

Signal-based figure/ground separation

James V. Mahoney

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

Abstract

This paper shows that by adopting strictly signal-based methods, wide-ranging figure/ground separation processes may be reduced to a small, fixed set of primitive operations, in a concise, parsimonious, and efficient way. In the signal-based approach, the primitives consume and produce pointwise descriptions of the scene, rather than compact descriptions. The expressive power of the approach derives from powerful primitives for integrating information across space, from the decomposition of geometric problems into pointwise subproblems, from universal adherence to a common data structure, and from the exploitation and preservation of locality.

1 Introduction

This paper shows that by adopting strictly signal-based methods, wide-ranging figure/ground separation processes may be reduced to a small, fixed set of primitive operations, in a concise, parsimonious, and efficient way. In the signal-based approach, the primitives consume and produce pointwise descriptions of the scene, rather than compact descriptions.

Voronoi tessellation provides a paradigmatic example of a signal-based method. The Voronoi region of a black connected component f in a binary image is the set of pixels that are closer to a pixel of f than to any black pixel not in f . The Voronoi tessellation is a partition of a binary image into Voronoi regions. It may be computed in two steps: first, each black pixel is assigned a label uniquely associated with its connected component; then each white pixel is assigned the label of the nearest black pixel. Suppose that each of these steps is a primitive operation. Four properties of this routine are characteristic of our methods:

Powerful primitives: The routine is expressed using primitives that establish visually important spatial relations, such as connectivity and proximity.

Minute subdivision: The given problem is broken down into the smallest possible subproblems—spatial relations among individual points.

Uniform data structure: The primitives produce data

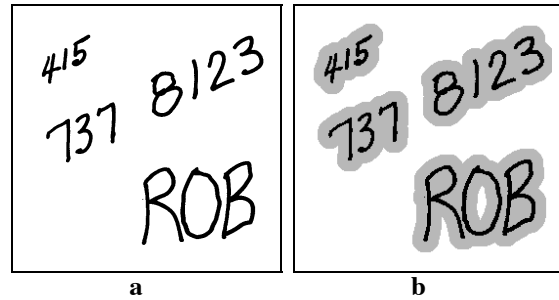


Figure 1: (a) This scene is structured by proximity and similarity relations. (b) The grey blobs define proximity groups by inclusion.

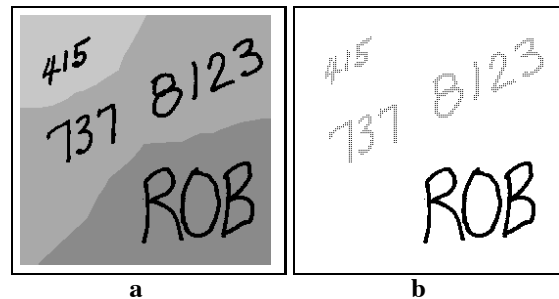


Figure 2: (a) Height-similarity groups of letters; for clarity, Voronoi regions defined by the group identifiers are shown by grey level. (b) Letters salient in height, shown in black.

structures of the same kind as they consume; therefore so does the routine.

Preservation of locality: All input and result arrays are the same size and in spatial registration.

These properties bear directly on our expressiveness goals. **Minute subdivision contributes to conciseness.** In geometric intersection by ANDing binary arrays, minute subdivision by itself gives the tremendous simplification relative to analytic intersection algorithms. The case of Voronoi tessellation illustrates that the scope of array based methods is extended by primitives that extract more global pixel relations.

