Hierarchical Curve Reconstruction. Part I: Bifurcation Analysis and Recovery of Smooth Curves. *

Stefano Casadei and Sanjoy Mitter

Massachusetts Institute of Technology, Cambridge Ma, 02139

Abstract. Conventional edge linking methods perform poorly when multiple responses to the same edge, bifurcations and nearby edges are present. We propose a scheme for curve inference where divergent bifurcations are initially suppressed so that the smooth parts of the curves can be computed more reliably. Recovery of curve singularities and gaps is deferred to a later stage, when more contextual information is available.

1 Introduction

The problem of curve inference from a brightness image is of fundamental importance for image analysis. Computing a curve representation is generally a difficult task since brightness data provides only uncertain and ambiguous information about curve location. Two sources of uncertainty are curve bifurcations (junctions) and "invisible curves" (e.g. the sides of the Kanisza triangle). Local information is not sufficient to deal with these problems and "global" information has to be used somehow. Methods based on optimization of a cost functional derived according to Bayesian, minimum description length, or energy-based principles [4, 9, 11, 12, 20] introduce global information by simply adding an appropriate term to the cost functional. These formulations are simple and compact but lead usually to computationally intractable problems. Moreover, it is often difficult or impossible to guarantee that the optimal solution of these cost functionals represents correctly all the desired features, such as junctions and invisible curves [16]. Curve evolution approaches, where the computed curves are defined to be the stationary solutions of some differential equation [7, 19], can be computationally efficient but usually require some external initialization in order to converge to the desired solution.

A way to exploit global information efficiently without the need of external initialization is to use a hierarchy of intermediate representations between the brightness data and the final representation. The complexity and spatial extent of the descriptors in these representations increase gradually as one moves up in this hierarchy. Global information is introduced gradually so that computation is always efficient.

* Research supported by US Army grant DAAL03-92-G-0115, Center for Intelligent Control Systems

An intermediate representation used by most hierarchical approaches is given by of a set of points or tangent vectors obtained by locating and thresholding the maxima of the brightness gradient [1, 6, 15]. Linking and recursive grouping techniques exist to connect these points into polygonal curves [8, 3]. This aggregation is based on "perceptual organization" criteria such as collinearity and proximity [10]. Iterative procedures, such as relaxation labeling, have also been proposed to infer a set of curves from a set of tangent vectors [13, 5].

To guarantee that computation is efficient and robust, the hierarchy of representations should be "smooth". That is any two consecutive levels of the hierarchy should be "close" so that each level contains all the information necessary to reconstruct the objects at the following level efficiently and robustly. Consider for instance the following hierarchy:

$$\begin{aligned} \textit{brightness data} &\rightarrow \textit{tangent vectors} \\ &\rightarrow \textit{smooth curves} \\ &\rightarrow \textit{curves with corners and junctions} \\ &\rightarrow \textit{closed partly invisible curves} \\ &\rightarrow \textit{overlapping regions ordered by depth.} \end{aligned}$$

Notice that the first stage of this hierarchy does not recover curve singularities (corners and junctions) nor invisible curves. These are recovered later when information about the smooth portions of the curves is available. Resolving uncertainty in small steps allows the algorithm to make difficult decisions when more information is present. In fact, what is uncertain or ambiguous at some level of the hierarchy might become certain and unambiguous at a higher level when more global and contextual information is available. Also, to achieve robustness, it is important that uncertainties be not eliminated arbitrarily, as is done by many threshold-based methods. Rather, these uncertainties should be represented explicitly and propagated to the higher levels. Thus it is important to understand what can be computed reliably at any particular stage and what should instead be deferred until more contextual information is present.

To illustrate this point, observe the output of the Canny edge detector followed by conventional greedy edge-point linking (figure 4, third and fourth columns). Every edge-point is linked to the neighboring point which is best aligned with the local estimate of edge orientation. If the thresholds of the edge-point detection algorithm are set to low values then many edges are completely missed (third column). On the other hand, if these thresholds are set to higher values, then more edge points are present but conventional edge linking fails in the vicinity of curve singularity and when edges are close to each other (fourth column). The algorithm is trying to make too many difficult decisions at the same time.

