Proc. Int. Joint Conf. Artificial Intelligence, Milano, Italy, Aug. 1987.
[41] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot," in *Int. Symp. Robotics Research,* Santa Cruz ,CA, Aug. 1987.
[42] G. Randall, S. Foret, and N. Ayache, "Final steps towards real time trinocular stereo vision," in *Proc. First European Conf. Computer Vision, ECCV'90,* Antibes, France, O. D. Faugeras, Ed. New York: Springer-Verlag, Apr. 1990, pp. 601–603.
[43] N. Ayache, *Vision Stéréoscopique et Perception Multisensorielle; Applications à la robotique Mobile.* Inter-Editions, 1989.
[44] ——, *"Artifical Vison for Mobile Robots: Stereo Vision and Sensor Fusion.* Cambridge, MA: MIT Press, 1990.
[45] N. Ayache and O. D. Faugeras, "Building, registrating and fusing noisy visual maps," *Int. J. Robotics Res. (Special Issue on Sensor Data Fusion),* vol. 7, no. 6, pp. 45–65, Dec. 1988.
[46] P. Grossman, "Compact—A surface representation scheme," in *Proc. 4th Alvey Vision Conf. (AVC'88),* Manchester, England, 1988.

Shape Representation by Multiscale Contour Approximation

Ann Bengtsson and Jan-Olof Eklundh

Abstract—**We present an approach for deriving qualitative descriptions of contours containing structures at different (unknown) scales. The descriptions are in terms of straight arcs, curved arcs with sign of curvature, corners, and points delimiting the arcs: inflexion points and transitions from straight to curved. Furthermore, the tangents at these points are derived.**

The approach is based on the construction of a hierarchic family of polygons, having the scale-space property of causality: structure can only disappear as scale goes from fine to coarse. Using the principle that structures that are stable over scale represent *significant* properties, the features of the descriptive representations are then derived.

Index Terms— **Corners, hierarchic family of polygons, inflexion points, multiscale polygon approximations, qualitative contour description, scale stability, straight and curved arcs, tangent directions.**

I. Introduction

The goal of computational vision is to derive descriptions of a scene from images of it. In particular, the descriptions could be in terms of primitives representing the geometric structure of the world. There are several reasons why such descriptions are important.

In a world of coherent objects and at the level of surfaces and volumes with their bounding contours, the geometric cues given by the contours are of paramount importance. Furthermore, geometric information about, e.g., the occluding boundaries of surfaces impose very strong restrictions on the possible structure of the scene. In fact, geometric features tend to be much more useful than photometric features in the computation of what is in the scene. This seems to be true for monocular and binocular scenes as well as for time-varying scenes, see, e.g., [1]–[6].

Different approaches exist to deriving geometric structure. The work presented here should be seen in the context when the structure is reflected in image curves derived from the intensity data. The geometric properties of such curves could be obtained by direct use of the intensity information, as in the work on curve tracing and curvature by Zucker and his co-workers, see, e.g., [7], or by groupings of edge elements, see, e.g., [8].

Manuscript received March 9, 1987; revised July 12, 1990. Recommended for acceptance by C. Brown. This work was supported by the National Swedish Board for Technical Development (STU).

The authors are with the Computational Vision and Active Perception Laboratory, Royal Institute of Technology, S-100 44 Stockholm, Sweden.

IEEE Log Number 9040025.

Here we address problems appearing in the case that edges are extracted and traced, as in the work discussed, e.g., in Faugeras *et al.* [9]. Hence, we assume that there is a set of sampled contours from which we need to explicitly extract important shape information. Such structure may be straight parts of the boundaries, corners, parallelism, and symmetry. Information about curvature and direction is also important. Furthermore, surface recovery techniques may require parameterized boundaries. Finally, it is in many cases necessary to segment boundary contours into meaningful primitive parts.

Characteristic to these problems is that the data to be considered are planar curves (in the images). Moreover, these curves contain details at various levels of scale and are also contaminated by noise. There exists a need for making the information given by the curves explicitly available for further processing by deriving some abstraction of them. Such a description should be much simpler than the given representation (which is discrete) and should not be too much influenced by noise or irrelevant details in the data.

One systematic and mathematically well-founded way of finding such descriptions or simplified representations of curves and contours is to approximate them with some family of functions. We shall also propose an approach of that sort. However, there are two issues which in the standard literature on curve approximation are either not addressed or not given tractable solutions. First, most numerical techniques require that the critical points that describe the shape of the curve are given, at least as a subset of the breakpoints. If this is not the case, one might end up with hard numerical problems, e.g., in spline approximation with variable knots, which gives rise to nonlinear problems. Secondly, the methods for approximation give no hints on how the parameters should be set up to help us find the critical points. Of course, the definition of what constitutes a good description of a shape or a curve is application dependent, but certain shape features should be reliably recoverable over large parameter ranges. In particular, several of the computational tasks presented above, e.g., the check for parallelism or the computation of curvature and angles require both a smoothed and a precise description of the data. If the smoothing depends critically upon some unknown tolerance and scale parameters, the tasks become impossible without operator intervention.

In this correspondence we present an algorithm for computing shape descriptions, that addresses these issues. The method is based on multiscale approximations with lines. A crucial first step is the generation of a hierarchic family of polygonal approximations. From this *family* the descriptive shape properties are derived by analysis of the features that are stable over scale. The principle used is the principle of transformational invariance, suggesting that structures that remain invariant under a set of transformations (here: smoothings), have significance. This is related to Lowe's ideas about nonaccidentalness, [8], but has no probabilistic component. As a final postprocessing step we also show that the polygons can be converted into a spline-representation. Although visually pleasing, the latter representation does not add any significant information about the abstract shape in our present framework.

II. On Earlier Work and Our Approach

If one wants to derive descriptions of planar curves such that geometric properties of the type mentioned above are explicitly represented, one is faced with two goals of a conflicting nature. First there is a need for finding a qualitative description of overall shape. Hence some simplification and/or smoothing must take place. This goal is important for recognition and for finding global structure. Secondly, there is a need for high precision detection of certain characteristics. Earlier we mentioned straight line segments, corners, angles, parallelism, and symmetry and their projections from 3- to 2-D. Recent work on shape and image structures has been addressing these problems by investigating multiscale representations. Based on early work on multiscale descriptions, e.g., Rosenfeld and Thurston [10], Witkin [11], introduced the notion of scale-space representation. In this approach one considers a one-parameter family of representations of a shape, the parameter being scale. Witkin suggests that one first smoothes the signal with a mask of variable size and then looks for structures which are stable over scale. Particular emphasis is given the problem of tracking the structures as the scale varies from coarse to fine. Later this problem has been given more precise solutions for image contours [12], [13]. Witkin [11] looked at waveforms but the approach applies to planar contours in general. Mokhtarian and Mackworth [14], use Gaussian smoothing in this case. However, they smoothed the coordinate functions separately, which led to some unreasonable effects. In [15] they tried to avoid these by renormalizing the curvature. However, the Gaussian smoothing has the effect of shrinking closed convex curves. Lowe [16] suggested a compensational mechanism for this problem.