Uniform data structure contributes to parsimony: any operation may be applied to any result, so routines may be combined in flexible ways, leading to extensive sharing and

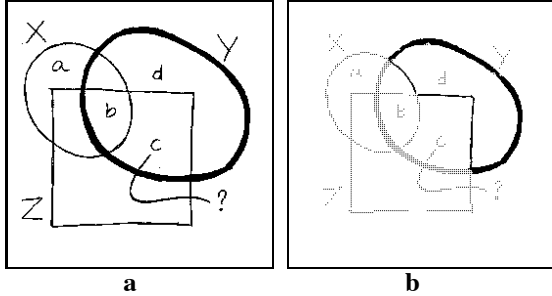


Figure 3: (a) A scene requiring separation of curvilinear figures. (b) A region boundary extracted using adjacency.

reuse. Proliferation of data structures leads to proliferation of primitive operations on them, and to costly costly coercions between them.

Preservation of locality leads to efficiency. The time-complexity of many analytic geometric algorithms—e.g., for intersection and Voronoi tessellation—is dominated by the cost of sorting [7]. This is because the input is a *list* of geometric objects, whereas the result depends on the relations among spatially neighboring objects. Sorting in the spatial dimensions is required to establish these neighbor relationships. Corresponding signal-based methods work without sorting, because proximity in the scene is preserved in the input image and throughout the computation. Uniform data structure, too, contributes to efficiency: when sorting is required—to establish object relations in non-spatial dimensions—we can use linear time (bucket) sort, because of the fixed-size and discreteness of the arrays.

The goal of this paper is to make the case for a signal-based approach to figure/ground separation. This is done by proposing a set of primitive operations (in Section 3) and then showing how a wide range of important figure/ground capabilities reduce to them (in Section 4). The capabilities we consider (in Section 2) in making this case are drawn from the restricted domain of graphic scenes. Nevertheless, they are quite diverse, so reduction to our primitives demonstrates significant scope and makes our case. Section 5 discusses related work and Section 6 makes concluding remarks.

2 Figure/ground capabilities

The following figure/ground separation capabilities seem to play a fundamental role in graphic scene analysis: (i) *grouping*: forming equivalence classes based on similarity or proximity; (ii) *selection*: extracting figures based on prominence; (iii) *proximity shifts*: extracting figures based on proximity relations to a given figure or location; (iv) *curve tracing*: separating intersecting curvilinear figures, often based on colinearity.

Grouping. Figure 1 (a) is highly structured by proximity and similarity relations. These relations enable the visual

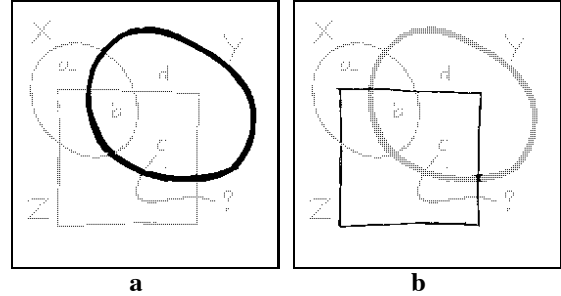


Figure 4: Curves with a salient local property extracted by selection (shown in black): (a) salient width; (b) salient curvature.

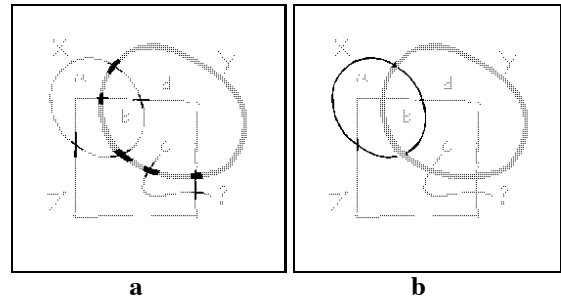


Figure 5: (a) Curve segmentation into junctions (shown in black) and segments (shown in grey). (b) A curve extracted by colinearity tracing through junctions.

system to form *groups*, i.e., equivalence classes of simple scene elements. Figure 1 (b) shows groups defined by proximity, alone, among the letters. Figure 2 (a) shows groups defined by similarity in height.

Selection. Figure 1 (a) also illustrates that property differences among scene objects can give rise to a sense of prominence or salience. Figure 2 (b) shows some letters that stand out due to height, relative to the other letters. Figure extraction based on a salience measure is called *selection*.

