Computer Vision and Image Understanding
Vol. 76, No. 1, October, pp. 19–35, 1999
Article ID cviu.1999.0790, available online at http://www.idealibrary.com on IDEAL®

Beyond the Uniqueness Assumption: Ambiguity Representation and Redundancy Elimination in the Computation of a Covering Sample of Salient Contour Cycles*

Stefano Casadei and Sanjoy Mitter

Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
E-mail: casadei@lids.mit.edu

Received October 1, 1998; accepted June 14, 1999

Perceptual organization provides an intermediate representation of data by means of object- and goal-independent information. The lack of complete information makes perceptual organization an intrinsically ambiguous process which invalidates the uniqueness assumption and requires instead the generation of multiple solutions. This raises the issue of eliminating redundancies which, in a recursive algorithm, might otherwise cause combinatorial explosion of the search space. These aspects of perceptual organization are illustrated in the context of cycle detection in a contour graph. A provably correct algorithm for this problem is proposed.

1. INTRODUCTION

An important component of many vision systems consists of organizing image descriptors into collections which are likely to originate from the same source in the scene (e.g., from the same object). To a certain degree, the computation of these collections can be accomplished by means of *generic* (i.e., object-independent) information so that this process can be used as an intermediate step toward object recognition [21]. Several decades ago Kanizsa [19] observed that generic properties of collections of contours, such as proximity, continuity, similarity, closure, and symmetry, are used by the human visual system to organize contour fragments into collections which give rise to the perception of objects. These properties have also been studied by many computer vision researchers and are used in several computer vision algorithms. The grouping of image descriptors according to the above properties is often referred to as *perceptual organization* and is based on the statistical assumption that these properties are more likely to occur among fragments arising from the same source (nonaccidentalness principle) [38, 21]. A good review and classification scheme for perceptual organization work in computer vision can be found in [29].

Providing a mathematical formulation for the problem of perceptual organization has proven to be a very elusive task. A reason for this difficulty is that the information used by perceptual organization is not sufficient to define uniquely what the image representation should be. In fact, by restricting the available information to being generic, i.e., independent of what objects are in the scene, there remains a substantial intrinsic ambiguity in the interpretation of the data which can not be resolved without artificial assumptions. Thus the goal of perceptual organization should be to enumerate all interpretations which are consistent with the data and the available information, leaving the task of eliminating intrinsic ambiguities to higher levels of processing.

Instead, many approaches to perceptual organization are based on *optimization* of a suitable cost functional and therefore assume that a unique solution exists and is obtainable by means of the available generic information. For instance, some graph-based formulations of contour grouping seek an optimal path between every pair of points [12], thus assuming that there exists at most one contour between any two points *and* that this unique contour can be recovered unambiguously by means of the generic information available at the perceptual organization level. Figure 8 in Section 5 shows an example of an image in which multiple contours exist between two points.

An alternative to optimization which overcomes the uniqueness assumption consists of exhaustive enumeration of all interpretations (e.g., contour groupings) which satisfy a certain criterion. For example, in the specific problem of graph-based contour grouping, one can seek all the paths between any pair of points which have nonnegligible probability of being near a true contour. In the context of perceptual organization, one restricts this probability to depend only on object-independent features such as proximity, colinearity, and similarity.

A major difficulty in this approach is that the cardinality of the sets of entities satisfying the criterion can become exponentially large as their spatial extent grows. In our contour grouping example, it is possible to argue that, due to the presence of multiple responses to the same object, the average number of “good” paths between two points depends exponentially on the number of vertices on the path. Roughly speaking, this is because the number of paths between p_1 and p_2 is given by $N(p_1, p_2) = N(p_1, p_3)N(p_3, p_2)$ if p_3 is a point on the path

* This research is supported by MURI grant DAAH04-96-1-0445, Foundations of Performance Metrics for Object Recognition.

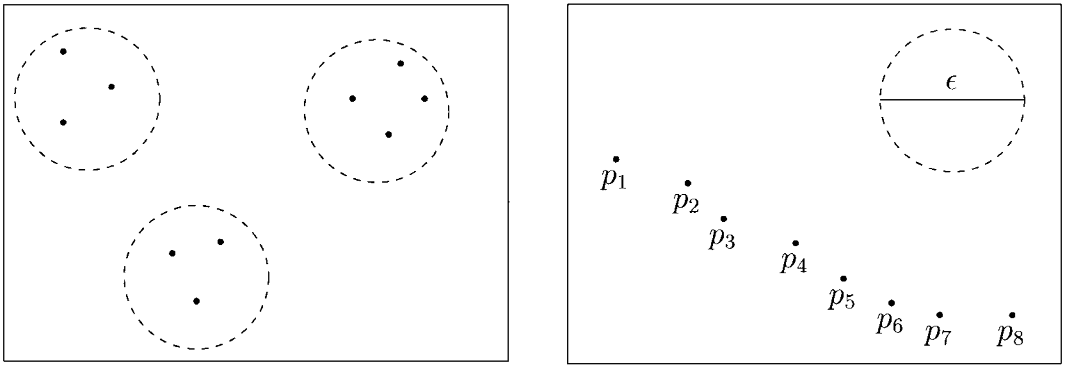


FIG. 1. The clustering condition in the $\mathbb{R}^2$ domain (for ϵ given by the diameter of the dashed circles). (left) A set of ten points satisfying the condition. Formally, for any three points p_1, p_2, p_3 in the set, if $d(p_1, p_2) \le \epsilon$ and $d(p_2, p_3) \le \epsilon$ then $d(p_1, p_3) \le \epsilon$. (right) a set which does not satisfy the condition. The clustering condition allows one to prune a finite set S by using only local tests. Consider for instance the following algorithm. Order the set S arbitrarily; i.e., $S = \{p_1, \ldots, p_n\}$. Let $S_0 = S$. For $k = 1, \ldots, N$ check whether $S_{k-1} \backslash \{p_k\}$ contains a point in the ϵ-neighborhood of p_k (this is a local test). If yes, then let $S_k = S_{k-1} \backslash \{p_k\}$, otherwise let $S_k = S_{k-1}$. If S is ϵ-clustered then this procedure yields a set S_N which ϵ-covers the original set $S = S_0$ and whose points are separated by a distance of at least ϵ. Notice that on the set on the right this greedy procedure prunes all points except for one (p_8).

from p_1 to p_2 and $N(p, q)$ denotes the number of paths between p and q. The phenomenon of multiple responses to the same object, illustrated in Fig. 3, seems to arise in many domains [26, 4, 5, 2].

Whereas optimization methods deal with multiple responses in a quite natural way by simply retaining a unique response, the more general enumeration approach pursued here has to deal with the combinatorial redundancy problem in a more explicit way. We suspect that the issue of eliminating redundancies is a fundamental one when one is attempting to represent ambiguities explicitly. Granted that the information available at the perceptual organization level is insufficient to resolve certain ambiguities, it is natural to ask whether this information is at least sufficient to represent these ambiguities efficiently. A way to answer this question affirmatively is to assume that the available information makes it possible to enforce a *clustering* condition (see Fig. 1) on the set of all possible interpretations. This condition ensures that redundant interpretations can be pruned by means of local computations without compromising the completeness of the computed representation. Here, completeness refers to the requirement that all viable interpretations be (approximately) represented.

This paper presents a provably correct algorithm for the detection of salient cycles with the intent to illustrate the above issues both theoretically and experimentally. Here “salient cycles” refers to cycles with certain geometric properties for which the probability of being ϵ-near to a *scene contour* (i.e., a contour in the image which corresponds to a boundary in the scene) is sufficiently high, as encoded by a certain function $P_\epsilon(\pi)$. It is not our goal to demonstrate that this specific function captures the human perception of saliency.

The paper is organized as follows. Some previous related work is reviewed in Section 2. Section 3 describes the graph data-structure on which the algorithm operates, and Section 4 describes the algorithm and states the main theoretical result. Experiments are shown in the next section, which is followed by the conclusions. The appendices contain formal definitions and the proof of the main theorem.

2. PREVIOUS WORK

The problem of representing salient contours by searching paths in a graph has been approached by means of dynamic programming methods in [32, 34, 12, 31]. The algorithm proposed here provides a technique which overcomes the limitations of dynamic programming methods. First, multiple paths between two vertices can be represented; second, the cost function does not need to be decomposable into a sum over the vertices of the path. The structure of the proposed algorithm is quite different from dynamic programming and is based on recursive path concatenation. The price for removing these restrictions is that more memory resources are needed since paths need to be stored explicitly.

An approach to perceptual organization similar to ours, i.e., where the goal is to detect all grouping satisfying a certain criterion, has been proposed in [18]. The algorithm described there provably recovers all convex sets of line segments in which the length of the line segments accounts for at least some given proportion of the length of the convex hull. Our work places more emphasis on the elimination of the combinatorial redundancies caused by multiple responses to the same contour (see also [4, 5]). Similar issues have also been studied in the context of object recognition [2].

Other graph-based grouping algorithms have been proposed in the context of figure–background separation [1, 33, 27]. These algorithms compute a bipartition of the graph which minimizes some measure of the similarity between the two partitions relative to the similarity *within* each partition.

Recursive composition of contour descriptors, which is an important ingredient of the algorithm described here and of the more general approach of which it is part [3, 5, 6], is a well-known methodology for visual interpretation [22, 30, 11, 23]. Usually, these descriptors are organized into hierarchies of boundary representations of increasing complexity, edgel-points, contour fragments, regions, etc. (see, for instance, Fig. 4). The contour models underlying these descriptors become more general as one moves up in the hierarchy. For instance, the low level might contain only smooth and high contrast contours whereas higher levels model also illusory contours and singularities (corners and junctions). In this hierarchical approach, uncertainties which can not be resolved at one level are propagated to the higher levels, where more contextual information is available to resolve them.

Typically, perceptual organization cues are employed at different levels of the hierarchy according to their scale. Proximity and smoothness occur at a small scale and are utilized at the lowest levels of the hierarchy. Colinearity and cocircularity occur at different scales depending on the size of the gap between contour fragments and on the curvature of contours. Many algorithms exploit colinearity and cocircularity [26, 24, 25, 16, 15, 20]. A semilocal convexity constraint can significantly simplify the search for contours [18], and convex contours can be used as primitives to construct more complex contours. Closure [12] and symmetry [8, 9] are global properties of contours. Still at a higher level, the occlusion relationship between sets of contours [24, 25, 37, 13, 36] can be exploited.

The strategy of deferring hard decisions (if any) until sufficient "soft" information has been processed is widespread in perceptual organization algorithms. For instance, the Hough transform and similar methods collect votes in a suitable contour parameter space and then detect the most voted contours [15]. Voting methods can also be used to detect sets of contour fragments with similar attributes [30]. Some approaches estimate explicitly the probabilities of contour hypotheses and use these estimates to focus the search [28, 10, 8, 14, 12]. Relaxation labeling is a powerful iterative technique to propagate and accumulate soft information [26]. Convolution with suitable kernels is a biologically motivated approach for accumulating evidence of contours [16, 20]. Spectral graph methods use the weights on the arcs of a graph to encode the similarity between image regions in a "soft" fashion, before a segmentation of the image is computed [33, 20]. Approaches motivated by statistical physics represent relationships between contour fragments by means of coupling constants and then formulate perceptual organization as a combinatorial optimization problem which can be solved by using techniques borrowed from statistical mechanics [17, 13]. A combinatorial formulation by means of integer programming has also been proposed [35, 36].

3. THE CONTOUR GRAPH

The input to the proposed algorithm consists of a triple (C, A, P_ϵ), where (C, A) is a directed graph whose vertices C are contour fragments and P_ϵ, $\epsilon > 0$, is a family of $[0, 1]$-valued functions defined on the family of paths in (C, A). For every path π, $P_\epsilon(\pi)$ represents the probability that there exists a scene contour ϵ-near the path π. The algorithm also requires the specification of two thresholds, $\epsilon > 0$ and $\delta > 0$. Paths with the same end-vertices and which are less than ϵ apart are considered to be generated by the same scene contour. A path π for which $P_{2\epsilon}(\pi)$ is less than δ will be neglected and pruned out by the algorithm.