An important aspect of this approach is that the smoothing is indiscriminate: one applies the smoothing independently of the structure of the original data. As a consequence important geometric features like corners and intersecting lines—indicating 3-D structure like occlusion or parallelism and collinearity—are represented only implicitly. At the coarser levels they occur in blurred form and hence lack the characteristic properties. At the finer levels, the geometry is more precise but the structure is hard to find since it is represented with a lot of fine details and noise. The precise description of a corner, a junction or a linear part of the boundary may never exist at any level of scale. The fact that one can filter out stable representations over scale (Witkin [11]) does not solve the problem. We need to remove unnecessary detail without destroying localization accuracy.

An attempt to attain this goal using a diffusion process is made in Kimia [17]; see also Kimia *et al.* [18]. Without contending that these general smoothing approaches cannot be applied to compute shape descriptions, we have taken an alternative approach to the problem which preserves geometric precision in a straightforward manner and derives explicit descriptions in terms of a set of shape descriptions. The approach is based on multiscale approximations with polygons. Mechanisms for detecting significant geometric events, like corners, changes from straight to curved segments, etc., are built into the procedure. In fact, our approach in that respect has some kinship with the method proposed by Fischler and Bolles [19]. They address a different problem, but also try to detect the different processes that may account for variations along a contour. However, considerations about fine details, noise, and reconstruction of geometric properties is outside the scope of their paper.

To summarize, the purpose of this work is not just to smooth the curve, but rather to derive a description of it in terms of a set of primitives. Furthermore, this description should capture the (intuitive) shape features of the curve, which are assumed to be apparent at the certain (unknown) scales. These scales are those at which the description remains stable as scale as varied, as proposed by Witkin [11]. Our approach here to this problem uses polygonal approximation. The descriptive primitives are qualitative: straight arcs, curved arcs with sign of curvature, corners (tangent discontinuities), and points delimiting the arcs, especially inflexion points. Beside these primitives we also derive the *tangents* at the obtained points, i.e., inflexions, corners (two tangents) and transitions from straight to curved.

Lindeberg and Weiss [20] argued that a well-behaved shape preserving curve smoothing scheme should have the following properties.

- Fine scale features should disappear before coarse scale features.
- A circle should not shrink (too much).
- A jagged curve should not grow or shrink (too much).
- A simple curve should remain simple.
- A closed curve should remain closed.
- The number of inflexion points should not increase, implying that no new singularities should be introduced in the Gauss map.
- The number of curvature extrema should not increase.

Our method satisfies all these properties except that there is no control over the curvature extrema. Empirically, it seems to fulfill also this condition.

We shall next describe this method in the following steps. First we shall present our basic polygonal approximation method. Using this algorithm we next show how the hierarchic *family* of polygon approximations is created. Finally we demonstrate how the descriptive primitives can be derived from this family. Some examples and an indication of how a spline-approximation can be performed as a postprocessing step complete our correspondence.

III. An Algorithm for Coarse-to-Fine Polygon Approximation

We shall in this section briefly describe a polygonal approximation algorithm which depends on only one variable and which gives coarse-to-fine representations of a contour, as this variable is varied. The algorithm is derived in [21] by a careful consideration of different design criteria proposed in the literature and how they meet the goals of this work. Before we find the multiscale approximations, we first need a suitable one level algorithm. To this end we consider some existing approaches and keep in mind that our aim is to represent image contours.

A. Motivations

Numerous techniques for fitting polygons to curves have been suggested. Pure interpolation techniques dominate, but true approximation schemes have also been proposed. In [21] we made an argument for the use of an interpolation scheme, primarily on the basis that curves obtained by edge detection followed by contour tracing will be sampled densely and rather uniformly in arc length.

In [21] there is a detailed analysis of the requirements on a polygon approximation algorithm in general and on one that could be used to create a multiscale shape representation in particular. The main conclusions of the analysis can be stated as follows.

- A split-and-merge technique can be used.
- Collinearity tests should be based on *both* the maximum error e and on the ratio of the absolute value of the accumulated signed area the curve defines with respect to the line and the length of the line $|A/L|$.
- The maximum error tests can be used in the split step, the test on signed area over length in the merge step, and multiples of the same threshold can be used; see below for an explanation.
- For efficiency reasons, the step size in the split step should be adapted so that it accounts for the variability of the data as well as possible. In principle, one starts with a large step, decreases it when the collinearity test fails (splits), increases it when the test succeeds; see [21] for details.

B. A Proposed Algorithm for Polygonal Approximation

As a consequence of these arguments, an algorithm embodying the features proposed above has been developed. This algorithm is of the split-and-merge type. It can in short be described as follows.

1) The collinearity tests are of the form $e \leq t$. The error e is the distance to the line segment and not the distance to the line. Modifications of t depending on the relative number of sign changes are optional.
2) The merge tests are of the form $|A/L| \leq t/2$ and the area is accumulated in case of successive merges. The same threshold t is used, implying conservative merges. See the next section.
3) The step size is adaptively modified.

Two things can be noted about this algorithm:

- It checks *both* the maximum error e and the signed swept area A. The arguments for these tests have been given earlier.
- A multiple of the same threshold t is used in the split and in the merge tests. This may not be ideal in a one-step algorithm, but

works well in our multiscale approach. For curve segments that project orthogonally onto the line segment a merge will indeed be performed if $e < t$, in particular if e is very small. Moreover, the use of $t/2$ ensures that if a merge is performed in the convex case, then $e \leq t$. (In fact, we have also used the threshold t with good results, even though this may lead to merges also when $e > t$).

This concludes our description of the principles of the algorithm used at one specific scale level. The exact algorithm depends to a certain degree on the earlier levels. We shall next describe this.

C. The Multiscale Algorithm

The algorithm just outlined is well suited for building up a hierarchical family of polygon approximations that exhibits the scale-space property of causality and of not introducing any new structure. Witkin [11] noted that the fingerprints, the contours tracing out the occurring inflexion points over scale, could serve as a basis for description over all scales. Two assumptions were made as follows.