Proximity shifts. Proximity relations also provide visual frames of reference, i.e., ways of “pointing to” particular fig-

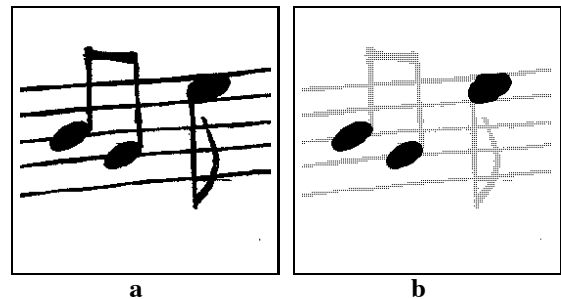


Figure 6: (a) The notes are superimposed on lines, but are easily separated visually. (b) Note heads extracted by a form of proximity grouping.

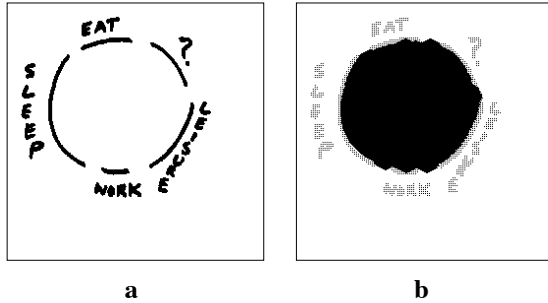


Figure 7: (a) We often perceive regions whose “boundaries” are not closed or connected. (b) The bounded region was extracted by a form of proximity grouping.

ures. In Figure 1 (a) we interpret “the string at the upper-left” to mean the string ‘415’. This interpretation relies on proximity of a given group, of letters, to a fixed reference location, a corner of the image. Similarly, interpreting “the letter nearest the ‘R’” relies on proximity to an already extracted figure (the ‘R’), while “the number to the left of the ‘8’” relies on proximity constrained to a range of directions.

Curve tracing. In the Venn diagram of Figure 3 (a), the set boundaries and other curves intersect to form a complex curvilinear figure. They must be separated from one another to interpret this scene. Some of them are separable by selection: the boundary of set “Y” is salient in width, and the boundary of set “Z” is salient in curvature (Figure 4 a, b).

This scene involves other separation problems, however, that require new methods. Extracting the boundary of the set labeled ‘X’ requires *curve tracing*, i.e., separating a curve by following it sequentially through its junctions with other curves (Figure 5). A prerequisite for this is the capacity to decompose a curvilinear figure into junctions, terminations, and simple curve segments (Figure 5 (a)). Tracing also requires a criterion for choosing an outgoing branch at each junction. Any set boundary in Figure 3 (a) may be extracted by a tracing process that chooses the branch colinear to the current segment at every crossing. When the goal is to find or follow a path, as in map reading, different choices are made at each crossing.

Consider, also, the problem of extracting the boundary of a given region, e.g., the region marked ‘d’. The boundary is not given by any intrinsic property, but by proximity. (See Figure 3 (b).)

Two further separation phenomena involve curves though not necessarily curve tracing. We often perceive compact shapes superimposed on curves as distinct figures (Figure 6). Conversely, we often perceive distinct regions whose “boundaries” are not closed or connected (Figure 7). (The bounding figures may not even be curvilinear.) We refer to such figures as *partially delineated* and *partial bounded*, respectively.

3 Primitive operations

The primitives are of two basic kinds. *Pointwise primitives* consist of common arithmetic and logical operations applied elementwise to array arguments. For example, the **add** operation adds corresponding array elements, resulting in an array of sums. *Integration primitives* combine values across sets of locations, using scalar $+$, **max**, **min**, \vee , or \wedge .