An example of a contour graph employed by the proposed algorithm is illustrated by Figs. 2 and 3. The multistage algorithm used to compute contour graphs is illustrated by Fig. 4 with a 16×16 subimage (this image contains only simple edges and therefore does not really motivate the need for contour graphs; see Figs. 2 and 3 for this purpose). The vertices of the contour graph (Fig. 2, top right, and Fig. 3, middle panels) consist of smooth fragments of contours where the brightness contrast is large enough to ensure reliable and efficient estimation by means of local operators. Here, "smooth" contour fragments refers to portions of contours away from singularities such as corners (including *curvature* discontinuities) and junctions.

To complete contours it is necessary to hypothesize several *hidden* contour fragments (see Fig. 2, bottom, and Fig. 3, right panels). These hidden contours represent portions of contours with low or zero brightness contrast and contour fragments near singularities such as corners and T-junctions. Hidden contours are represented by the arcs of the contour graph (C, A). An arc $a = (c_1, c_2)$ represents the hypotheses that the two contour fragments c_1 and c_2 are consecutive segments of the same contour.

In order for the contour graph to contain, with high probability, at least one path for every scene contour, it is necessary to hypothesize many hidden contours, which results in a large number of arcs in the graph. Most of the hypothesized hidden contours do not correspond to a scene contour (as most of the vertices do not). Moreover, it often occurs that contour fragments containing hidden portions are represented more than once. This is illustrated by the a-b-c-d and m-n-o-p portions of the lamp boundary (Fig. 3). Thus the challenge faced by the proposed algorithm is to compute a small number of paths in the graph (small relative to the total number of paths) in such a way that each scene contour is approximated by at least one path in the computed representation.

4. DESCRIPTION OF THE ALGORITHM

The strategy of the algorithm is to generate increasingly longer paths and to use the function $P_\epsilon(\cdot)$ to prune out implausible paths (the assumption being that more global information is available to test longer paths so that most of the long spurious paths will be eventually pruned out). Moreover, whenever two paths with the same end-vertices are less than ϵ apart from each other, one of the two paths is pruned out to eliminate redundancies. The ϵ-clustering condition is invoked to ensure that this does not compromise the completeness of the final representation. Figure 5 illustrates why nearby paths with different end-vertices should not be compressed down.

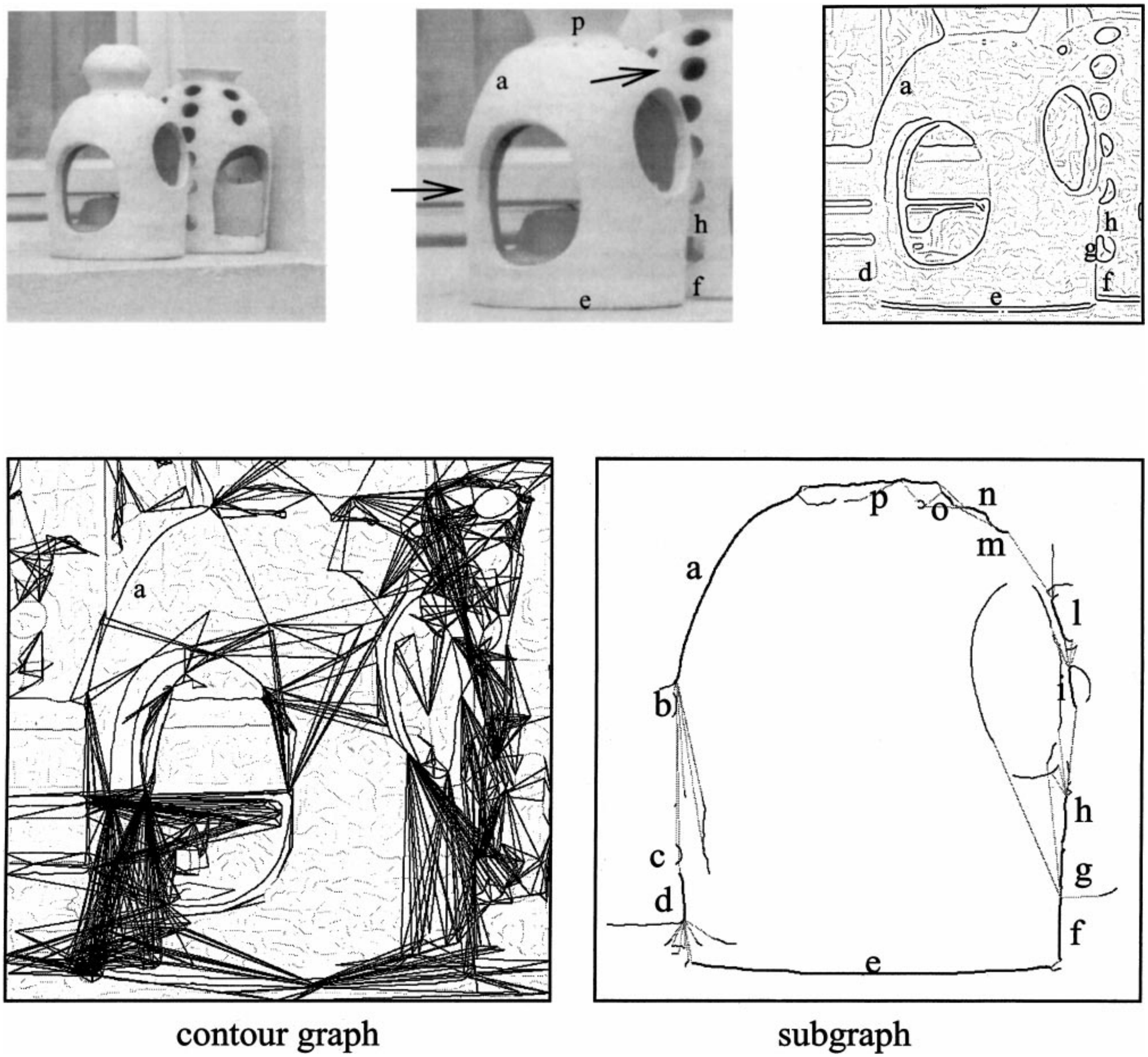


FIG. 2. Contour graph. (top) A brightness image (left), a 160 × 160 subimage of it, and the 196 polygonal contours extracted by the algorithm described in [5]. These contours are segmented into "smooth" primitives, which form the vertices of the contour graph (there are 1448 of them in this example). Most of these contours have very low brightness contrast (encoded into the gray level of the contours) and do not belong to any significant part of the scene. Notice, however, that some of the vertices on the contour of the lamp ("d," "h," top right) have low contrast too. (bottom-left) The arcs of the graph, which correspond to the hypothesized hidden contours. For instance, notice the six arcs leaving from the lower end of the vertex "a" (more clearly visible in the bottom-right panel and in Fig. 3). (bottom-right) Fourteen vertices ("a" through "p") forming the contour of the lamp. For each of these vertices, all the out-arcs and the relative vertices are also shown.

One component of the algorithm organizes the construction of increasingly longer paths in a systematic fashion. To this purpose, the set of vertices C is given an arbitrary "raster" order, represented by the integer-valued function $\rho\colon C \to [1, \ldots, N]$, where N is the cardinality of C. This function specifies the order in which vertices are "visited" by the algorithm. Right before the kth step of the algorithm, $1 \leq k \leq N$, only paths traversing vertices with raster indices less than k are allowed. Then, the kth iteration concatenates paths terminating at the kth vertex with paths starting at the kth vertex so that paths traversing the first k vertices are allowed at the end of the kth iteration.

To simplify the presentation of the algorithm we only consider the representation of closed contours. This assumption eliminates the problem of testing whether a contour is maximal. Thus the objective of the algorithm is to generate a sampling of the set of cycles in the graph so that every cycle corresponding to a scene contour is approximated by a cycle in the computed

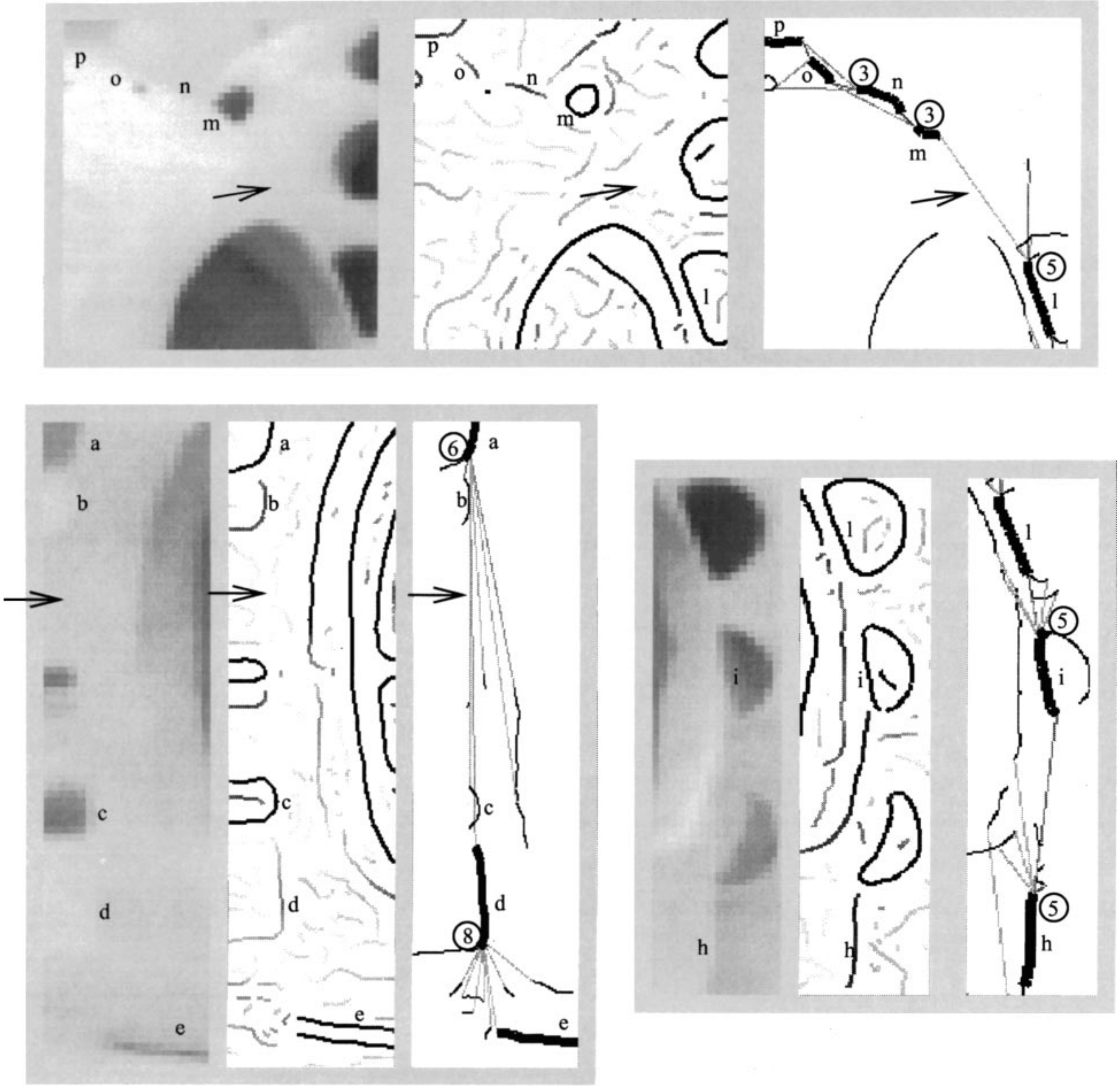


FIG. 3. Three enlargements from Fig. 2. The panels in each enlargement illustrate the subimage, the vertices of the graph, and the arcs, respectively. In the middle panels the gray level encodes the contrast of the contours. In the right panels vertices are black and arcs are gray. Thicker contours are those belonging to the lamp boundary in Fig. 2, bottom-right. The circled numbers denote the out-degree of a vertex. The arrows indicate the location of an illusory contour. Notice the multiple equivalent paths ("multiple responses to the same contour"): ad ≡ abcd ≡ acd; mnop ≡ mop ≡ mnp.

sample (with high probability). Furthermore, we restrict ourselves to simple cycles, that is, cycles with nonrepeating vertices. The set of simple paths in (C, A) is denoted $\mathcal{S}(C, A)$.