1) *Identity:* Two extrema at different scales but on the same zero-contour (of $F_{xx}^{(\sigma)}$ in 1-D, $F^{(\sigma)}$ being the signal blurred with a Gaussian with s.d. σ) in scale space emanate from a single underlying event.
2) *Localization:* The true localization of an event (zero-contour) is the localization obtained as $\sigma \to 0$.

In our case we apply these assumptions at each approximation level, that is we perform the tracking as we build up the set of descriptions at varying scale.

In principle, we proceed as follows. First all the necessary splits are performed, so that segments satisfying the collinearity criterion are found. After each successful split operation, identification and localization are made by a search in scale-space. New breakpoints, not appearing at finer scales, can occur but are then inserted also at the finer levels. When the contour is processed in its entirety in this way, a merge step is performed to give a representation at the current scale level. We note that the mentioned insertion of new points may add some breakpoints to this representation in the final result.

More precisely, the algorithm runs in the following way.

Given a discrete contour $P = \{p_i \mid i = 1, \cdots, N\}$, the scale levels are defined by

$$\epsilon_s = s \cdot \Delta\epsilon, \qquad s = 0, \cdots S.$$

The approximating polygon at level s is $P^{(s)} = \{p_i^{(s)} \mid i = 1, \cdots N^{(s)}\}, s = 0, \cdots, S$. In particular, we identify $P^{(0)}$ with P.

Since the algorithm uses interpolation, $P^{(s)} \subseteq P^{(0)}$ for all s. Therefore, $P^{(0)}$ induces an order on all the points $\{p_i \mid i = 1, \cdots, N\}$. We shall use the terms preceding and succeeding points with reference to this order.

We assume that at any given step of the algorithm described below the step size is well-defined. It can, for example, be given as a fixed output value or adaptively computed. For simplicity we shall omit the specification of the step size in the description of the different parts of the algorithm.

In the algorithm at level s we first perform a split step, computing a polygon

$$Q^{(s)} = \{q_i^{(s)} \mid i = 1, \cdots, N^{(s)}\}.$$

From this polygon $P^{(s)}$ is computed by a merge step. The procedure is the following.

Assume that the points $\{q_i^{(s)} \mid i = 1, \cdots, I\}$ have been computed. The error is $\epsilon^{(s)}$ and the step size is given. Let $p_i = q_I^{(s)}$ be the left endpoint of our next line segment.

1) Use the given step size to find a new right endpoint p_r.
2) Test for collinearity between p_i and p_r, error $\epsilon^{(s)}$.
3) If 2) fails, do a split and find a new right endpoint p_r. Go to 2).
4) If 2) succeeds, find the rightmost point in $T_r^{(s)} = \{p_i^{(t)} \mid 0 < t < s, p_i^{(t)}$ succeeds $p_r\}$ that satisfies the collinearity test together with p_i and with error $\epsilon^{(s)}$. If there is such a point define it to be $q_{I+1}^{(s)}$ and to to 7).
5) If 4) fails, do an analogous test for $T_i^{(s)} = \{p_i^{(t)} \mid 0 < t < s, p_i^{(t)}$ succeeds p_i and precedes $p_r\}$. If a point is found go to 7).
6) If 5) also fails, define $q_{I+1}^{(s)}$ as p_r.
7) Insert $q_{I+1}^{(s)}$ as a point in $P^{(t)}, t < s$, unless it is already included.
8) Compute the signed area $a_I^{(s)}$ of the triangle $q_{I-1}^{(s)}q_I^{(s)}q_{I+1}^{(s)}$ and save $|a_I^{(s)}/l_I^{(s)}|$ if it is $< \epsilon^{(s)}$, where $l_I^{(s)} = d(q_{I-1}^{(s)}, q_{I+1}^{(s)})$.
9) Given $Q^{(s)}$, perform a merge test using the parameter $|A/l|$ introduced above. This results in $P^{(s)}$, a (tentative) polygon approximation at level s.

The *tentativeness* depends on the fact that 7) can modify $P^{(s)}$ later.

A remark can be made about the implementation of 9). For efficiency the signed area, $a_I^{(s)}$, of the triangle $q_{I-1}^{(s)}q_I^{(s)}q_{I+1}^{(s)}$ is computed upon the localization of q_{I+1}. The point $q_I^{(s)}$ is then marked for a possible merge if $|a_I^{(s)}/l_I^{(s)}| \leq \epsilon^{(s)}$. The marked points are ordered with respect to the increasing values of the test parameter. The merge test is then performed in this order. The test parameters are recomputed as points are removed so that they represent the appropriate values. This procedure may imply that merges that could have been made at level s (on the basis of the error criterion) are never considered. However, this will be remedied at higher levels.

We have now presented an algorithm for multiscale polygon fitting which produces the polygons $\{P^{(s)} \mid s = 0, \cdots, S\}$. This set of polygons form a coarse-to-fine representation of the input contour and gives as such a description of its shape. If we want to draw conclusions about the shape like those listed in the introduction, we need to filter out the important features of the description. Of particular interest are the descriptions which are maximally stable over scale. We shall call it the scale-invariant description in the interval $[0, S]$. In a sense this shape description is obtained without resort to any arbitrarily chosen parameters. In [22] we gave examples of such descriptions.

However, there are certain limitations to such scale-invariant descriptions. If the contour is a noisy version of a polygon with its dominating structures at roughly one scale level, then the scale-invariant description is good. If, on the other hand, different structures occur at different scales and, moreover, if the contour is curved, then the stable as well as the unstable parts of the scale-space representation are indicative of the structure of the contour. We shall now show how the *family* of polygon approximations over scale can be used to derive a shape preserving representation.

IV. Deriving the Stable Shape Features

In this section we will present the method for deriving the stable descriptions of the curves, or rather the features contained in these descriptions, on the basis of the family of polygons found by the technique described in the previous section.

Hence, we assume that we initially have a curve, given as an ordered set of points in the plane, $P_0 = \{p_i \mid i = 1, \cdots, N\}, p_i = (x_i, y_i)$, where N is a fairly large number, typically $100 \leq N \leq 10\,000$. From P_0 and a tolerance step ϵ, which could be the grid resolution, we derive a set of polygon approximations,[1] $\{P_i \mid i = 1, \cdots, S\}$, where $P_0 \supseteq P_1 \supseteq \cdots \supseteq P_{S-1} \supseteq P_S$, and P_k is a polygon within tolerance $k\epsilon$ from P_0. We will also show how for each P_k the family of polygons can be used to construct a polygon Q_k with the same level of detail as P_k, but more similar in shape to the contour at hand.