The inputs and results of the pointwise primitives are either all 2D or all 1D. *2D integration primitives* consume and produce 2D arrays, operating in the spatial dimensions. *1D integration primitives* consume and produce 1D arrays, operating in the figure property dimensions. These primitives are used along with scalar computations, e.g., operations involving global properties of a figure or the entire scene. The *bridging* integration primitives connect the 2D, 1D, and scalar realms.

2D integration primitives —

translate uniformly translates the elements of its input array by given x, y offsets.

labelcc associates a unique integer identifier with each black connected component in a binary input array, and assigns to each black pixel the identifier of its component.

project accumulates values along pixel rows or columns. E.g., downward projection assigns to location p a combination of p ’s value in the input array and the result of downward projection at the location immediately above p .

read assigns to each location the value of the nearest non-zero location in the input.

spread takes a data array and an address array as arguments. An *equivalence class* of locations is a set of locations having identical values in the address array. Within each equivalence class, **spread** combines data values into a single scalar and then assigns each scalar result to every location in the equivalence class from which it was computed.

1D integration primitives —

translate-1d and **project-1d** are 1D counterparts of **translate** and **project**.

Bridging integration primitives —

accumulate is a generalization of histogramming. A value in the (2D) *address* array argument specifies the output bucket in the 1D array result into which a value at the corresponding location of the (2D) *data* array argument should be combined. (I.e., all the data values in an equivalence class are combined into a single scalar.)

distribute: a value at a particular location p in the (2D) address array argument specifies a bucket in the (1D) data array argument. The value in that bucket is assigned to the corresponding location (p) in the (2D) result array. (I.e., this operation copies a scalar value associated with an equivalence class to each location in the class.)

(Note that **spread** is the composition of **accumulate** and

distribute.)

global accumulate combines all values in the (1D or 2D) input array into a single scalar.

The linear time imperative. These primitives meet the requirement that *run time and space must be linear in the number of array locations*, on a serial machine. This restricts integration primitives to processes that move a small, fixed neighborhood across the arrays in a fixed number of passes, where a pass may visit each location no more than k times, for some small, fixed k . E.g., **read** is a simple extension of a standard distance transform ([4]) that raster scans the array, in two passes, always examining a five-pixel neighborhood.

4 Figure/ground routines

All the figure/ground separation examples shown in this paper are actual results of applying routines defined along the lines below. In the following sketches, only the integration operations involved in a routine are mentioned explicitly.

Proximity shifts. All proximity shift routines apply the fact that f is the nearest connected component to a given location p iff f 's Voronoi region includes p .

Fixed reference shifts: The reference “the lower leftmost figure” is taken to mean “the figure nearest the bottom left corner pixel of the image.” A corner pixel is represented by an array constant. The required figure, f , is extracted in three steps: (i) all Voronoi regions are labeled uniquely; (ii) the Voronoi region, r , of f is extracted by a kind of seed fill, implemented using **spread**; (iii) r is intersected with the input image to give f .

Neighbor shifts: Let r_a, r_b be the Voronoi regions of figures A, B, respectively, in Figure 8. Consider the set of pixels q_1 in r_a that are adjacent to a pixel of r_b . (Such a set is termed a *Voronoi border* of r_a .) Let m_1 be the pixel in q_1 that is closest to A. Let s_1 be a pixel in r_b adjacent to m_1 . s_1 may serve as a seed for extracting the neighbor B.

m_1 may be extracted in four main steps: (i) the distance transform is computed, using **read** (applied to pixel x, y coordinates made by **project**); (ii) all Voronoi regions are labeled uniquely; (iii) all Voronoi borders are labeled uniquely, using **translate**; (iv) the relevant m_i in each Voronoi border of r_a are marked based on the desired proximity relation. (I.e., in step (iv), extracting the nearest neighbor and extracting the neighbor in a given direction range involve different constraints.) Given the marked m_i , the adjacent s_1 are marked using **translate**.