It should be noted that once the raster order function ρ has been fixed, each cycle can be recursively decomposed in a unique way by splitting paths at the vertex with highest raster index. This decomposition can be represented by a *parsing tree* (see Fig. 6). Conversely, each cycle can be uniquely composed by means of $l - 1$ concatenations by traversing the parsing tree in the other direction (bottom-up). Notice that the nodes of the parsing tree are paths π for which the raster indices at the extremities, denoted $\rho_{\text{fi}}(\pi)$ and $\rho_{\text{la}}(\pi)$, are larger than the maximum raster index of the internal vertices, denoted $\rho(\pi)$:

$$\rho_{\text{fi}}(\pi) > \rho(\pi), \quad \rho_{\text{la}}(\pi) > \rho(\pi). \tag{1}$$

Simple paths satisfying (1) will be said to be *ρ-regular* (or simply *regular*).

PROPOSITION 1. *Given a raster order ρ, every path $\pi \in \mathcal{S}(C, A)$ has a unique recursive decomposition into ρ-regular paths, specified by its parsing tree.*

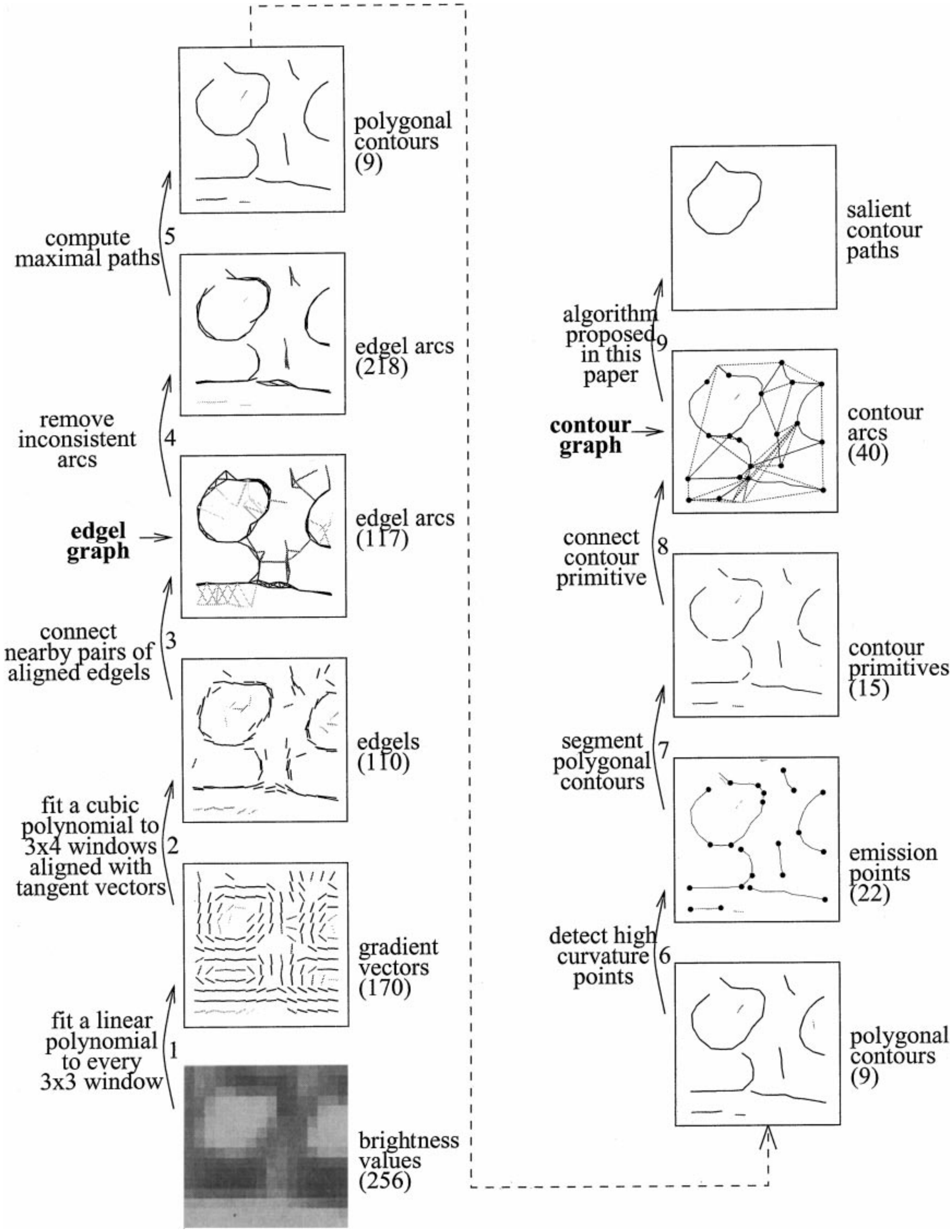


FIG. 4. The hierarchical algorithm used to compute contour graphs. The numbers between parentheses indicate the number of descriptors in each representation. (1) The brightness gradient is estimated at every location in the image. Gradient vectors whose magnitude is small compared to the estimated noise amplitude are discarded. (2) A cubic brightness model is used to estimate the contour location to subpixel accuracy and to perform more model-based pruning. (3) Edgels are composed into edgel-arcs (pairs of edgels). Only sufficiently aligned pairs of edgels are retained. (4) Some edgel-arcs are removed to eliminate divergent bifurcations (see [4, 5, 7]). (5) The polygonal contours consist of arbitrarily long chains of edgel-arcs. (6, 7) The polygonal contours are decomposed into contour primitives. (8) Pairs of contour primitives are composed into contour-arcs to bridge gaps due to lack of contrast along the boundary and contour singularities, such as corners and junctions. Several features (such as proximity, colinearity, and good continuation) are evaluated and used to discard unplausible contour-arc hypotheses. Compare with Figs. 2 and 3, which provide a more illustrative example of a contour-graph. (9) The algorithm proposed in this paper is used to detect salient cycles in the contour graph.

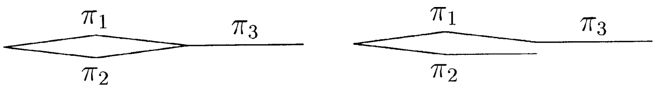


FIG. 5. The paths π_1 and π_2 represent multiple responses to the same edge. (left) Both π_1 and π_2 can be concatenated with π_3. Therefore, either path can be safely compressed out. (right) Here, π_2 can not be concatenated with π_3. Therefore, compression can not be done unless more complicated nonlocal tests are carried out to ensure that the surviving path yields a "maximally long" path.

TABLE 1
The Cycle Detection Algorithm

```
1. Initialization. $k = 0$. For every $1 \leq r, s \leq N$, let $X_{r,s}$ contain the length-one path with vertices $r$ and $s$ if $(r, s)$ is an arc of the graph; otherwise let $X_{r,s}$ be empty. If $r > s$, append the content of $X_{r,s}$ to $F_r$, else if $s > r$ append it to $L_s$.
2. Main loop. At time $k$, $1 \leq k \leq N$, do the following steps.
   3. Concatenation. Concatenate all the paths in $L_k$ with all the paths in $F_k$. Let $Z_k = L_k \circ F_k$ denote the resulting set of paths.
   4. Pruning. Remove from $Z_k$ all nonsimple paths and all paths for which $P_{2\epsilon}(\pi) < \delta$.
   5. For all $\pi \in Z_k$ do the following. Let $r = \rho_{\text{fi}}(\pi)$ $s = \rho_{\text{la}}(\pi)$.
      6. Update. Append $\pi$ to $X_{r,s}$. If $r > s$, append $\pi$ to $F_r$, else if $s > r$ append $\pi$ to $L_s$.
      7. Compression. Let $U_{r,s} \subset X_{r,s}$ be the set of paths with the same end-vertices as $\pi$ and which have distance from $\pi$ less than $\epsilon$. If $U_{r,s}$ is not empty then purge $\pi$ (i.e., remove $\pi$ from the arrays $F_r$, $L_s$, $X_{r,s}$ to which it belongs).
      7'. Compression'. If $\pi$ has higher probability than all the paths in $U_{r,s}$ then purge all the paths in $U_{r,s}$.
8. Return the set
   $$\bigcup_{r=1}^{N} X_{r,r}.$$
```

Note. Step **7′** is a possible refinement of **7** which makes it possible to replace the ϵ-clustering condition in Theorem 3 with a decay condition on $P_\epsilon(\cdot)$. (See discussion in Section 6.)

The detailed description of the algorithm is in Table 1. For $1 \leq r, s \leq N$, F_r, L_s, $X_{r,s}$ denote arrays of regular paths with first vertex r, last vertex s, and first, last vertices r, s, respectively. For simplicity, vertices will be identified with their raster indices. The following proposition can be easily proved by induction on k. Let $F_r[k]$, $L_s[k]$, $X_{r,s}[k]$ denote the set of paths in each of the arrays at the end of the kth iteration (assume $\rho(\pi) = 0$ if π does not have internal vertices).

Proposition 2. *For all $0 \leq k \leq N$ and $1 \leq r, s \leq N$ the following hold.*

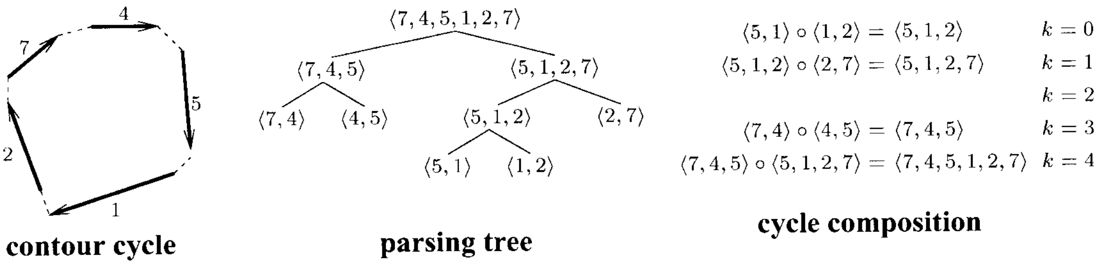
1. *If $\pi \in F_r[k]$, then $\rho(\pi) \leq k$ and $r = \rho_{\text{fi}}(\pi) > \rho_{\text{la}}(\pi) > k$.*
2. *If $\pi \in L_s[k]$, then $\rho(\pi) \leq k$ and $s = \rho_{\text{la}}(\pi) > \rho_{\text{fi}}(\pi) > k$.*
3. *If $r \neq s$ then $F_r[k] = \bigcup_s X_{r,s}[k]$ and $L_s[k] = \bigcup_r X_{r,s}[k]$.*
4. *The paths in $F_r[k]$, $L_s[k]$, $X_{r,s}[k]$ are regular.*
5. *If $\pi_1, \pi_2 \in X_{r,s}[k]$ then $d(\pi_1, \pi_2) \geq \epsilon$.*

Notice also that the construction of a path of lenght l requires exactly $l - 1$ concatenations and $l - 1$ probability evaluations (l if the path is a cycle).

The main theoretical result which characterizes the representation computed by the algorithm is that, with high probability (which can be made arbitrarily large by changing δ), every closed contour in the scene implicitly represented by a cycle in the graph (C, A) is *explicitly* approximated by one of the computed cycles in the arrays $X_{r,r}[N]$. It should be noted that by including a sufficiently large number of vertices and arcs in (C, A) one can guarantee with high probability that all scene contours are implicitly represented in (C, A) and, therefore, that they are also explicitly represented by the computed cycles (by virtue of the main theorem). Of course, there is a trade-off between computational resources and the size of (C, A), which translates into a trade-off between computational resources and the miss rate.

The following notation is used in the theorem (see the appendices for precise definitions and a proof). The set of scene contours is denoted Γ. The graph (C, A) is *ϵ-clustered* if the clustering condition (informally introduced in Fig. 1) is satisfied by the paths in (C, A). A cycle π is *ϵ-simple*, denoted $\pi \in \mathcal{S}^\epsilon(C, A)$, if none of its proper subpaths contains a contour which is ϵ-near a cycle in the graph. $X_{r,r}^{\epsilon,\delta}[k]$ denotes the cycles computed by the algorithm with thresholds ϵ and δ after the kth iteration. Let us assume that $\delta < (\bar{l} - 1)^{-1}$, where $\bar{l}$ is the maximum length of a cycle in (C, A).