In fact, the polygons Q_k are in [23] used to produce spline approximations. However, the goal here is to derive descriptive shape features that are stable over scale, not just a perceptually pleasing approximation. Hence we proceed by, from polygon Q_k, characterizing the points and segments of the curve as corners,

[1] In the preceding section $P^{(s)}$ denoted a polygon at level s. From now on P_s denotes a polygon at level s.

straight segments and curved segments. The curved segments are divided at points of inflexion into arcs of positive and negative curvature. Finally, the tangents at inflexion points and corners are estimated. In this computation all the data in P_0 are used. The procedure can be summarized in the following steps.

1) Choose a scale, i.e., an integer value k, that is stable.
2) At the chosen scale k, find corner points and segment the contour into straight curved parts.
3) Find the positions of the inflexion points of the curved parts of the contour.
4) The tangent values at corners and inflexion points are estimated.

We will next describe these steps.

A. Detecting Stable Scales

In his work on scale space, Witkin [11] suggested that those features of the signals he studied that tended to leap out at the eye were such features that showed stability over scale, i.e. the features existed over a broader scale interval than other features (at the same locations). Scale in our case corresponds to the number k, as polygon P_k is computed with tolerance $k\epsilon$. In line with the ideas of Witkin, and as a consequence of the argument made in Section I, we consider the number of inflexion points of the polygons from the family versus scale, that is the tolerance $k\epsilon$. An inflexion point of a polygon is defined as in Fig. 1. A stable scale is a plateau of this function; see Fig. 2 for an example. Note that these plateaus can be ranked using their *lifelength* (width) and that no threshold actually is needed. In practice we are however only interested in the most prominent stable scales, and therefore a threshold is used. Possibly the plateaus are not completely flat, which could imply the use of a second threshold. However, this has not been required in the examples given here.

B. Corners, Straight and Curved Parts

Let k denote a number where $k\epsilon$ belongs to a stable tolerance interval. At this scale, segmentation of the contour into straight and curved parts and identification of corner points will be made. Essentially, straight parts of the contour will be represented with lines in the polygon that are longer than those lines that constitute the curved parts. Furthermore, polygon segments in parts of the contour that are mainly straight at level k will be more stable over scale with respect to the number of points than will the lines of the curved parts. These observations are the basis for the segmentation procedure which goes as follows. As in [21] we start with the polygon P_k and from this we make a new polygon Q_k by descending hierarchy of polygons d levels and replacing the representation of a segment with its lower level representation as long as no inflexion points are added to the contour. The polygon so constructed will essentially represent the same shape as P_k does but the properties of the line lengths will be more emphasized in Q_k.

The lengths of all segments are computed and checked in the following way. Segments longer than $Ck\epsilon$ are labeled as straight. Let l_1 and l_2 be the lengths of two consecutive segments. Additional straight parts are defined as the longer of l_1 and l_2 where $l_1/l_2 < r$ or $l_1/l_2 > 1/r$. The values of C and r used in our examples below are $C = 8$ and $r = 5$. Points separating straight parts are corner points. Also points with sharp vertex angles, typically $v < 70°$, are labeled corner points.

The corner points and the points where our representation changes from straight to curved are called break points. The position of a corner point is obvious but the position of a breakpoint between straight and curved segments is not. However, we have chosen to use a point on the long polygon segment in this way: let l_1 and l_2 be the lengths of the longer and shorter segments, respectively. The breakpoint is chosen as the point on the long segment located a distance $l_2/2$ from the vertex point, thus allowing the curved part to extend a little into the straight segment. This is incidentally useful if one wants to produce a smooth looking approximation later on.

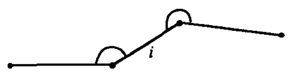

Fig. 1. An inner segment of the polygon is an inflexion segment (i) if the vertex angles at the segments endpoints are on different sides of 180°.

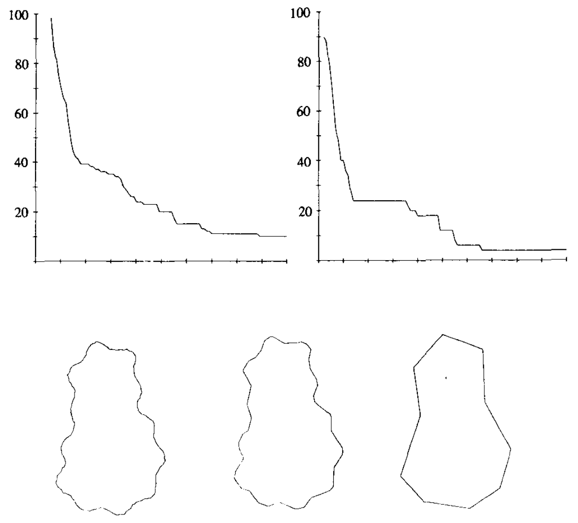

Fig. 2. The leftmost contour consists of about 200 points and it contains some noise. A family of polygons with this contour as P_0 is constructed. The number of points of these polygons versus scale is shown in the left graph and the number of inflexions versus scale in the right graph. In the latter, the two widest plateaus are easily seen. They correspond to the two most stable polygons with 24 and 4 inflexions, respectively (middle and rightmost contours).

C. Positions of Inflexion Points

In Section IV-B we showed how to find the positions of two kinds of breakpoints, corners, and points where straight and curved parts join. A third kind of points that we wish to locate are the points of inflexion.

The simplest way to estimate the position of an inflexion point from a polygon is to use the coordinates of the midpoint of the inflexion segment (Fig. 1). If we use this simple method on a polygon Q_k constructed as described earlier, using points from $P_k, P_{k-1}, \cdots, P_{k-d}$, we will in most cases get a more accurate position than we get from P_k. (If the inflexion segment is short, the position of the point is known with better accuracy than if the segment is long.) In some cases, however, the inflexion segment will be the same or almost the same in Q_k and P_k. Then, an alternative procedure is applied. We compute the positions of the inflexion points at a number of scales, according to the simple rule above. Then, we match these points over scale in consistent way, in order to connect the inflexions of P_k to inflexions of P_{k-d}, which will be better localized. The idea behind this procedure is that features observed at a coarse scale always have their origin at a finer scale (cf. the identity assumption of Section III-C used for the polygon approximation scheme). The following three observations are used in the matching procedure.