Uncertainty in edge linking occurs at bifurcations, namely points which can be linked to more than one other point. Notice that the paths from a bifurcation remain close to each other in some cases while in other cases they diverge. The first kind of bifurcation will be called *stable* and the second type *divergent.*

Stable bifurcation are typically caused by multiple responses to the same edge or uncertainty in curve localization. Divergent bifurcations can be either due to the topology of the curves to be reconstructed (e.g. junctions) or to noise and interference from nearby edges. Conventional edge linking does not distinguish between these different types of bifurcations and decides how each bifurcation should be disambiguated in a single step based solely on local similarity properties. Instead, the approach described in this paper resolves ambiguity at bifurcations in more than one stage. The first stage, described in this paper, "disables" temporarily divergent bifurcations and recovers smooth curves by disambiguating stable bifurcations. Curve singularity (junctions and corners) are left for a subsequent stage.

There are three reasons why divergent bifurcations are initially disabled. First, this eliminates the risk that a spurious path created by noise disrupts tracking of the true curve (see for instance the fourth column of figure 4). Secondly, it simplifies the task of resolving stable bifurcations since tracking is reduced to a one dimensional problem. Thirdly, the lost information about curve singularity is usually inaccurate because local edge-point detectors are known to have poor performance near singular points.

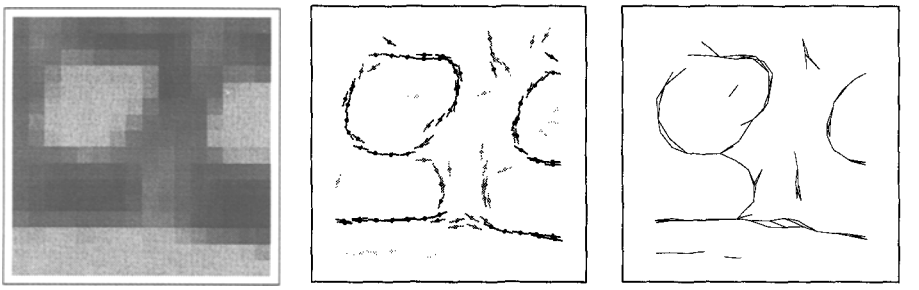
2 Suppression of divergent bifurcations

Let P be a set of points in $\mathbf{R}^2$ which represent sampled estimates of the location of the curves (figure 1(b)). In the experiments shown here this set has been obtained by locating the maxima of the brightness gradient in the gradient direction to sub-pixel accuracy. The gradient and its derivative have been estimated by fitting linear and cubic polynomials to blocks of 9 to 12 pixels respectively. For each $p \in P$, $\theta(p)$ denotes the estimated orientation of the curve given by the orthogonal direction to the brightness gradient and $\phi(p)$ denotes the gradient magnitude.

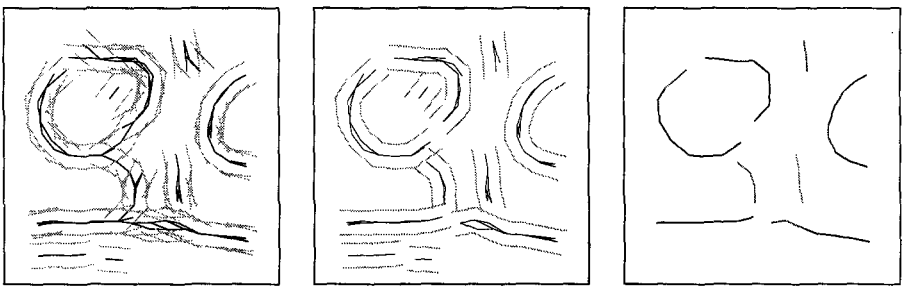
Let S (figure 1(c)) be the set of segments obtained by connecting every pair of points in P estimated from adjacent blocks of pixels (each point is connected to 8 other points). Segments whose orientation differs from the estimated curve orientation at its endpoints by more than $\Theta = 40°$ are discarded.