Theorem 3. *Let (C, A) be an ϵ-clustered graph. Let $\pi \in \mathcal{S}^\epsilon(C, A)$ be a cycle and let l be its length. Then, with probability*



4
7
5
2
1
contour cycle
⟨7, 4, 5, 1, 2, 7⟩
⟨7, 4, 5⟩
⟨5, 1, 2, 7⟩
⟨7, 4⟩
⟨4, 5⟩
⟨5, 1, 2⟩
⟨2, 7⟩
⟨5, 1⟩
⟨1, 2⟩
parsing tree
⟨5, 1⟩ ∘ ⟨1, 2⟩ = ⟨5, 1, 2⟩ k = 0
⟨5, 1, 2⟩ ∘ ⟨2, 7⟩ = ⟨5, 1, 2, 7⟩ k = 1
k = 2
⟨7, 4⟩ ∘ ⟨4, 5⟩ = ⟨7, 4, 5⟩ k = 3
⟨7, 4, 5⟩ ∘ ⟨5, 1, 2, 7⟩ = ⟨7, 4, 5, 1, 2, 7⟩ k = 4
cycle composition

FIG. 6. The parsing tree (center) and the sequence of concatenations (right) associated with a simple cycle with five vertices (left) with indices 1, 2, 7, 4, 5. Notice that the root of the tree is the sequence of vertices obtained by opening the cycle at the vertex with the highest raster index. The cycle can be composed from its arcs by traversing the tree bottom-up and by concatenating paths at vertices with increasing raster indices.

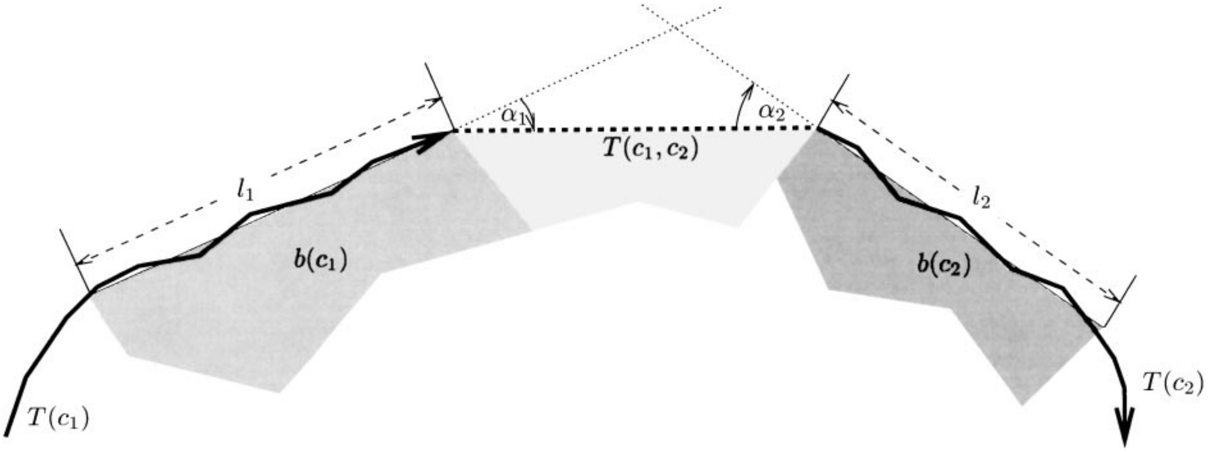


FIG. 7. A contour-arc $a = (c_1, c_2)$. Its probability $P_\epsilon(a)$ depends on l_1, l_2, α_1, α_2, and the brightness values $b(c_1)$ and $b(c_2)$.

at least $1-(l-1)\delta$, *if there exists a scene contour* $\gamma \in \Gamma$ *such that* $d(\gamma, \pi) \leq \epsilon$ *then there exists a cycle*

$$\hat{\pi} \in X^{\epsilon,\delta}_{\rho(\pi),\rho(\pi)}[k], \quad k \geq \rho(\pi)$$

such that $d(\pi, \hat{\pi}) \leq \epsilon$ *and* $d(\gamma, \hat{\pi}) \leq 2\epsilon$.

5. EXPERIMENTAL RESULTS

5.1. The Probabilistic Saliency Function

The log-probability of paths, $\log P_\epsilon(\pi)$ is modeled as the sum of two terms, one local, given by a sum over the arcs of the path, and one global, which depends on global features of the path

$$-\log P_\epsilon(\pi) = \sum_{i=1}^{l} f_{\mathrm{arc}}(a_i) + f_{\mathrm{gl}}(\pi),$$

where $a_1, \ldots, a_l$ denote the arcs of π. The functions f_{arc} and f_{gl} are nonnegative, with large values indicating unlikely hypotheses. The global term takes into account global features of π, namely, self-intersection of the path (in which case $f_{\mathrm{gl}}(\pi) = \infty$) and a measure of convexity of the path. The local term has been constructed by using a model very similar to the one proposed in [12]. Each local contribution $f_{\mathrm{arc}}(a)$, $a = (c_1, c_2)$, depends on the following features (see Fig. 7):

- The lengths l_1 and l_2 of the longest straight-line segments which can be fitted to the polygonal lines $T(c_1)$ and $T(c_2)$ with a given upper bound on of the fitting error (see Fig. 7).
- The length of the straight-line interpolant $T(c_1, c_2)$.
- The two orientation changes α_1 and α_2 induced by the interpolation.
- The difference $|b(c_1) - b(c_2)|$ of the estimated image brightness on the foreground side of the two polygonal descriptors c_1 and c_2.

The score $f_{\mathrm{arc}}(a)$ is computed by considering three possible reasons a gap occurs in a contour:

- The contour was originally connected at the polygonal-contour level and was split during the computation of the contour primitives (stages 6 and 7 in Fig. 4). In this case $f_{\mathrm{arc}}(a)$ is set to zero.
- The contour is split because of a sharp orientation or curvature change (corner). We expect potentially large values of α_1 and α_2 but small values of $|b(c_1) - b(c_2)|$ and of the length of $T(c_1, c_2)$.

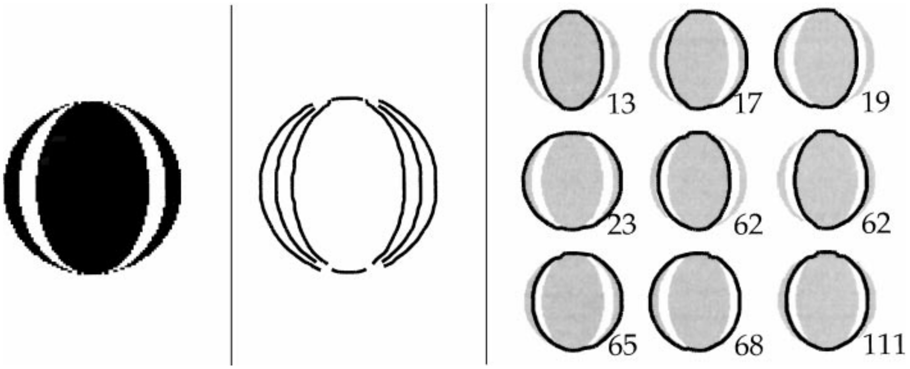


FIG. 8. An image (left) in which multiple closed contours pass through the same two points. Three of these contours (those which are symmetric with respect to the vertical axis) are perceptually salient. The algorithm described in this paper computes the nine contours shown on the right.

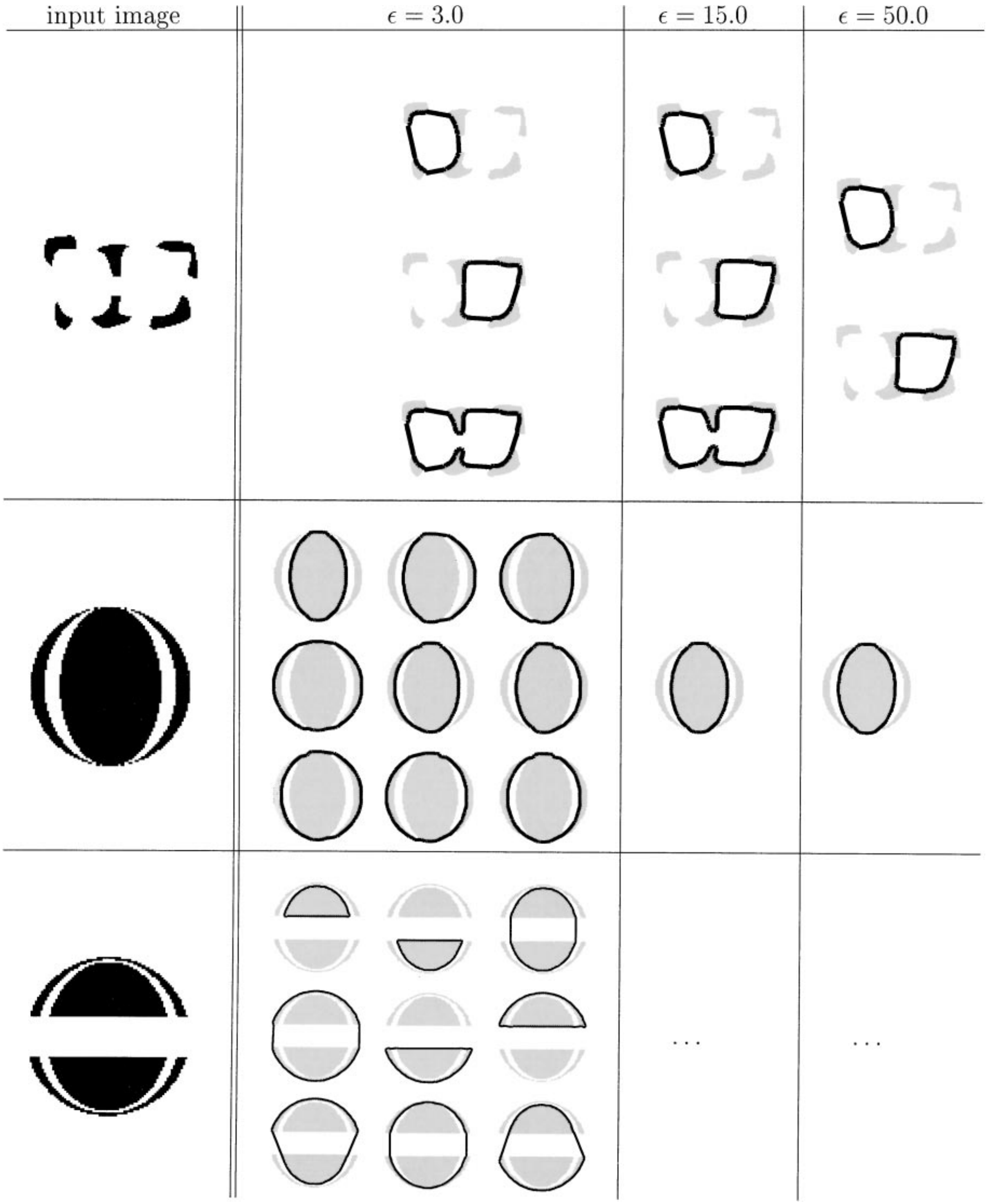


FIG. 9. Dependency on the compression parameter ϵ. Cycles produced for $\epsilon = 3.0$, $\epsilon = 15.0$, and $\epsilon = 50.0$. In the last row, the output does not depend on ϵ. In first row, the convexity constraint in the cost function was switched off.

- The contour is split because of a loss of contrast. The length of $T(c_1, c_2)$ can be large but the angles α_1 and α_2 should be small.

A probability estimate is computed for each of the hypotheses and $f_{\mathrm{arc}}(a)$ is set to the largest of these values.