1) As we move from fine to coarser scale (increasing k), the level of detail must decrease and structure must not be created. Globally, this means that the total number of inflexion points of the polygon will decrease. Locally, it means that two inflexion points at a coarser scale cannot be matched to a single point at finer scale.
2) When we match inflexion points from polygons P_k and P_{k+1}, some points of P_k will remain unmatched if the number of

inflexion points is larger in P_k than in P_{k+1}. We say that inflexion points *disappear* when we move from fine to coarse scale. For closed polygons, the disappearing inflexion points will always be neighboring pairs.

3) For open curves, the inflexion points closest to the ends of the curve may disappear as singletons while inner points behave as if they belonged to a closed curve.

Let the finest scale used be $k_f = k - d$. Start with this scale and the nearest coarser, $k_f + 1$. First we find exact *matches* of inflexion points between the two scales. For an exact match we demand that the endpoints of the inflexion segments shall be identical at the two levels. Then, we try to match the lists of inflexion points from the two levels in the regions between the exactly matching points. For each inflexion point i at scale $k_f + 1$ that has not been matched exactly, we set up a list of *matching candidates* from level k_f. This list consists of two inflexion points at level k_f that are closest in arc length to i followed by the two points that are closest in Euclidean distance (often, these points overlap, so the candidate list has only two or three points). We create an initial match by combining each inflexion point at level $k_f + 1$ with the first point in its candidate list. If this matching is consistent with the conditions 1)–3) we accept this matching and proceed to match inflexion points from levels $k_f + 1$ and $k_f + 2$. If the matching is not consistent in some region, we will try the rest of the points from the candidate list for each inflexion at level $k_f + 1$. It is done in a simple and systematic way, trying each possible combination of matches from the candidate lists and trying to use points from the beginning of the list first. Three problems can be foreseen with this procedure. First, the naive approach to try all possible combinations could give rise to a *combinatorial explosion* and thus be very time consuming. The second problem is that we may be unable to find a solution with the points from the candidate lists. In our examples, the first problem does not occur because we use such a small tolerance step ϵ, implying that polygons at consecutive scales are close. This means that they have many points in common which leads to a large number of exact matches and few inflexion points in the lists we have to match. Regarding the second problem, the solution of that would simply be to use longer candidate lists. This problem has not been encountered in our examples. A third problem that we have not considered is that the solution we get may not be unique. We just accept the first match that is consistent with our rules and we do not know if there are other consistent solutions.

D. Estimation of Tangents

Given a point at or close to a noisy contour, we wish to find a tangent value that approximates the tangent of the contour near or at the point. The polygon approximations we work with are in a sense coarse approximations of tangents to the noisy curve, but although we have good polygon approximations, the lines of the polygons are not good as tangents to the contour at the breakpoints. A very simple way of computing a tangent at a point is to fit a straight line to data in a neighborhood of the point, e.g., by using the method of least squares. However, the best fitting straight line will in many cases not have the slope that one would intuitively think is the right tangent. We believe that higher order approximations are better suited to find correct tangent values. Below, we show the three situations, where we fit tangents to data and then we will describe the procedure used in each case.

Case 1: Corners.

Here, we have different tangents on the different sides of the corner point. When the right (left) tangent is computed we will only use data to the right (left) of the corner. A straight line fit is not sufficient to catch the slope of a curve near a corner point as seen in Fig. 3.

If very few data close to the interesting point were used to fit the line, the result might be better, but it would be highly dependent on the amount of noise in P_0. As the noise level is expected to be large in most cases, this method will be very unstable.

Case 2: Inflexion points.

In this case, we will use data on both sides of the point at which we wish to find the tangent. Also in this case, fitting of a straight line

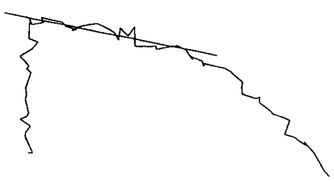
Fig. 3. Least squares fit of a straight line to part of a noisy curve. Although the curve is approximated reasonably well by the line, the tangent of the corner point is not. See Fig. 7 for a better result.

is not good enough. Reasons are the same as in Case 1.

Case 3: Smooth breakpoints without change in the sign of curvature.

In this case, a straight line approximation may give a good tangent value.

In Cases 1 and 3 we have data representing a curve with a constant sign of curvature, i.e., no inflexion points are present (at the level of scale we have chosen). In these cases we fit second order polynomials to data and take the tangents of these polynomials at or near the breakpoints as approximate tangents to the contour.

In Case 2, second order polynomials are not suitable. Here we know that data represent a contour with an inflexion point. As second order polynomials have no inflexion points we will not use them in this case but try with polynomials of degree 3 instead.

Note that the approximations by second and third order polynomials are used only for tangent estimation, not for curve representation.

Approximation Method: The data we wish to approximate with polynomials are a number of points in the *xy*-plane: $\{(x_i, y_i);\ i = 0, 1, \cdots, m\}$. However, we will not try to fit y as a function of x (or vice versa) as we cannot expect our data to behave like such a function and we desire an approximation method that is independent of how the contour is rotated in the *xy*-plane.

The way to solve this problem is to introduce a parameter t, and let x and y depend on this parameter. For each point (x_i, y_i) a value t_i is chosen in such a way that t_i increases monotonically with i. Then functions can be fitted to $\{(x_i, t_i);\ i = 0, 1, \cdots, m\}$ and $\{(y_i, t_i);\ i = 0, 1, \cdots, m\}$ independently by the ordinary method of least squares. These functions will be independent of how the data are rotated in the *xy*-plane in the sense that if all data points $X_i = (x_i, y_i)^T$ are rotated by a standard 2 by 2 rotation matrix R, i.e., each X is replaced with $X' = RX$, then this corresponds to the same rotation of the approximating functions: $X_R^*(t) = RX^*(t)$. $X^*(t) = (x^*(t), y^*(t))^T$ is the solution obtained when we fit our set of basis functions to the original non-rotated data. $X_R^*(t)$ is the solution to the same problem but using data rotated with matrix R. A proof of this can be found in Bengtsson [23].

The simplest way to set the values of the parameter t is to let it vary equidistantly, i.e., let $t_i = i$ or $t_i = i/m$ where $i = 0, 1, \cdots, m$. However, it has been shown in experiments (de Boor [24]) that if distances between consecutive data points vary much along the curve, this kind of parameterization can give rise to unpleasant and quite unexpected features, such as loops, in the approximating curve. To avoid such problems, we let the values of the parameter t depend on the distances between data points, the arc lengths in our raw data P_0.