Curve segmentation. Junctions, terminations, and curve segments are separated by labeling every pixel p with a measure of local branching, and then thresholding the result. This measure is based on the *exit count* of each square region r of width w that contains p . The exit count of r is the number of times that black components intersect the border

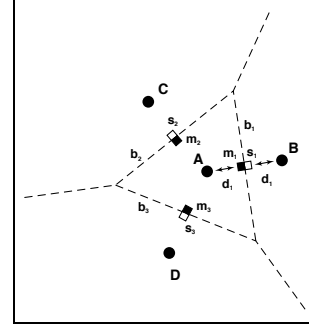


Figure 8: Proximity shifts are achieved using raster Voronoi representation and seed fill operations.

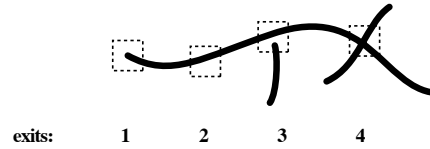


Figure 9: Line features are associated with a measure of local topology, the *exit count*.

of r . If r has suitable size and position, the exit count reflects the local branching of a curvilinear figure that r partially includes. E.g., the exit count is four if r wholly contains a crossing of two lines.

To allow varying curve width, given constant w , the curves are first thinned. The positioning issue is addressed using a voting scheme—heuristically, meaningful positionings are more common than spurious ones. To apply this idea, exit count is computed at all positions. The branching factor at a pixel p is taken to be the modal value of exit count among regions that include p . Exit and vote counts are computed “concurrently” at every location using a few **project** and **translate** operations.

Tracing. Whereas a step of a conventional curve tracing process operates on a small pixel neighborhood, signal-based tracing operates on an entire curve segment at a step. Figure 10 illustrates the situation at a junction. The segment

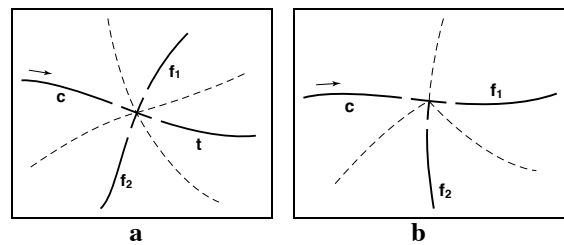


Figure 10: Signal-based tracing: (a) at a crossing, the colinear segment usually corresponds to the non-flanking region; (b) at a fork, both outgoing segments are in flanking regions.

last extracted, c , terminates in a newly encountered junction j . In Figure 10 (a), j is a crossing with three emanating segments. Two of them, f_1 and f_2 , are angular neighbors of c , while the other, t , is not. Tasks like “follow the left branch” or “follow the colinear branch,” involve establishing which emanating segment bears the specified relationship to c . To support this, segments are initially labeled with local orientation.

Signal-based tracing is simplest if it operates on the Voronoi regions of the input segments, rather than the segments themselves, because angular neighbor relations among segments correspond to pixel adjacency relations among segment Voronoi regions. The routine involves **spread** and **translate** operations.

Jordan boundary separation. The boundary of a given white region, r , may be extracted by applying a proximity relation, rather than tracing (Figure 3). A black pixel is in the boundary of r if it is within about a line-width away from both a pixel of r and any white pixel not in r . To allow for unknown or spatially varying line width, this idea is applied to the skeleton of the image, and the definition is recast in terms of pixel adjacency, rather than a distance threshold. The routine involves only **translate**.

Extracting skeletons. The *neighbor direction* at a white pixel is the direction to the nearest black pixel, computed using **read**. A *ridge pixel* is a pixel whose neighbor direction differs from that of some four-adjacent pixel by more than a given threshold. We extract *ridges* using a form of hysteresis. Let r_l, r_h be the binary arrays of ridge pixels defined at the low and high thresholds, respectively. Ridges are black components of r_l that include black components of r_h , extracted using **spread**. The *skeleton* of a figure consists of the ridges in its complement.