As figure 1(c) illustrates the planar graph associated with the set of segments S contains both stable and divergent bifurcations. Roughly speaking, stable bifurcations are associated with *collateral* paths, namely paths which remain "close" to each other whereas divergent bifurcations occur when these paths diverge. The definition below makes precise the distinction between collateral and divergent paths. The first part of the algorithm (table 1) computes a divergence-free subgraph of S by detecting pairs of diverging segments and "suppressing" the "weakest" segment of each such pair (figures 1(d) and 1(e)).

For any polygonal path π whose segments are elements of S let $N_w(\pi)$ be the set of points in $\mathbf{R}^2$ with distance from π less or equal to some $w > 0$ ($w = 0.75$ in our experiments). The boundary of $N_w(\pi)$, $\beta_w(\pi)$, is decomposed into its left,



(a) Brightness image. (b) The points P. (c) The segments S...



(d) ...and their lateral boundaries (gray). (e) Subgraph with no divergent bifurcations. (f) Final result.

Fig. 1. The steps of the proposed algorithm. A set of points P (b) is computed by locating the maxima of the brightness gradient. A planar graph (c) is obtained by connected neighboring points. Figure (d) shows the lateral boundaries $\beta_w^{\text{lat}}(s)$ (gray) of every segment in $s \in S$. Whenever the lateral boundary of some segment intersects another segment, the weaker of the two segments is suppressed. The resulting planar graph (e) does not contain divergent bifurcations. Finally, (f) shows the polygonal curves obtained by the procedure in table 2.

right, bottom and top parts (see figure 2): $\beta_w(\pi) = \beta_w^{\text{rig}}(\pi) \cup \beta_w^{\text{top}}(\pi) \cup \beta_w^{\text{left}}(\pi) \cup \beta_w^{\text{bot}}(\pi)$. Let also $\beta_w^{\text{lat}}(\pi) = \beta_w^{\text{left}}(\pi) \cup \beta_w^{\text{rig}}(\pi)$ (lateral boundary).

Definition 1 *Two paths π_1, π_2 (figure 2) are said to be* independent *if*

$$\pi_1 \cap N_w(\pi_2) = \pi_2 \cap N_w(\pi_1) = \emptyset \tag{1}$$

They are collateral *if they are not independent and*

$$\pi_1 \cap \beta_w^{\text{lat}}(\pi_2) = \pi_2 \cap \beta_w^{\text{lat}}(\pi_1) = \emptyset \tag{2}$$

Finally, π_1, π_2 are divergent if

$$\pi_1 \cap \beta_w^{\text{lat}}(\pi_2) \neq \emptyset \quad \text{or} \quad \pi_2 \cap \beta_w^{\text{lat}}(\pi_1) \neq \emptyset \tag{3}$$