Let

$$d_0 = 0.$$

$$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \qquad i = 1, 2, \cdots, m.$$

$$D = \sum_{i=1}^{m} d_i.$$

$$t_0 = 0.$$

$$t_i = \frac{1}{D} \sum_{j=1}^{i} d_j, \qquad i = 1, 2, \cdots, m.$$

Now, we have $\{(x_i, t_i), (y_i, t_i);\ i = 0, 1, \cdots, m\}$, where the parameter t ranges from $t_0 = 0.0$ to $t_m = 1.0$ and small steps in

t mean that data points are close. Polynomials of order 2 or 3, the order depending on which of the cases we have, will be fitted to $x(t)$ and $y(t)$ independently. The method of least squares is used in a standard way, with *QR*-factorization of the coefficient matrix.

Two slightly different polynomial expressions have been tested in the second order case as well as in the third order case. Let t_c denote the parameter value of the point where we seek the tangent. The dot above x and y stands for derivative with respect to t.

In the second order case, the first type of approach is to assume that we know that the polynomial we fit must pass through the point $(x_c, y_c) = (x(t_c), y(t_c))$. The second approach is to make no such assumption, we just wish to find the tangent in a neighborhood of (x_c, y_c). With the first approach we fit the two parameters in each of the polynomials

$$x^*(t) = x_c + a_1(t - t_c) + a_2(t - t_c)^2$$

and

$$y^*(t) = y_c + b_1(t - t_c) + b_2(t - t_c)^2$$

and with the second approach, three parameters per polynomial are needed. We fit

$$x^*(t) = a_0 + a_1 t + a_2 t^2$$

and

$$y^*(t) = b_0 + b_1 t + b_2 t^2.$$

The results of the two approaches were essentially the same in the sense that the shapes of the approximating curves based on the tangents computed from these polynomials were very similar.

In the third order case, the first approach is to require the polynomial to pass through the point (x_c, y_c) and to have zero curvature at this point. The last requirement leads to a minimization problem with nonlinear constraints. To avoid this we made a simplification. Essentially, zero curvature at t_c is equivalent to $\dot{x}\ddot{y} - \dot{y}\ddot{x} = 0$ at t_c. We require $\ddot{x}(t_c) = \ddot{y}(t_c) = 0$ which is not absolutely necessary for zero curvature but gives the ordinary and simple kind of minimization problem. The polynomials to fit in this case are

$$x^*(t) = x_c + a_1(t - t_c) + a_3(t - t_c)^3$$

and

$$y^*(t) = y_c + b_1(t - t_c) + b_3(t - t_c)^3.$$

The second kind of approach is to fit the third order polynomials without constraints and compute the tangent close to (x_c, y_c).

$$x^*(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

and

$$y^*(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3.$$

In the first approach we compute two parameters for each polynomial and with the second approach four parameters per polynomial are necessary. We found that the two-parameter method was more robust with respect to changes in the number of points used in the fitting procedure and therefore more reliable.

When the polynomials $(x^*(t), y^*(t))$ are computed, we take the derivatives of these polynomials at the point $t = t_c$, $(\dot{x}^*(t_c), \dot{y}^*(t_c))^T$, as an estimate of the tangent, see [23]. We will only use the direction of this vector, not its absolute value.

To perform the computations we have to choose the points to use in the least square fit. The number of points should in some way depend on the chosen scale. The actual dependency is not obvious but it is reasonable that on a fine scale we will use fewer points (or at least a smaller part of the contour) than on a coarse scale. We will assume that our possible scales will be above the noise level and as we stated earlier, in the presence of noise, we must not use too few points. The absolute minimum is 2–4 points (depending on which case we have), meaning interpolation by the chosen kind of polynomial. The maximum is set by the following argument. When approximating the tangent in one breakpoint, we will not use data points beyond any adjacent breakpoint (at the current scale). This is reasonable because a breakpoint indicates a significant shape-change and a specific tangent value should only depend on the two curved shape elements adjacent to it (one in the case of a corner tangent). In our experiments, we let the algorithm start with between 10 and 20 points and reduce this number, if necessary, according to the argument above. It is also reasonable that the total curvature of the contour corresponding to the chosen $m + 1$ points is not too large.

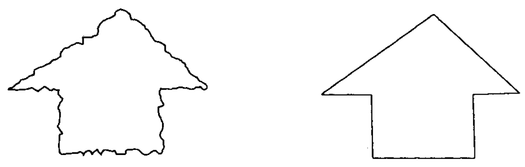
Fig. 4. A noisy polygonal contour and the corresponding most stable polygon.

When corner tangents are computed, the $m + 1$ points are chosen from just one side of the corner point. In the other two smooth cases, we try to choose the points symmetrically around the breakpoint.

E. Summary

The four previous subsections described the steps of the procedure outlined at the beginning of Section IV. We will next demonstrate the performance of the entire method, including the multiscale polygon approximation, on some examples.

V. Results

In Fig. 2 the detection of stable scales is illustrated. It is notable that the obtained polygons represent the shape reasonably well in this case where the noise level is low. If we take a noisy but simple polygonal shape, like the arrow in Fig. 4, then the most stable representation will be a scale invariant polygon for a very large scale interval. In fact, from the noisy arrow, containing 696 points on a 1000 × 1000 grid, the 7 arcs and corners are obtained if $S = k_{\max}$ is allowed to be over 200. More interesting is, of course, the case when the shape contains straight *and* curved arcs. This is the case in Fig. 5(a), containing an image of the letter B, represented as three curves on a grid of 1000 × 1000 points. The width of the letter is about 60ϵ (ϵ = the grid increment) and the corresponding scale interval is considered. In this interval there is one stable scale. Two straight parts are at this scale the left vertical part of the entire letter and of the upper hole. The lower hole is not found to be straight. The tangents found at the corners are indicated in the figures, as well as some other estimated tangents. These are used in an algorithm for fitting conic splines, using the so-called guided form for conics; see Pavlidis [25] and Bengtsson [23]. We will not describe this method here, since it does not refine the qualitative description we are deriving here. However, the output of this algorithm, which is seen in Fig. 5(c), illustrates the classification into straight and curved segments.

It should be noted that in Figs. 5–8, the estimated tangents are drawn symmetrically around their corresponding breakpoint.

Fig. 6 shows a noisy pear-like figure with some estimated tangents, especially at the inflexions, at the most stable scale. Again a postprocessed spline version is also shown.

Another example is shown in Fig. 7. Here we especially see the result obtained by application of the tangent estimation technique in Fig. 1, as opposed to a direct fit as shown in Fig. 3. In this case as well as for the regularly patterned figure in Fig. 8, we obtain two straight and one curved arc, and this description again occurs at the most stable scale.