Figure property labeling. In Figure 1 (a), selection and grouping involved a measure of height for each letter. Such global figure properties are typically computed “concurrently” using **spread**. Simple properties computed in this way include area, perimeter, bounding box width and height, diameter (number of skeleton points), and girth (modal distance-to-boundary at skeleton points).

Selection. Selection makes explicit use of the global distribution of the values of a given image property P . Assume P is given as a 2D array \mathbf{P} . The values of P are sorted into increasing order in a 1D array by applying **accumulate**. The differences, or “gaps”, between consecutive values in this ordering are computed using **translate-1d** and **project-1d**. A *saliency threshold* is any value of P falling within the widest gap (or a “wide enough” gap). t is extracted using the **global accumulate** operation. Thresholding \mathbf{P} by t extracts the salient figures as a binary array.

Similarity grouping. Given a property P of some initial units—e.g., individual pixels, connected components, etc.—a *similarity grouping* is a partitioning of the elements into equivalence classes, and a *group* is one of these classes. Grouping involves two stages. First, a similarity threshold τ on differences in P is computed. Second, equivalence classes of elements are defined by applying the threshold—two objects are in the same group if their values of P differ by less than τ .

Consider the sequence, G , of all groupings defined as τ ranges from zero to infinity. *Stability* is a general heuristic criterion for choosing τ , as a function of the persistence of each distinct grouping in G . We pick τ to give a grouping that “lasts long enough” in this evolution; e.g., the most stable non-trivial grouping. The *stability measure* of a grouping is the length of the interval of τ over which it persists.

The τ associated with a desired degree of stability may be calculated directly from the distribution of P , without computing any groupings. Since τ is a threshold on the *gaps* between consecutive values of P , the groupings only change when τ exactly matches a gap. The stability measures of all groupings are given by (i) computing and sorting the gaps and (ii) taking the differences between consecutive gaps. Extracting these values, and then τ , involves the same primitives we used to extract t .

Establishing equivalence classes of P , given τ , involves a transitive closure operation on a 1D array, implemented using **accumulate**, **translate-1d**, **project-1d**. When P is given as a 2D array, the final step is to label 2D locations with their equivalence class identifiers, using **distribute**.

Proximity grouping. There is a direct analogy between similarity and proximity grouping: difference in P (distance in the property dimension) corresponds to spatial distance. Using the stability scheme, τ is computed from the distribution of *spatial gap sizes* between neighboring components. (A gap size is the minimal distance value along a ridge, computed using **spread**.) Then, spatial equivalence classes are formed by thresholding the distance transform *below* τ . Black connected components in the result define groups of input elements by inclusion (Figure 1).

Partially-bounded regions are extracted in the same way, except that the distance transform is thresholded *above* τ (Figure 7). Black connected components in the result are then “grown” by a radius of τ , by distance thresholding, to give the desired regions. With foreground and background reversed, this very scheme extracts partially delineated regions (Figure 6).

5 Related work

The reduction of human perceptual capabilities to a small, fixed set of basic operations is the heart of Ullman’s *visual routines* theory [10], from which our work descends. The basic operations Ullman proposed include *coloring*, *trac-*

ing, shifts, indexing, and marking, which relate more closely to the routines of Section 2 than to our primitives. For example, Ullman’s indexing operation—a shift to an odd-man-out location—corresponds to a combination of selection (applied to local differences) and proximity shift.

Mathematical morphology ([8], [5]) is a signal-based framework—it employs 2D arrays uniformly and it preserves locality—but one limited in integration primitives to **translate** alone. I.e., Morphology’s *dilate* and *erode* operations may be recast as sequential uniform translations. Extracting global relations therefore involves extensive iteration, and this can be prohibitively inefficient and/or awkward. In practise, morphology is often combined with global operations such as connected component labeling, histogramming, multi-resolution processing [1], etc.

