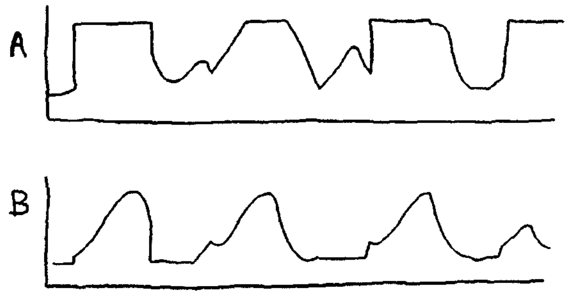
iii. Analog signal moved on top of "digital" signal.



iv. User selects topmost curve elements of composite signal, leaving "gated" signal unselected.



v. Topmost curve elements moved away, leaving behind "gated" signal.



vi. Drawing cleaned up by filling in gaps.

Figure 7: b. Combining sketches of two electrical signals to observe the result when the analog signal (A) is gated by the digital signal (B).