In all these examples there is one outstanding stable scale. That is not the case in Fig. 9(a), taken from Fischler and Bolles [19] but with positional noise added. Again we use our spline approximation

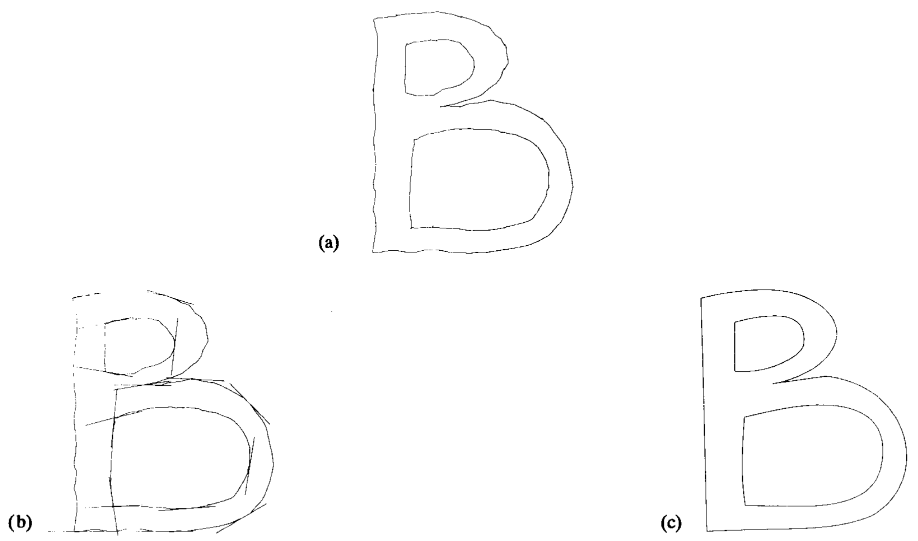

Fig. 5. (a) Original noisy contour. (b) Estimated tangents. No tangents are indicated at the straight segments. First, tangents at corners and inflexion points (none present here) are computed. Later, during the curve fitting procedure, additional tangents are computed. All these tangents are shown in the figure. (c) Conic spline representation illustrating the labeling.

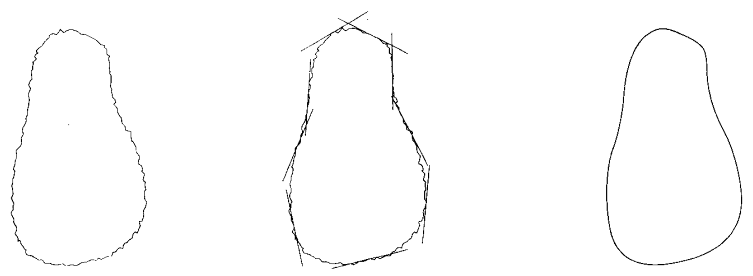

Fig. 6. A noisy pear-like shape. The only stable polygon has four inflexions. The tangents at the inflexion points are drawn along with the additional tangents required by the spline procedure. The rightmost shape shows the final approximation.

algorithm to illustrate which arcs are labeled as straight and curved [see Fig. 9(b)] when the most stable scale up to a level of 40ϵ is considered (the original contains 496 points). (In this early experiment the stability was defined by the number of points versus scale.) It can be seen that the final output is intuitively reasonable. The right part of the wavelike pattern is labelled as built up by straight part, but this is plausible from the noisy original. For a less noisy version, Fig. 9(c), we obtain an intuitively correct result, Fig. 9(d).

VI. Summary and Discussion

We have presented an approach for deriving qualitative descriptions of contours that contain structures at different (unknown) scales. The descriptions are in terms of straight arcs, curved arcs with sign of curvature, corners and points delimiting the arcs: inflexion points and transitions from straight to curved. Besides this, the tangents at these points are also derived (one-sided in the appropriate cases).

The method is based on the derivation of a multiscale, hierarchic family of polygon approximations, which satisfies intuitive criteria for curve smoothing, like those suggested by Lindeberg and Weiss [20]. It uses stability over scale as a criterion for selecting significant structures, as is prescribed by scale-space theory. The *family* of polygons allows a classification of the arcs and localization of inflexions and a polynomial approximation step is used to derive tangents.

Examples with noisy line drawings show that the method allows computation of intuitive shape features robustly, at least if the resolution in the sampling of the curve is sufficient.

A number of problems are however yet not investigated. For instance, we have not studied if scale should be transformed to allow proper comparisons over scale. In Lindeberg and Eklundh [26] it is shown that for the scale-space embeddings according to the diffusion equation such transformations are needed. In our experiment they seem to be superfluous, since mainly only two scales occur globally. This indicates, in fact, that future experiments should be performed on true edge data from realistic images containing structures at many different salient scales. We are also working on such problems. However, the problem of properly tracing and segmenting the contours must then also be solved. Our current belief is that these problems should be solved in a closed feedback loop and not in a one-pass procedure, as is commonly done. Hence, the segmentation and description steps should be coupled through feedback, in analogy to any segmentation and interpretation process. Our current work is pursuing this approach and we think that the applicability of our approach to difficult realistic imagery can only be verified in such a framework.

In spite of these open questions we believe that our method is robust and gives appropriate results. Hence, it can serve as a preprocessing to other processes for computing geometric structure and groupings, such as, e.g., proposed by Ulupinar and Nevatia [27], or for an approximation with some set of functions that parameterizes

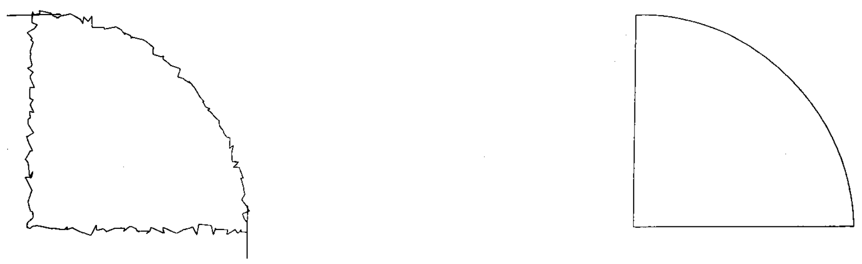

Fig. 7. For this jagged quarter circle, the only stable shape is the one with two straight parts and one curved part. The estimated tangents are drawn directly on the original noisy contour to the left and to the right is the final approximation.