5.2. *Experiments*

First, some controlled experiments with synthetic images will be described. Figure 8 illustrates the representation obtained by the algorithm on an image which contains contour fragments belonging to several closed contours. The middle panel shows the

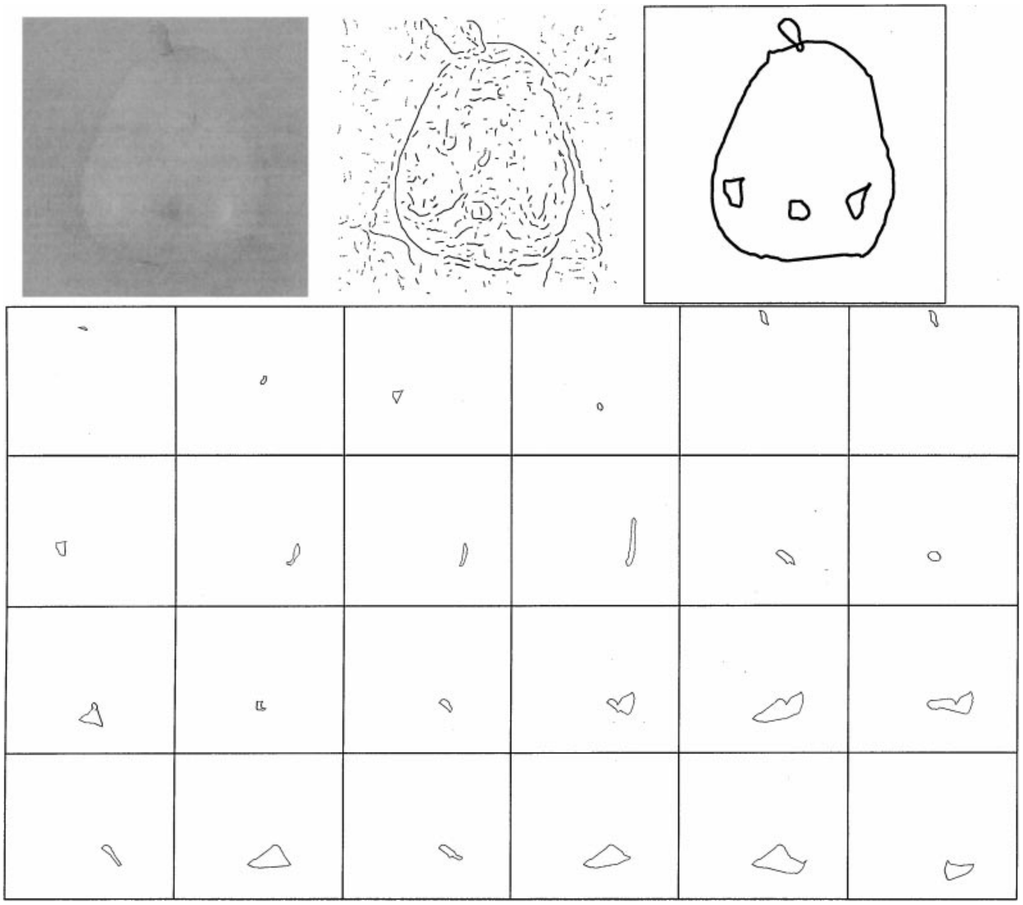


FIG. 10. Pear image, obtained from Lance Williams' web site. The algorithm generates 31 cycles. Five of the cycles are shown in the top right panel. Twenty-four of the remaining ones are shown below. The polygonal lines from which the vertices of the contour graph have been obtained is shown in the top middle panel.

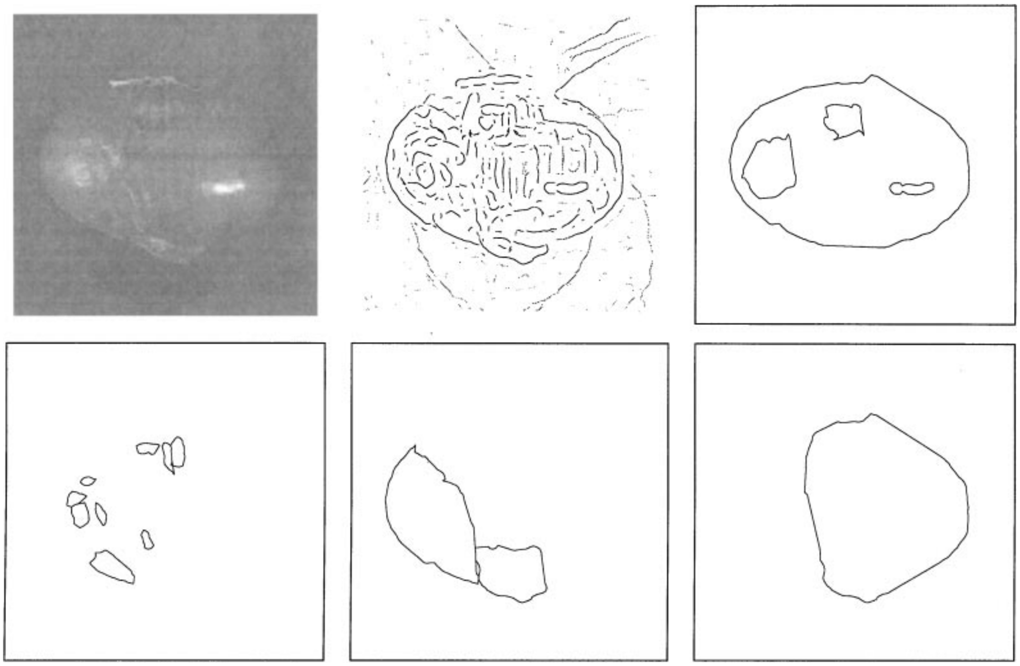


FIG. 11. Onion image, obtained from Lance Williams' web site. The algorithm generates 34 cycles. Four of the cycles are shown in the top right panel. Thirteen of the other ones are shown below. The polygonal lines from which the vertices of the contour graph have been obtained are shown in the top middle panel.

eight vertices of the contour graph. The two short horizontal contour fragments (top and bottom of the image) belong to all of the nine cycles computed by the algorithm (right panel). Numbers indicate the cost of each cycle, defined as minus log probability (conveniently scaled). An optimization algorithm would have produced only one interpretation (the one with lowest cost).

To illustrate how compression affects the representation computed by the algorithm, experiments with three different values of ϵ have been carried out on three different synthetic images (Fig. 9). The cost function was not changed as ϵ varied. Recall that the compression stage removes a newly created path whenever an existing path with the same end-vertices exists in its ϵ neighborhood. The first column contains the three images used for the test. The other three columns show the contours computed for $\epsilon = 3.0$, $\epsilon = 15.0$, and $\epsilon = 50.0$ (in pixel units). For the image in the first row, three contours are produced in the first two trials. Notice that the largest contour shares several vertices with each of the two other contours. When $\epsilon = 50.0$ the largest contour is not produced. This is an indication that the distance between the largest contour and the smaller ones is between 15.0 and 50.0. For the second image (the same used for the first experiment) nine contours are produced for $\epsilon = 3.0$ and only one for larger values of ϵ. Finally, in the last row, the nine lowest-cost contours do not share more than one vertex so that they are never compressed out. Notice that three of the contours (third, fourth, and eighth) are very close to each other.

The remaining experiments have been carried out on three natural images. (See Figs. 10–13.) The threshold δ has been set by trial and error so that the algorithm terminates in a few minutes. The compression parameter ϵ has been varied between 3.0 and 12.0 while the cost function has been kept unchanged. The number of cycles produced by the algorithm is in the range 20–120. Each contour graph contains a few hundred vertices and approximately the same number of hidden contour hypotheses

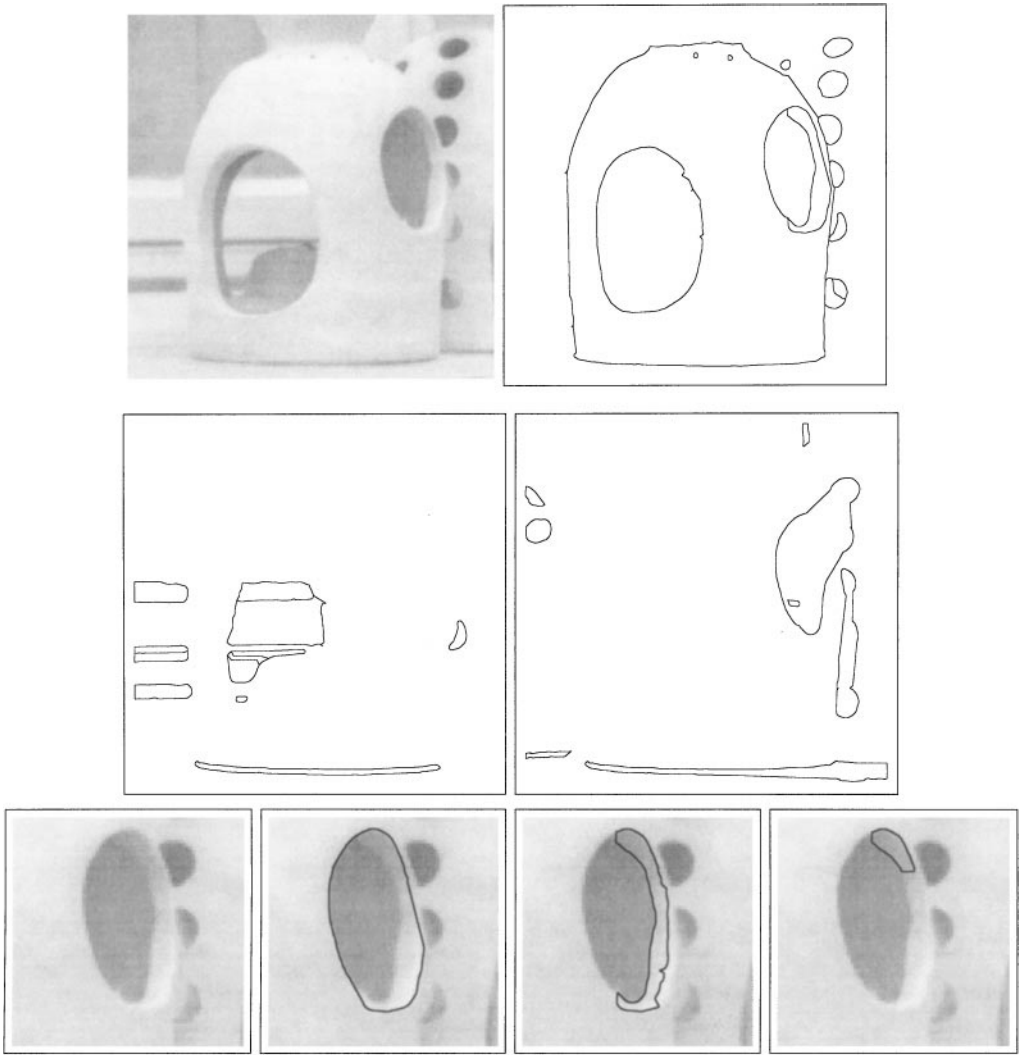


FIG. 12. (top-right) Fourteen of the computed cycles. (center-left) Twelve of the 21 spurious cycles corresponding to uniform image patches. (center-right) Eight of the 19 remaining spurious cycles. (bottom row) Three cycles sharing vertices.

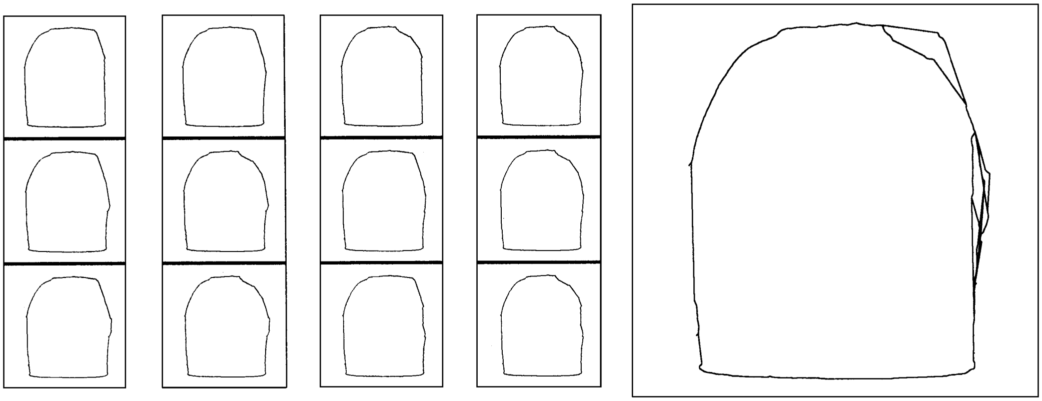


FIG. 13. Twelve redundant cycles corresponding to the lamp boundary obtained with $\epsilon = 3.0$. For $\epsilon = 12.0$, these paths are compressed down to two paths.