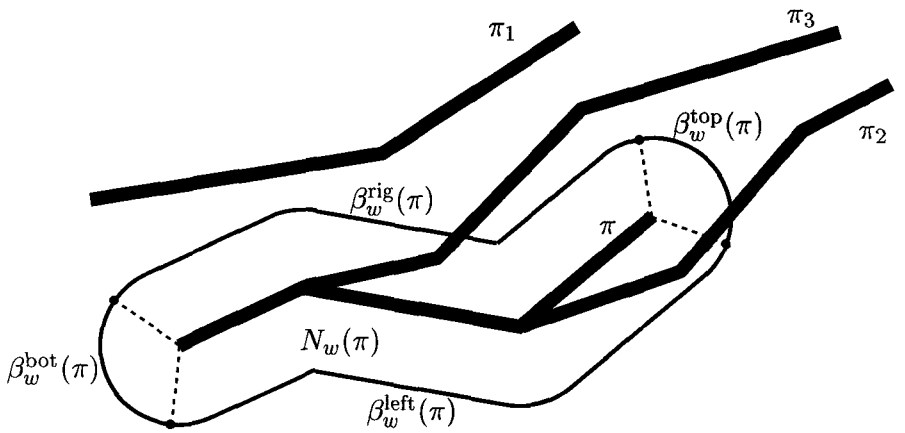


Fig. 2. Notation: $N_w(\pi)$ denotes the w-neighborhood of π. The boundary of $N_w(\pi)$, $\beta_w(\pi)$, is decomposed into four parts. The bottom and top parts are arcs of amplitude 120° and radius w. The paths π and π_1 are independent because their distance is greater than w, namely $\pi_1 \cap N_w(\pi_2) = \pi_2 \cap N_w(\pi_1) = \emptyset$. π and π_3 are divergent because $\pi_3 \cap \beta_w^{\text{rig}}(\pi) \neq \emptyset$. Finally, π and π_2 are collateral because π_2 intersects the boundary of $N_w(\pi)$ only at $\beta_w^{\text{top}}(\pi)$.

Then let $\delta_w(\pi_1, \pi_2)$ be the boolean variable which indicates whether two paths are divergent or not. Thus, $\delta_w(\pi_1, \pi_2) = 1$ if π_1 and π_2 are divergent and $\delta_w(\pi_1, \pi_2) = 0$ otherwise.

Definition 2 *A set of segments S is said to be* divergence-free *if it does not contain divergent regular paths, namely if for any pair of regular paths π_1, π_2 in S, $\delta_w(\pi_1, \pi_2) = 0$.*

For a definition of "regular" and a proof of the following results see [2]. The algorithm described in table 1, which extracts a divergence-free subset from S, hinges on the following proposition which "localizes" the notion of divergence.

Proposition 1 *Let S be a set of segments such that the orientation difference between two connected segments is never larger than 60°. Then S is divergence-free if and only if $\delta_w(s_1, s_2) = 0$ for any $s_1, s_2 \in S$.*

This proposition ensures that the property of being divergence-free is equivalent to a local condition which involves only pairs of neighboring segments. Notice that it assumes that the orientation difference between two connected segments is $\leq 60°$. Thus, to apply this result one has to eliminate from the graph all the segments which violate this condition (lines 3-6 in table 1).

The non-maximum suppression procedure described in table 1 computes a divergence-free subset S^{DF} of S. For this purpose, a positive function $\phi : S \to \mathbf{R}^+$

1	For every $s \in S$
2	$\alpha(s) := 1$
3	For every $s_1, s_2 \in S$, s_1, s_2 connected, $T(s_1, s_2) \geq 60°$
4	If $\phi(s_1) \leq \phi(s_2)$
5	$\alpha(s_1) := 0$
6	$S' = \{s \in S \ : \ \alpha(s) = 1\}$
7	For every $s_1, s_2 \in S'$ such that $\delta_w(s_1, s_2) = 1$
8	If $\phi(s_1) \leq \phi(s_2)$
9	$\alpha(s_1) := 0$
10	$S^{\mathrm{DF}} = \{s \in S \ : \ \alpha(s) = 1\}$

Table 1. The non-maximum suppression procedure to eliminate divergent bifurcations. $T(s_1, s_2)$ denotes the orientation difference between the segments s_1, s_2. The variable $\alpha(s)$ is initially set to 1 to indicate that every segment is initially "active". The loop in lines 3-5 ensures that S' does not contain pairs of connecting segments with orientation difference $> 60°$. This guarantees that proposition 1 can be applied to S'. The loop in lines 7-9 ensures that S^{DF} is divergence-free by disactivating the weaker segment of any pair of inconsistent (i.e. divergent) segments in S'.

is used to decide which of a pair of segments should be suppressed. The function ϕ used in our implementation is given by $\phi(s) = \min(\phi(p_1), \phi(p_2))$ where p_1 and p_2 are the end-points of s.

Proposition 2 *The graph S^{DF} defined in table 1 is divergence-free.*

If the length of the segments in S is bounded from above by some $L > 0$, then the two loops of the procedure need not be carried out over all pairs of segments. In fact, for each segment, it is enough to consider all the segments in a neighborhood of size $L/2 + w$ around its midpoint. If we further assume that the density of segments in the image is also bounded from above, then the complexity of the procedure is linear in the number of segments. The result of this procedure applied on the segments in figure 1(c) is shown in figure 1(e).