Signal-based image analysis is a form of data parallel computation, and some of our primitives are found in data parallel languages—e.g., languages for the Connection Machine [9]—which are often applied to 2D image analysis problems on mesh and hypercube computers. They provide **translate**, **project**, and the pointwise primitives. (**project** is the parallel prefix or ‘scan’ operation.) Their general global *send* and *get* operations subsume **accumulate** and **distribute**. Not being specialized to image analysis, they do not include operations related to **labelcc** or **read**.

Cass [3] introduced a *generalized histogram* operation that corresponds to the **accumulate** operation. He also composed it with a subsequent distribution process. This was in the context of a data parallel model-based recognition system, not an image analysis “language.”

Seymour [6] is a general purpose data parallel language that incorporates additional global operations including parallel prefix, sort, broadcast/report, associative read/write, merge, and reduction. Though not specialized to image analysis, Seymour is closely related to our approach in that it extends standard data parallelism with higher-level “integration” operations. Most global operations in Seymour amount to compositions of **accumulate** and **distribute**; e.g., its “semi-group operation” is equivalent to **spread**. It does not provide operations related to **labelcc** and **read**.

Many image processing/analysis languages have been proposed, but two recent ones ([11], [2]) stand out in providing a unified framework for implementing global operations such as histogramming, connected component labeling, distance transforms, etc., with parallel hardware in mind. Our “language” takes such operations as primitive, and supports processes at a higher level.

6 Conclusion

We proposed a signal-based approach to figure/ground separation, to achieve conciseness, parsimony, and efficiency. We argued that signal-based methods meet these goals through minute problem decomposition, uniform data struc-

ture, and preservation of locality, given powerful primitives for integrating information across space. We examined diverse figure/ground separation capabilities drawn from the domain of graphic scenes. We introduced a set of primitive operations, and sketched how the figure/ground capabilities reduce to them.

The primitives and routines have been widely applied to many diagram understanding and page segmentation tasks, but a discussion of these applications is beyond the scope of the paper.

Most of the routines run in $O(N)$ time on a serial machine, N being the number of pixels. Indeed, N is typically much larger than the number of geometric objects involved in corresponding analytic methods, but this is balanced somewhat by the raster-scan uniformity of the primitives, which fully exploits serial memory bandwidth. The signal-based approach becomes competitive given a range of further optimizations that are beyond the scope of this paper.

The demonstrations were limited to graphic images, where our practical experience lies, but the primitives are quite general. We are beginning to extend the figure/ground routines to other kinds of scenes.

References

- [1] D. Bloomberg. Multiresolution morphological approach to document image analysis. *1st ICDAR*, Saint-Malo, 1991: 963–971.
- [2] J. Brown and D. Crookes. A high level language for parallel image processing. *Image and Vision Computing*, 12, 1994:67–79.
- [3] T. Cass. Robust 2D model-based object recognition. A. I. TR 1132, 1988. Cambridge, MA: M.I.T. Artificial Intelligence Laboratory.
- [4] P.E. Danielsson. Euclidean distance mapping. *CGIP Processing*, 14, 1980:227–248.
- [5] R. Haralick, S. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE PAMI*, 9, 1987 :532–550.
- [6] R. Miller and Q. Stout. Seymour: a portable parallel programming language. *Structured Programming* 11, 1990:157–171.
- [7] F. Preparata and M. Shamos. *Computational geometry: an introduction*. New York: Springer-Verlag, 1985.
- [8] J. P. Serra. *Image analysis and mathematical morphology*. New York: Academic Press, 1982.
- [9] Thinking Machines Corporation. C* reference Manual. Version 5.0 Cambridge, MA, 1988.
- [10] S. Ullman. Visual Routines. *Cognition* 18, 1984: 97–159.
- [11] J. A. Webb. Steps toward architecture-independent image processing. *Computer*, 25, 2, 1992:21–31.