(contour arcs). The number of paths constructed and evaluated by the algorithm is a few thousand. A significant portion of these paths (between a quarter and a half, approximately) is removed by the compression step. At the end, almost half of the connected pairs of vertices have two or more paths connecting them.

The cost of the computed cycle does not carry much information about their saliency. Therefore, the most significant cycles have been selected manually and shown separately from the spurious ones. In this regard, we would like to stress that the purpose of these experiments is not to show that the proposed algorithm is able to assess the relative saliency of contours. Rather, the goal is to generate a small set of contours which contains with high probability all the desired contours. More suitable saliency functions can be used to rank the computed cycles according to human saliency.

6. CONCLUSIONS AND FUTURE WORK

The problem of eliminating redundancies when computing an intrinsically ambiguous representation has been discussed in the context of the detection of salient cycles in a contour graph. We have argued that these redundancies can lead to combinatorial explosion of the search space if not quickly removed.

A critical assumption needed to perform redundancy compression reliably is the ϵ-clustering condition. We believe that this condition can be replaced by a decay condition on the pruning function $P_\epsilon(\pi)$ similar to the decay conditions introduced in [5, 7]. Roughly speaking, such a decay condition would require the function $P_\epsilon(\pi)$ to attain a local maximum (in a noise robust sense) in the vicinity of scene contours. If such a condition holds, and if the variant **7**$'$ is used to perform compression (Table 1), we believe that one can prove the main result without requiring the ϵ-clustering condition.

The proposed algorithm reconstructs each cycle with a minimal number of concatenations, and in this sense, it requires a minimal number of primitive operations to explore the set of cycles in the graph. The formalism used here can be extended to detect open paths by introducing one special vertex, c_∞, and by connecting it to every other vertex. An open path $\pi = \langle c_0, \ldots, c_l \rangle$ is then represented by the cycle $\tilde{\pi} = \langle c_\infty, c_0, \ldots, c_1, c_\infty \rangle$ in the extended graph. However, unless the extended pruning function $P_\epsilon(\pi)$ encodes information about maximal paths, there might be an increase in computation proportional to the length of the open maximal paths. To see why this might be the case, notice that an open maximal path π of length l contains $l(l-1)$ subpaths which might have to be explicitly reconstructed if the end-vertices of π are not known.

In the experiments carried out with the current implementation, the introduction of the convexity constraint in the cost function was essential to control the complexity of the search. This suggests that more general shapes might need to be modeled as piecewise-convex contours and be reconstructed from a layer of convex components.

APPENDIX A: DEFINITIONS AND NOTATION

1. Contours. A *contour* c is an oriented one-dimensional manifold embedded in the real plane $\mathbb{R}^2$. A contour is *closed* if it is homeomorphic to a circle and is *open* otherwise (i.e., if it is homeomorphic to a straight line segment). See Fig. 14. Let $T(c) \subset \mathbb{R}^2$ be the set of points of the manifold c. The set $T(c)$ is called the *trace* of c.

2. Contour Graph. A graph (C, A) where C is a set of contours is called a *contour graph*. For any arc $a = (c_1, c_2) \in A$, let $T(a) = T(c_1, c_2)$ be the straight line segment which connects the "head" of c_1 to the "tail" of c_2. A path in (C, A), called a *contour path*, is denoted $\pi = \langle c_0, \ldots, c_l \rangle$, where l is the length of the path. The trace of π is

$$T(\pi) = T(c_0) \cup T(c_0, c_l) \cup \cdots \cup T(c_{l-1}, c_l) \cup T(c_l).$$

The contour path π is said to be *simple* if $T(\pi)$ is the trace of a contour, namely, if $T(\pi)$ is the set of points of an embedded manifold in $\mathbb{R}^2$. Let $\mathcal{S}(C, A)$ be the set of simple paths in the contour graph (C, A). The contour path π is a *simple cycle* if

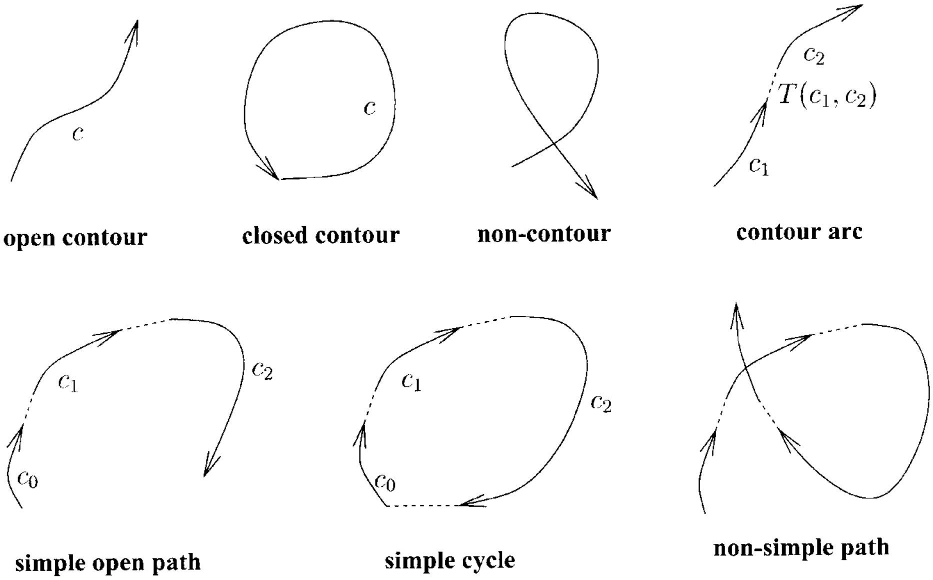


FIG. 14. Different types of contours and contour paths.

$T(\pi)$ is homeomorphic to a circle and is a *simple open* path if $T(\pi)$ is homeomorphic to a straight line segment (see Fig. 14). For any set of paths X let $\kappa_0 X$ denote the subset of simple paths: $\kappa_0 X = X \cap \mathcal{S}(C, A)$. If $\pi = \langle c_0, \ldots, c_l \rangle$ and $\pi' = \langle c'_0, \ldots, c'_{l'} \rangle$ are contour paths such that the last vertex of π_1 is the same as the first vertex of π_2, $c_l = c'_0$, then the concatenation $\pi \circ \pi'$ is given by:

$$\pi \circ \pi' = \langle c_0, \ldots, c_l = c'_0, \ldots, c'_l \rangle.$$

A contour γ is said to be *embedded* in a path π if $T(\gamma) \subset T(\pi)$; it is embedded in (C, A) if it is embedded in a path of (C, A).

3. Distance Function. For any two contours c_1, c_2, let $H(c_1, c_2)$ be the set of all homeomorphisms between c_1 and c_2. For any $\mu \in H(c_1, c_2)$, define

$$d_\mu(c_1, c_2) = d_\mu(T(c_1), T(c_2)) = \max_{p \in T(c_1)} \|p - \mu(p)\|$$

and let

$$d(c_1, c_2) = \min_{\mu \in H(c_1, c_2)} d_\mu(c_1, c_2).$$

It can be proved that d is a metric and that $d(c_1, c_2)$ is greater or equal to the Hausdorff distance between $T(c_1)$ and $T(c_2)$. If π_1 and π_2 are simple contour paths, then let $d(\pi_1, \pi_2) = d(T(\pi_1), T(\pi_2))$.

Proposition 4. *Let c_1, c_2 be contours and let c'_2 be a subcontour of c_2. Then there exists a subcontour of c_1, denoted c'_1, such that*

$$d(c'_1, c'_2) \leq d(c_1, c_2).$$

Proof. Let $\mu \in H(c_1, c_2)$ be the homeomorphism which achieves the distance $d(c_1, c_2)$ and let c'_1 be the subcontour of c_1 defined by $\mu(T(c'_1)) = T(c'_2)$. We have

$$d(c'_1, c'_2) \leq \max_{p \in T(c'_1)} \|p - \mu(p)\| \leq \max_{p \in T(c_1)} \|p - \mu(p)\| = d(c_1, c_2).$$

■

Proposition 5. *Let $\pi_1, \pi_2, \pi'_1, \pi'_2$ be simple contour paths such that $\pi_1 \circ \pi_2$ and $\pi'_1 \circ \pi'_2$ are defined and simple. Then,*

$$d(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2) \leq \max(d(\pi_1, \pi'_1), \quad d(\pi_2, \pi'_2)). \tag{2}$$

Proof. Let $\mu_1 \in H(\pi_1, \pi'_1)$, $\mu_2 \in H(\pi_2, \pi'_2)$, and let $\mu_1 \circ \mu_2 \in H(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2)$ be defined in the obvious way. Then we have

$$d_{\mu_1 \circ \mu_2}(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2) = \max(d_{\mu_1}(\pi_1, \pi'_1), \quad d_{\mu_2}(\pi_2, \pi'_2)).$$

Since $\mu_1 \circ \mu_2 \in H(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2)$ for all $\mu_1 \in H(\pi_1, \pi'_1)$, $\mu_2 \in H(\pi_2, \pi'_2)$, we have

$$\begin{aligned} & d(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2) \\ & = \min_{\mu \in H(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2)} d_\mu(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2) \\ & \leq \min_{\mu_1 \in H(\pi_1, \pi'_1), \mu_2 \in H(\pi_2, \pi'_2)} d_{\mu_1 \circ \mu_2}(\pi_1 \circ \pi_2, \pi'_1 \circ \pi'_2) \\ & = \min_{\mu_1 \in H(\pi_1, \pi'_1)} \min_{\mu_2 \in H(\pi_2, \pi'_2)} \max\bigl(d_{\mu_1}(\pi_1, \pi'_1), \quad d_{\mu_2}(\pi_2, \pi'_2)\bigr) \\ & = \max(d(\pi_1, \pi'_1), d(\pi_2, \pi'_2)). \quad ■ \end{aligned}$$

4. Scene Contours. Let I be the observed image and let (C_I, A_I) be the contour graph computed on input I by some deterministic algorithm. The information which we wish to extract

from the observed image will be assumed to be a set of contours, called *scene contours*, and will be denoted Γ. The set Γ will be assumed to contain all the subcontours of its elements.

$$\gamma \in \Gamma,\ T(\gamma') \subset T(\gamma) \Rightarrow \gamma' \in \Gamma.$$

Thus, Γ is really the set of all the contour fragments in the scene.

5. Probabilities of Paths. The image I and the image contours Γ will be assumed to be joint random variables. Since the contour graph (C_I, A_I) is a function of I, (C_I, A_I) is also jointly distributed with Γ and I. For any path π in (C_I, A_I) and $\epsilon > 0$ let $P_\epsilon(\pi \mid I)$ be the conditional probability that there exists $\gamma \in \Gamma$ such that $d(\gamma, \pi) < \epsilon$. Since I will be fixed once and for all, we will also use the notation $P_\epsilon(\pi) = P_\epsilon(\pi \mid I)$. For any $\epsilon > 0$ and $0 \le \delta \le 1$, let $\kappa_{\epsilon,\delta}$ be the pruning operator which removes nonsimple paths and paths for which the probability $P_\epsilon(\pi)$ is less than δ:

$$\kappa_{\epsilon,\delta} X = \{\pi \in X : P_\epsilon(\pi) \ge \delta\} \cap \mathcal{S}(C, A). \tag{3}$$

6. Raster Order Function ρ. Let $\rho \colon C \to \{1, \ldots, N\}$ be a bijective map. For simplicity, a vertex $c \in C$ will be identified with its raster index $\rho(c)$. If π is a simple path, then let $\rho(\pi)$ be the maximum raster index among its internal vertices. Thus, if π is a simple cycle, then $\rho(\pi)$ is the maximum index of all its vertices because all its vertices are internal. If $\pi = \langle c_0, \ldots, c_l\rangle$ is open, then

$$\rho(\pi) = \begin{cases} \max\limits_{1 \le j \le l-1} \rho(c_j) & \text{if } l \ge 2 \\ 0 & \text{if } l = 1. \end{cases} \tag{4}$$

If $\pi = \langle c_0, \ldots, c_l\rangle$ is open let $\rho_{\text{fi}}(\pi) = \rho(c_0)$, $\rho_{\text{la}}(\pi) = \rho(c_l)$. If π is a simple path, let $c^*_\pi(1)$ be the internal vertex of π with maximum index; that is, $\rho(c^*_\pi(1)) = \rho(\pi)$. Similarly, $c^*_\pi(2)$ is the unique internal vertex with second highest index.