3 Computing the longest paths with minimum curvature

After eliminating the divergent bifurcations, the remaining ones, which are stable, can be disambiguated by computing a set of "maximally long" paths in S^{DF} which "cover" every possible path in S^{DF} (the notions of "maximally long" and "cover" are discussed more rigorously in [2]). The basic idea is as follows. Whenever a segment is connected to more than one other segment, the one which yields the "longest" path is selected. If this is not sufficient to remove all the ambiguities, namely there are two or more choices yielding "equally long" paths,

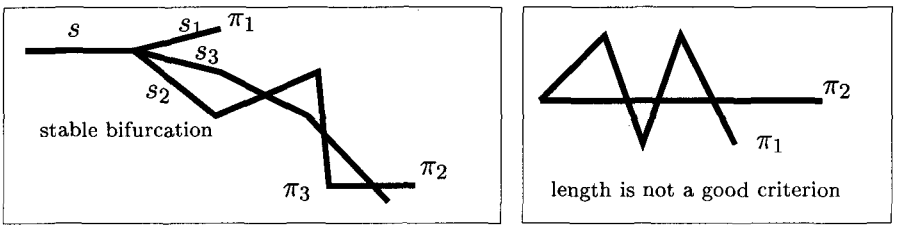


Fig. 3. Left: The segment s is connected to three other segments, s_1, s_2, s_3 which yield three different paths π_1, π_2, π_3. The algorithm in table 2 links s to s_3. Right: The length of π_1 is greater than the length of π_2. Yet, the path π_2 covers π_1 while π_1 does not cover π_2.

then the one which minimizes the total turn (that is the sum of the orientation differences of consecutive segments) is selected (see figure 3, left).

Notice that the length of a path is not the right measure to decide whether a path covers or is "longer" than another path (see figure 3, right). A more reliable way to check that a path π_1 covers π_2 is to test whether π_1 intersects $\beta_w^{\text{top}}(\pi_2)$. In fact, one can prove that if π_1, π_2 are two regular paths in S^{DF} having the same initial segment then $\pi_1 \cap \beta_w^{\text{top}}(\pi_2) = \emptyset$ implies that π_2 covers π_1 in a precise sense [2]. Conversely, $\pi_1 \cap \beta_w^{\text{top}}(\pi_2) \neq \emptyset$ implies that π_1 covers π_2.

Let $C(s)$ denote all the segments connected to $s \in S^{\text{DF}}$ which form an obtuse angle with it. The algorithm computes three variables for each segment:

- $\nu(s) \in C(s)$ is segment linked to s.
- $\lambda(s)$ is the last segment of $\pi^\star(s)$, where $\pi^\star(s)$ denotes the "best" path starting at s. Initially, $\lambda(s)$ is set equal to *nil*.
- $\tau(s)$ is the total turn of $\pi^\star(s)$.

The segments of the the path $\pi^\star(s)$ are $s, \nu(s), \nu(\nu(s)), \ldots, \lambda(s)$. The variables $\nu(s)$, $\lambda(s)$, $\tau(s)$ are computed by a procedure *link*(s) which calls itself recursively on the elements of $C(s)$. After the procedure *link*(s) returns, the variables $\nu(s)$, $\lambda(s)$, $\tau(s)$ have already converged to their final value because this problem is a special case of dynamic programming. This approach is similar to the saliency maximization technique used in [17, 18]. However, the method proposed here is more computationally efficient since the procedure is called exactly once on every segment and no multiple iterations of any sort are needed. A way to visualize the algorithm is to think that the procedure *link*(s) explores all the possible paths originating from s but it prunes the search every times it hits a segment which has already been solved. The procedure is described in table 2 for the case where no close loops are present. The details to deal with close loops are explained elsewhere. Since each step can be done in constant time and the procedure is called exactly once for every segment (this is ensured by the condition at line 8) the procedure runs in linear time in the number of segments.