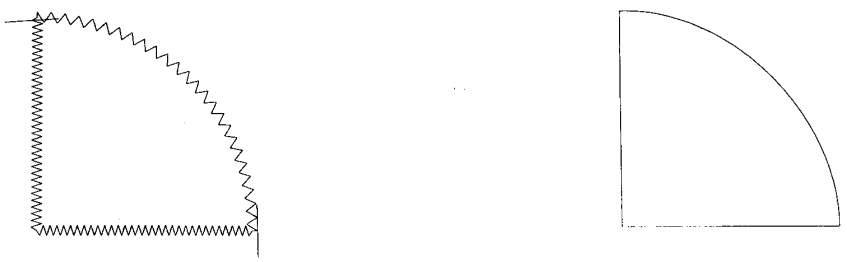

Fig. 8. A slightly different version of the same pattern. Again good tangent estimates are found. We note that our algorithm has not chosen the corner breakpoints symmetrically. The top-left corner is on the inside of the curved part and the bottom-right corner is on the outside.

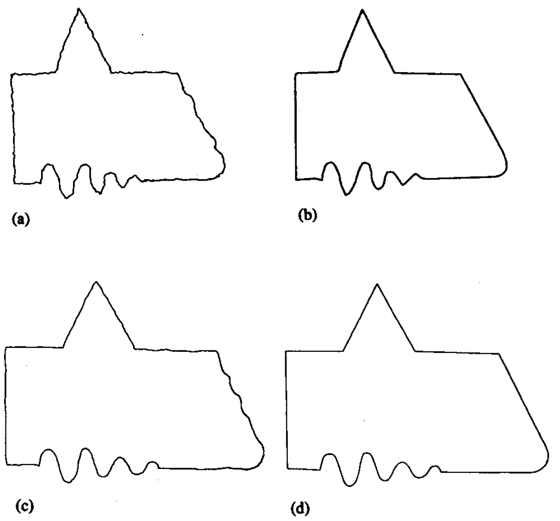

Fig. 9. (a) The figure from Fischler and Bolles [19], noisy version. (b) Final result, the most stable representation up to 40ε, using conic splines for the curved arcs. (c) Less noisy original. (d) Final result with (c) as input.

the contour in a suitable way.

Acknowledgment

We would like to thank J. Howako, who contributed substantially to the early parts of this work. Finally, we thank T. Lindeberg for his valuable suggestions during continuous discussions.

References

[1] H. G. Barrow and J. M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artificial Intell.*, vol. 17, pp. 75–116, 1981.

[2] T. Kanade, "Geometrical aspects of interpreting images as a three-dimensional scene," *Proc. IEEE*, vol. 71, pp. 789–802, 1983.

[3] D. G. Lowe and T. O. Binford, "The recovery of three-dimensional structure from image curves," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-7, pp. 320–326, 1985.

[4] H. H. Baker and T. O. Binford, "Depth from edge and intensity based stereo," in *Proc. 7th IJCAI*, Vancouver, B.C., Canada, 1981, pp. 631–636.

[5] J. K. Aggarwal and A. Mitiche, "Structure and motion from images," in *Proc. Darpa Image Understanding Workshop*, 1985, pp. 89–98.

[6] H. H. Bülthoff and H. A. Mallot, "Interaction of different modules in depth perception," in *Proc. 1st ICCV*, London, 1987, pp. 295–305.

[7] S. W. Zucker, C. David, L. Dobbins, and L. Iverson, "The organization of curve detection: Coarse tangent fields and fine spline coverings," in *Proc. 2nd ICCV*, Tarpon Springs, FL, 1988, pp. 568–577.

[8] D. G. Lowe, *Perceptual Organization and Visual Recognition.* Boston, MA: Kluwer Academic, 1985.

[9] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot," in *Proc. Int. Symp. Robotics Research*, Santa Cruz, CA, 1987.

[10] A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Trans. Comput.*, vol. C-20, pp. 562–569, 1971.

[11] A. P. Witkin, "Scale-space filtering," in *Proc. IJCAI-83*, Karlsruhe, West Germany, 1983, pp. 1019–1022.

[12] J. Koenderink, "The structure of images," *Biol. Cybern.*, vol. 50, pp. 363–370, 1984.

[13] A. L. Yuille and T. Poggio, "Scaling theorems for zero-crossings," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 15–25, 1986.

[14] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 34–43, 1986.

[15] A. K. Mackworth and F. Mokhtarian, "The renormalized curvature scale-space and the evolution properties of planar curves," in *Proc. CVPR*, Ann Arbor, MI, 1988, pp. 298–303.

[16] D. G. Lowe, "Organization of smooth image curves at multiple scales," in *Proc. 2nd ICCV*, Tarpon Springs, FL, 1988, pp. 558–567.

[17] B. B. Kimia, "Conservation laws and a theory of shape," Ph.D. dissertation, McGill Univ., Montreal, P.Q., Canada, 1989.

[18] B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "Towards a computational theory of shape," in *Proc. 1st ECCV*, Antibes, 1990, pp. 402–407.

[19] M. A. Fischler and R. C. Bolles, "Perceptual organization and the curve partitioning problem," in *Proc. IJCAI-83*, Karlsruhe, West Germany, 1983, pp. 1014–1018.

[20] T. Lindeberg and R. Weiss, "Some thoughts about curves, surfaces, and singularity theory," CVAP, Royal Inst. Technol., Stockholm, Sweden, Rep. TN-04, 1990.

[21] A. Bengtsson, J. O. Eklundh, and J. Howako, "Shape representation by multiscale contour approximation," Royal Inst. Technol., Stockholm, Sweden, Rep. TRITA-NA-8607, 1986.
[22] J. O. Eklundh and J. Howako, "Robust shape description based on curve fitting," in *Proc. 7th ICPR,* Montreal, P.Q., Canada, 1984, pp. 109–112.
[23] A. Bengtsson, "An algorithm for estimation of shape properties at multiple scales," Royal Inst. Technol., Stockholm, Sweden, in preparation, 1989.
[24] C. de Boor, *A Practical Guide to Splines.* New York: Springer-Verlag, 1978.
[25] T. Pavlidis, "Curve fitting with conic splines," *ACM Trans. Graphics,* vol. TOGS-1, pp. 1–31, 1983.
[26] T. Lindeberg and J. O. Eklundh, "On the computation of a scale-space primal sketch," submitted for publication.
[27] F. Ulupinar and R. Nevatia, "Using symmetries for analysis of shape from contour," in *Proc. 2nd ICCV,* Tarpon Springs, FL, 1988, pp. 414–426.