7. ρ-Decomposition of Simple Paths. If $\pi = \langle c_0, \ldots, c_l\rangle$ is a simple open path, then its *ρ-decomposition* is given by

$$\pi = \langle c_0, \ldots, c^*_\pi(1)\rangle \circ \langle c^*_\pi(1), \ldots, c_l\rangle.$$

If π is a simple cycle, then its ρ-decomposition is given by

$$\pi = \langle c^*_\pi(1), \ldots, c^*_\pi(2)\rangle \circ \langle c^*_\pi(2), \ldots, c^*_\pi(1)\rangle.$$

The parsing tree of a simple path is obtained by applying this decomposition recursively $l - 1$ times (Fig. 6). A path π is *regular* if it is a simple cycle or if it is a simple open path such that $\rho_{\text{fi}}(\pi) > \rho(\pi)$ and $\rho_{\text{la}}(\pi) > \rho(\pi)$. Note that all the paths in a parsing tree are regular.

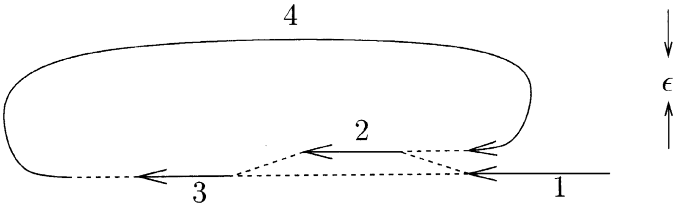


FIG. 15. The path $\pi = \langle 1, 3, 4, 2\rangle$ is simple but not ϵ-simple. In fact, its trace $T(\pi)$ contains a subcontour ϵ-near the cycle with vertices 2, 3, 4.

8. Metric Properties of Sets of Paths. Let X, X_1, X_2 be finite sets of paths and let $\epsilon \ge 0$. The set X is *ϵ-separated* if for any $\pi_1 \in X, \pi_2 \in X$, we have $d(\pi_1, \pi_2) > \epsilon$. The set X_2 is an *ϵ-covering* of X_1, if for every $\pi_1 \in X_1$, there exists $\pi_2 \in X_2$ such that $d(\pi_1, \pi_2) \le \epsilon$. The set X_2 is an *ϵ-sampling* of X_1 if $X_2 \subset X_1$ and X_2 is an ϵ-covering of X_1. An ϵ-sampling is *minimal* if it is ϵ-separated.

The set X is *ϵ-clustered* if for any three paths $\pi_1, \pi_2, \pi_3 \in X$ $d(\pi_1, \pi_2) \le \epsilon$ and $d(\pi_2, \pi_3) \le \epsilon$ imply $d(\pi_1, \pi_3) \le \epsilon$. A contour graph is ϵ-clustered if every set of paths with the same end-points is ϵ-clustered.

9. Compressed-Union Operation. Let $X_1 \cup_\epsilon X_2$ denote an arbitrary minimal ϵ-sampling of $X_1 \cup X_2$. If X_1 is ϵ-separated, then such a set can be constructed as follows. Initialize $X := X_1$. Order the elements of X_2. Then repeat the following step for each $\pi_2 \in X_2$. If $d(\pi_1, \pi_2) > \epsilon$ for every $\pi_1 \in X_1$, then add π_2 to X, $X := X \cup \{\pi_2\}$. Finally, define $X_1 \cup_\epsilon X_2 = X$. Notice that if $\epsilon = 0$, then $X_1 \cup_\epsilon X_2 = X_1 \cup X_2$.

10. ϵ-Simple Paths. (See Fig. 15.) A simple open path π in a contour graph (C, A) is *ϵ-simple* if there exists no closed contour γ' embedded in (C, A) ϵ-near a contour γ embedded in π. That is, there exists no pair of contours (γ, γ'), such that $T(\gamma) \subset T(\pi)$; γ' is closed and embedded in (C, A); $d(\gamma, \gamma') \le \epsilon$. A simple cycle is said to be ϵ-simple if all its strict subpaths are ϵ-simple. The set of ϵ-simple paths in (C, A) is denoted $\mathcal{S}^\epsilon(C, A)$.

Proposition 6. *Let $\pi \in \mathcal{S}^\epsilon(C, A)$ be open and let π' be a path such that $d(\pi, \pi') \le \epsilon$. Then $\pi' \in \mathcal{S}(C, A)$.*

Proof. For the purpose of contradiction, let π' be not simple. Then there exists a closed contour γ' embedded in (C, A) such that $T(\gamma') \subset T(\pi')$. From $d(\pi, \pi') \le \epsilon$ and from Proposition 4, there exists a contour γ, $T(\gamma) \subset T(\pi)$, such that $d(\gamma, \gamma') \le \epsilon$. Since γ' is a closed contour embedded in (C, A), this contradicts the fact that π is an open ϵ-simple path.

Proposition 7. *A subpath of an ϵ-simple path is ϵ-simple.*

APPENDIX B: PROOF OF THE MAIN THEOREM

A compact way to characterize the cycle detection algorithm of Table 1 is by means of the discrete-time dynamical system

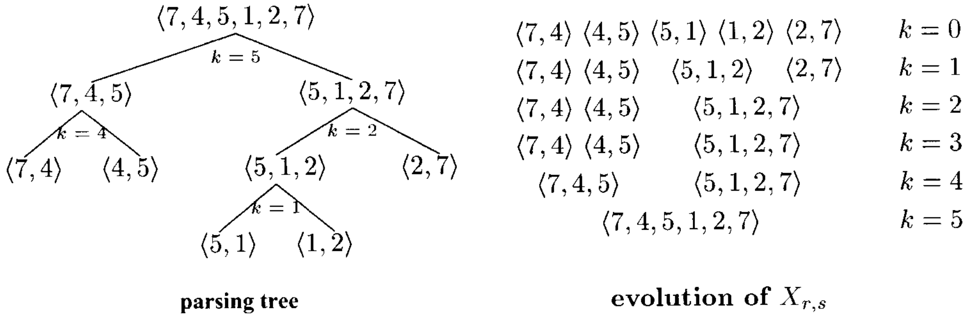

⟨7, 4, 5, 1, 2, 7⟩
k = 5
⟨7, 4, 5⟩
⟨5, 1, 2, 7⟩
k = 4
k = 2
⟨7, 4⟩
⟨4, 5⟩
⟨5, 1, 2⟩
⟨2, 7⟩
k = 1
⟨5, 1⟩
⟨1, 2⟩
parsing tree
⟨7, 4⟩ ⟨4, 5⟩ ⟨5, 1⟩ ⟨1, 2⟩ ⟨2, 7⟩ k = 0
⟨7, 4⟩ ⟨4, 5⟩ ⟨5, 1, 2⟩ ⟨2, 7⟩ k = 1
⟨7, 4⟩ ⟨4, 5⟩ ⟨5, 1, 2, 7⟩ k = 2
⟨7, 4⟩ ⟨4, 5⟩ ⟨5, 1, 2, 7⟩ k = 3
⟨7, 4, 5⟩ ⟨5, 1, 2, 7⟩ k = 4
⟨7, 4, 5, 1, 2, 7⟩ k = 5
evolution of X_{r,s}

FIG. 16. Evolution of the representation of the cycle of Fig. 6 by means of the variables $X_{r,s}[k]$. At time $k = 0$, the cycle is represented by five paths of length one. At all times $k \geq 5$, the cycle is represented by one path of length five.

$$X_{r,s}[0] = \begin{cases} \{\langle r, s\rangle\} & \text{if } \langle r, s\rangle \in \mathcal{S}(C, A) \\ \emptyset. & \text{otherwise,} \end{cases} \tag{5}$$

$$X_{r,s}[k+1] = \begin{cases} X_{r,s}[k] \cup_\epsilon \kappa(X_{r,k+1}[k] \circ X_{k+1,s}[k]), & \text{if } \min(r, s) > k+1 \\ X_{r,s}[k], & \text{otherwise,} \end{cases} \tag{6}$$

where $\kappa = \kappa_{2\epsilon,\delta}$, as given by (3). For any set of paths X, let $\kappa_0 X = X \cap \mathcal{S}(C, A)$. Let $\Omega_{r,s}$ be the regular paths with endpoints r and s:

$$\Omega_{r,s} = \{\pi \in \mathcal{S}(C, A) : \rho_{\text{fi}}(\pi) = r > \rho(\pi),\ \rho_{\text{la}}(\pi) = s > \rho(\pi)\}. \tag{7}$$

Figure 16 illustrates the evolution of the sets $X_{r,s}$ over time.

PROPOSITION 8. *Let $\epsilon = 0$ and $\kappa = \kappa_0$ in (6). Then,*

$$X_{r,s}[k] = \{\pi \in \Omega_{r,s} : \rho(\pi) \leq k\} \equiv X^0_{r,s}[k]. \tag{8}$$

Proof. The direction $X_{r,s}[k] \subset X^0_{r,s}[k]$ is given by Proposition 2 in Section 4. To prove $X^0_{r,s}[k] \subset X_{r,s}[k]$, let $\pi \in X^0_{r,s}[k]$ so that $\rho(\pi) < \rho_{\text{fi}}(\pi) = r$, $\rho(\pi) < \rho_{\text{la}}(\pi) = s$, and $\rho(\pi) \leq k$. Let us proceed by induction and let us assume $X^0_{r,s}[j] \subset X_{r,s}[j]$ for all $j \leq k-1$. Since for $\epsilon = 0$, $\cup_\epsilon = \cup$, we have from (6) $X_{r,s}[i] \subset X_{r,s}[k]$ if $i \leq k$. Therefore, since $\rho(\pi) \leq k$, it is sufficient to prove $\pi \in X_{r,s}[\rho(\pi)]$.

Let π_1, π_2 be the ρ-decomposition of π. Then, since $\rho(\pi_1) \leq \rho(\pi) - 1$, $\rho(\pi_2) \leq \rho(\pi) - 1$, we have $\pi_1 \in X^0_{r,\rho(\pi)}[\rho(\pi) - 1]$, $\pi_2 \in X^0_{\rho(\pi),s}[\rho(\pi) - 1]$, and therefore, from the inductive hypothesis, $\pi_1 \in X_{r,\rho(\pi)}[\rho(\pi) - 1]$, $\pi_2 \in X_{\rho(\pi),s}[\rho(\pi) - 1]$. Then,

$$\pi = \pi_1 \circ \pi_2 \in X_{r,\rho(\pi)}[\rho(\pi) - 1] \circ X_{\rho(\pi),s}[\rho(\pi) - 1].$$

Hence, from (6) at time $k = \rho(\pi) - 1$, $\pi \in X_{r,s}[\rho(\pi)]$. ■

The following theorem, which follows from Prop. 8, guarantees that the proposed algorithm, if no pruning or compression is applied, visits each simple cycle exactly once by using a minimal number of concatenations.

THEOREM 9. *Let π be a simple cycle in the contour graph (C, A) and let $\epsilon = 0$ and $\kappa = \kappa_0$ in (6). Then,*

$$\pi \in X_{\rho(\pi),\rho(\pi)}[k], \quad k \geq \rho(\pi).$$

Furthermore, the cycle π is obtained by means of the $l - 1$ concatenations specified by the parsing tree of π.

The following result provides a sufficient condition for the proposed algorithm to approximate all ϵ-simple cycles in the case in which compression is applied but pruning is not. Let $X^\epsilon_{r,s}[k]$ be given by (6) with $\kappa = \kappa_0$ and let $X^0_{r,s}[k]$ be given by (8).