After the procedure *link* has been called on all the segments, all the bifurcations have been disambiguated and every segment is linked to at most one

Definition of $link(s)$
1 If $C(s) = \emptyset$
2 $\lambda(s) := s$
3 $\tau(s) := 0$
4 $\nu(s) := nil$
5 return
6 else
7 For every $s' \in C(s)$
8 If $\lambda(s') = nil$
9 $link(s')$
10 $C^\star = \{s' \in C(s) \ : \ \beta_w^{\text{top}}(\lambda(s')) \cap \pi^\star(s'') = \emptyset, s'' \in C(s)\}$
11 $\nu(s) = \text{argmin}_{s' \in C^\star} \tau(s')$
12 $\tau(s) = \tau(\nu(s)) + T(s, \nu(s))$
13 $\lambda(s) = \lambda(\nu(s))$
14 return

Table 2. The procedure $link(s)$. The set $C^\star$ contains all the segments $s' \in C(s)$ yielding a path $\pi^\star(s')$ which is "maximally long" among the paths originating from s. In fact, if $s' \in C^\star$, the top boundary of the last segment of $\pi^\star(s')$, $\lambda(s')$, is not "cut" (i.e. intersected) by any other other path $\pi^\star(s'')$ originating from s. Note that the variables of all the segments in $\pi^\star(s'')$, $s'' \in C(s)$ have already been computed before execution arrives at line 10. The test $\beta_w^{\text{top}}(\lambda(s')) \cap \pi^\star(s'') = \emptyset$ can be done in constant time by detecting what segments intersect $\beta_w^{\text{top}}(\lambda(s'))$ and then checking whether any of these segments has its variable λ equal to some $\lambda(s'')$, $s'' \in C(s)$.

segment. To extract the paths explicitly one has to identify a good set of initial points. The details of how this is done are explained in [2].

4 Experimental results

The result of the algorithm on four test images are shown in figure 4, second column. The parameter w has been set to 0.75 for all the experiments. The other parameters (used to compute P) have been kept constant also. The results are compared against Canny's algorithm with sub-pixel accuracy followed by greedy edge linking, as implemented in [14] (smoothing scale = 0.7)

5 Conclusions

To infer curves reliably and efficiently from a set of edge-points or tangent vectors it is necessary to deal with bifurcations appropriately and resolve uncertainties in the right context. We proposed a hierarchical scheme for curve reconstruction whose early stages are based on the distinction between stable and divergent