LEMMA 10. *Let (C, A) be an ϵ-clustered graph. Let $\pi \in X^0_{r,s}[k] \cap \mathcal{S}^\epsilon(C, A)$. Then, there exists $\tilde{\pi} \in X^\epsilon_{r,s}[k]$ such that $d(\pi, \tilde{\pi}) \leq \epsilon$.*

Proof. From $\pi \in X^0_{r,s}[k]$ and Proposition 8 we have $r = \rho_{\text{fi}}(\pi)$ and $s = \rho_{\text{la}}(\pi)$. Let us proceed by induction. For $k = 0$ the statement is true because $X^0_{r,s}[0] = X^\epsilon_{r,s}[0]$ as given by (5). Let the statement be true for all $j \leq k-1$ and let $\pi \in X^0_{r,s}[k] \cap \mathcal{S}^\epsilon(C, A)$. Let $\pi = \pi_1 \circ \pi_2$ be the ρ-decomposition of π; that is, $\rho_{\text{la}}(\pi_1) = \rho_{\text{fi}}(\pi_2) = \rho(\pi)$. Thus,

$$\max(\rho(\pi_1), \rho(\pi_2)) \leq \rho(\pi) - 1 \leq k - 1.$$

Therefore, by the inductive hypothesis, there exist

$$\tilde{\pi}_1 \in X^\epsilon_{r,\rho(\pi)}[\rho(\pi) - 1], \quad \tilde{\pi}_2 \in X^\epsilon_{\rho(\pi),s}[\rho(\pi) - 1] \tag{9}$$

such that $d(\pi_1, \tilde{\pi}_1) \leq \epsilon$ and $d(\pi_2, \tilde{\pi}_2) \leq \epsilon$. Let $\tilde{\pi} = \tilde{\pi}_1 \circ \tilde{\pi}_2$. From Proposition 5, $d(\pi, \tilde{\pi}) \leq \epsilon$. Equation (6) for $k = \rho(\pi) - 1$ is

$$X^\epsilon_{r,s}[\rho(\pi)] = X^\epsilon_{r,s}[\rho(\pi) - 1] \cup_\epsilon \kappa_0\big(X^\epsilon_{r,\rho(\pi)}[\rho(\pi) - 1] \circ X^\epsilon_{\rho(\pi),s}[\rho(\pi) - 1]\big). \tag{10}$$

From (9), $\tilde{\pi} \in X^{\epsilon}_{r,\rho(\pi)}[\rho(\pi)-1] \circ X^{\epsilon}_{\rho(\pi),s}[\rho(\pi)-1]$. Since $\pi \in \mathcal{S}^{\epsilon}(C, A)$ and $d(\pi, \tilde{\pi}) \leq \epsilon$, we have from Proposition 6 that $\tilde{\pi}$ is simple so that $\tilde{\pi}$ is not pruned out by κ_0. Therefore, since for any sets X_1, X_2, $X_1 \cup_{\epsilon} X_2$ is an ϵ-sampling of $X_1 \cup X_2$, there exists $\pi' \in X^{\epsilon}_{r,s}[\rho(\pi)]$ such that $d(\tilde{\pi}, \pi') \leq \epsilon$. Since $d(\pi, \tilde{\pi}) \leq \epsilon$ and since (C, A) is ϵ-clustered, we have $d(\pi, \pi') \leq \epsilon$. From $\pi' \in X^{\epsilon}_{r,s}[\rho(\pi)]$, by using (6) and the ϵ-clustering hypothesis, from time $\rho(k)$ up to time $k-1$, it follows that there exists $\hat{\pi} \in X^{\epsilon}_{r,s}[k]$ such that $d(\pi', \hat{\pi}) \leq \epsilon$, and therefore $d(\pi, \hat{\pi}) \leq \epsilon$. ■

Proof of Theorem 3. Let $\pi_1, \ldots, \pi_{l-1} = \pi$, be the sequence of regular paths (with length greater than one) in the parsing tree of the cycle π. We have $\pi_i \in \mathcal{S}^{\epsilon}(C, A)$, $1 \leq i \leq l-1$. From Lemma 10, the unpruned dynamical system $X^{\epsilon}_{r,s}[k]$ ϵ-approximates each of the paths $\pi_1, \ldots, \pi_{l-1}$. That is, there exist $\hat{\pi}_1, \ldots, \hat{\pi}_{l-1}$ such that $\hat{\pi}_i \in X^{\epsilon}_{\rho_{\mathrm{fi}}(\pi_i), \rho_{\mathrm{la}}(\pi_i)}[\rho(\pi_i)]$ and $d(\pi_i, \hat{\pi}_i) \leq \epsilon$.

If there exists $\gamma \in \Gamma$ such that $d(\gamma, \pi) \leq \epsilon$ then, from Proposition 4, for each $i = 1, \ldots, l-1$ there exists $\gamma_i \in \Gamma$ such that $d(\gamma_i, \pi_i) \leq \epsilon$, and therefore, $d(\gamma_i, \hat{\pi}_i) \leq 2\epsilon$. By the definition of $\kappa_{2\epsilon,\delta}$, if $\hat{\pi}_i$ is pruned out, then the probability that there exists a scene contour 2ϵ-near it is less than δ. Therefore, the probability is less than δ that there exists $\gamma \in \Gamma$ such that $d(\gamma, \pi) \leq \epsilon$ and that π_i is pruned out. Hence, by using the union bound, the probability that there exists $\gamma \in \Gamma$, $d(\gamma, \pi) \leq \epsilon$, and that at least one of the paths π_i $i = 1, \ldots, l-1$ is pruned out is at most $(l-1)\delta$. Thus, with probability at least $1 - (l-1)\delta$, if there exists $\gamma \in \Gamma$, $d(\gamma, \pi) \leq \epsilon$, none of the paths $\hat{\pi}_1, \ldots, \hat{\pi}_{l-1}$ is pruned out, and therefore $\hat{\pi} \in X^{\epsilon,\delta}_{\rho(\pi),\rho(\pi)}[\rho(\pi)]$. ■

REFERENCES

1. A. Amir and M. Lindenbaum, Quantitative analysis of grouping processes, in *European Conference on Computer Vision*, pp. 1:371–384, 1996.
2. Louay Bazzi, *Efficient Detection of Planar Patterns Embedded in Cluttered Observations*, Master's thesis, MIT, 1998.
3. S. Casadei and S. K. Mitter, A hierarchical approach to high resolution edge contour reconstruction, in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, pp. 149–153, 1996.
4. S. Casadei and S. K. Mitter, Hierarchical curve reconstruction. Part 1: Bifurcation analysis and recovery of smooth curves, in *European Conference on Computer Vision*, pp. 199–208, 1996.
5. S. Casadei and S. K. Mitter, Hierarchical image segmentation—Part i: Detection of regular curves in a vector graph, *Int. J. Comput. Vision* **27**(3), March 1998, 71–100.
6. S. Casadei and S. K. Mitter, A perceptual organization approach to contour estimation via composition, compression and pruning of contour hypotheses, Technical Report LIDS-P-2415, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, April 1998.
7. S. Casadei and S. K. Mitter, An efficient and provably correct algorithm for the multiscale estimation of image contours by means of polygonal lines, *IEEE Trans. Inform. Theory* **45**(3), April 1999.
8. R. L. Castano and S. Hutchinson, A probabilistic approach to perceptual grouping, *Comput. Vision Image Understand.* **64**(3), Nov. 1996, 399–419.
9. T. J. Cham and R. Cipolla, Geometric saliency of curve correspondences and grouping of symmetric contours, in *European Conference on Computer Vision*, pp. I:385–398, 1996.
10. I. J. Cox, J. M. Rehg, and S. Hingorani, A Bayesian multiple-hypothesis approach to edge grouping and contour segmentation, *Int. J. Comput. Vision* **11**(1), Aug. 1993, 5–24.
11. J. Dolan and E. Riseman, Computing curvilinear structure by token-based grouping, in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, 1992.
12. J. H. Elder and S. W. Zucker, Computing contour closure, in *European Conference on Computer Vision*, pp. 399–411, 1996.
13. D. Geiger and K. Kumaran, Visual organization of illusory surfaces, in *European Conference on Computer Vision*, pp. I:413–424, 1996.
14. D. Geman and B. Jedynak, An active testing model for tracking roads in satellite images, *IEEE Trans. Pattern Anal. Machine Intell.* **18**(1), Jan. 1996, 1–14.
15. G. Guy and G. G. Medioni, Inferring global perceptual contours from local features, *Int. J. Comput. Vision* **20**(1/2), 1996, 113–133.
16. F. Heitger and R. von der Heydt, A computational model of neural contour processing: Figure-ground segregation and illusory contours, in *International Conference on Computer Vision*, pp. 32–40, 1993.
17. L. Herault and R. Horaud, Figure–ground discrimination: A combinatorial approach, *IEEE Trans. Pattern Anal. Machine Intell.* **15**(9), Sept. 1993, 899–914.
18. D. W. Jacobs, Robust and efficient detection of salient convex groups, *IEEE Trans. Pattern Anal. Machine Intell.* **18**(1), Jan. 1996, 23–37.
19. G. Kanizsa, *Organization in Vision: Essays on Gestalt Perception*, Praeger, New York, 1979.
20. Thomas Leung and Jitendra Malik, Contour continuity in region based image segmentation, in *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, 1998.
21. D. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic, Dordrecht/Norwell, MA, 1985.
22. D. Marr, *Vision*, Freeman, New York, 1982.
23. R. Mohan and R. Nevatia, Perceptual organization for scene segmentation and description, *IEEE Trans. Pattern Anal. Machine Intell.* **14**, June 1992.
24. M. Nitzberg and D. Mumford, The 2.1-d sketch, in *International Conference on Computer Vision*, pp. 138–144, 1990.
25. M. Nitzberg, D. Mumford, and T. Shiota, *Filtering, Segmentation, and Depth*, Vol. 662 of Lecture Notes in Computer Science, Springer-Verlag, Berlin/New York, 1993.
26. P. Parent and S. W. Zucker, Trace inference, curvature consistency, and curve detection, *IEEE Trans. Pattern Anal. Machine Intell.* **11**, Aug. 1989.
27. P. Perona and W. Freeman, An axiomatic approach to grouping, in *ECCV98*, 1998.
28. S. Sarkar and K. L. Boyer, Perceptual organization using Bayesian networks, in *IEEE Computer Vision and Pattern Recognition or CVPR*, pp. 251–256, 1992.
29. S. Sarkar and K. L. Boyer, Perceptual organization in computer vision: A review and a proposal for a classificatory structure, *IEEE Trans. Systems Man Cybernetics* **23**, 1993, 382–399.
30. S. Sarkar and K. L. Boyer, A computational structure for preattentive perceptual organization: Graphical enumeration and voting methods, *IEEE Trans. Systems Man Cybernetics* **24**, 1994, 246–267.
31. J. R. Serra and Brian Subirana, Adaptive non-Cartesian networks for vision, in *IX International Conference on Image Analysis and Processing, Firenze, Sept. 1997.*

32. A. Sha'ashua and S. Ullman, Structural saliency: The detection of globally salient structures using a locally connected network, in *International Conference on Computer Vision*, pp. 321–327, 1988.

33. J. Shi and J. Malik, Normalized cuts and image sementation, in *IEEE Computer Vision and Pattern Recognition*, pp. 731–737, 1997.

34. J. B. Subirana-Vilanova, Curved inertia frames and the skeleton sketch: Finding salient frames of reference, in *International Conference on Computer Vision*, pp. 702–708, 1990.

35. L. R. Williams and A. R. Hanson, Perceptual completion of occluded surfaces, in *IEEE Computer Vision and Pattern Recognition*, pp. 104–112, 1994.

36. L. R. Williams and A. R. Hanson, Perceptual completion of occluded surfaces, *Comput. Vision Image Understand.* **64**(1), July 1996, 1–20.

37. L. R. Williams and D. Jacobs, Stochastic completion fields: A neural model of illusory contour shape and salience, in *International Conference on Computer Vision*, pp. 408–415, 1995.

38. A. P. Witkin and J. M. Tenenbaum, On the role of structure in vision, in *Human and Machine Vision*, pp. 481–543, Academic Press, San Diego, 1983.