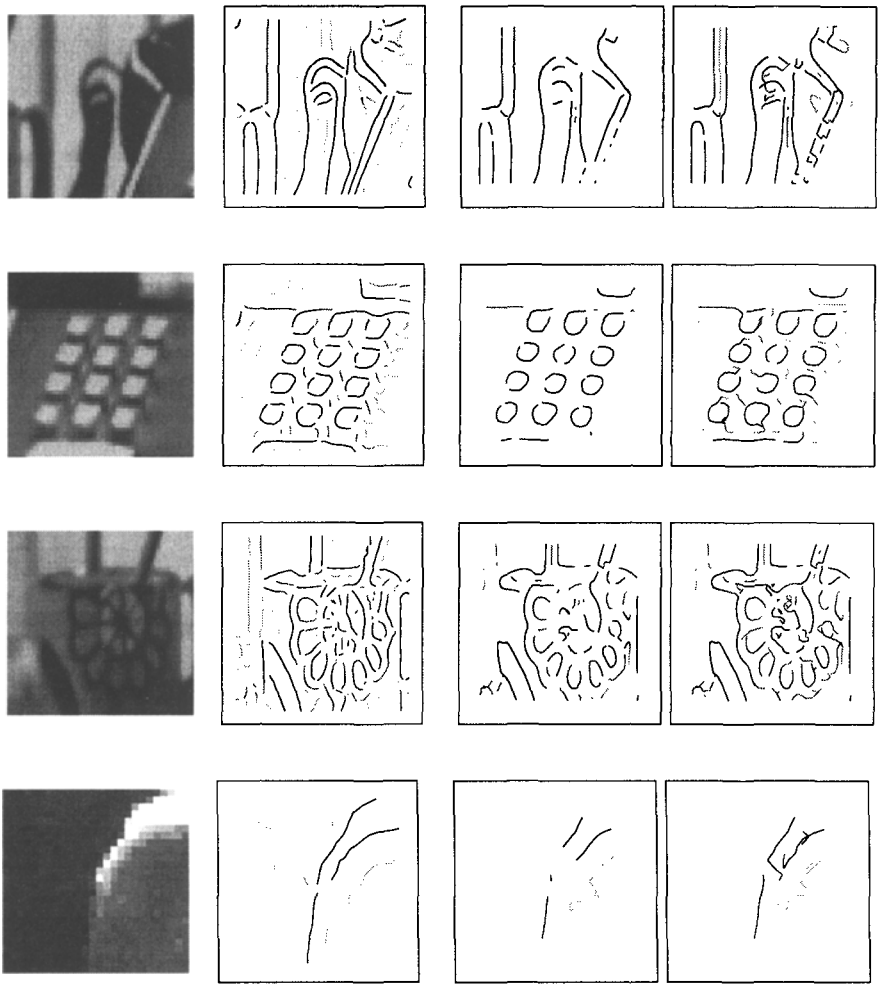


Fig. 4. The results obtained by the proposed algorithm are shown in the second column. The third and fourth column are the result of Canny's algorithm with two different sets of thresholds. The gray level of the edges is proportional to the magnitude of the brightness gradient.

bifurcations. The first stage of this hierarchy, described in this paper, recovers only the smooth portions of the curves. Work is currently being done to design and implement the following stages which combine the representation computed by the first stage with more global information to recover junctions, gaps and to construct a region-based description of the image. Recently, some theoretical results have been proved which guarantee robust performance of the algorithm in reconstructing a class of smooth curve models.

References

1. J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.
2. S. Casadei and S.K. Mitter. A hierarchial approach to high resolution edge contour reconstruction. Unpublished, October 1995.
3. J. Dolan and E. Riseman. Computing curvilinear structure by token-based grouping. In *Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1992.
4. S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on PAMI*, 6:721–741, November 1984.
5. E.R. Hancock and J. Kittler. Edge-labeling using dictionary-based relaxation. *IEEE trans. Pattern Anal. Mach. Intell.*, 12:165–181, 1990.
6. R.M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:58–68, 1984.
7. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.
8. D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer, 1985.
9. J. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *J. Am. Stat. Ass.*, 82(397):76–89, Mar. 1987.
10. Rakesh Mohan and Ramakant Nevatia. Perceptual organization for scene segmentation and description. *IEEE trans. Pattern Anal. Mach. Intell.*, 14, June 1992.
11. D. Mumford and J. Shah. Boundary detection by minimizing functionals. *Image Understanding 1989*, 1:19–43, 1989.
12. M. Nitzberg and D. Mumford. The 2.1-d sketch. In *Proceedings of the Third International Conference of Computer Vision*, pages 138–144, 1990.
13. Pierre Parent and Steven W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE trans. Pattern Anal. Mach. Intell.*, 11, August 1989.
14. P. Perona, 1995. Lecture notes, Caltech.
15. P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *Proceedings of the Third International Conference of Computer Vision*, pages 52–57, Osaka, 1990. IEEE Computer Society.
16. T. J. Richardson. Scale independent piecewise smooth segmentation of images via variational methods. Technical Report LIDS-TH-1940, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, feb 1990.
17. A. Sha'ashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *Second International Conference on Computer Vision (Tampa,, FL, December 5–8, 1988)*, pages 321–327, Washington, DC,, 1988. Computer Society Press.
18. J.B. Subirana-Vilanova and K.K. Sung. Perceptual organization without edges. In *Image Understanding Workshop*, 1992.
19. A. Tannenbaum. Three snippets about curve evolution, 1995.
20. S.C. Zhu, T.S. Lee, and A.L. Yuille. Region competition: unifying snakes, region growing, and bayes/mdl for multi-band image segmentation. Technical Report CICS-P-454, Center for Intelligent Control Systems, march 1995.