Hierarchical Image Segmentation – Part I: Detection of Regular Curves in a Vector Graph *

Stefano Casadei, Sanjoy Mitter
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
e-mail: casadei@lids.mit.edu

May 17, 1996

Abstract

The problem of edge detection is viewed as a hierarchy of detection problems where the geometric objects to be detected (e.g. edge points, curves, regions) have increasing complexity and dimensions. A very early stage of this hierarchy consists in detecting the regular portions of the visible edges. The input to this stage is given by a graph whose vertices are tangent vectors representing local and uncertain information about the edges. A precise model relating the input vector graph to the curves to be detected is proposed. An algorithm is described which provably solves the corresponding detection problem robustly and efficiently. The stability of curve reconstruction in the presence of uncertain information and multiple responses to the same edge is analyzed and addressed explicitly by the algorithm.

*Research supported by US Army grant DAAL03-92-G-0115, Center for Intelligent Control Systems and US Army grant DAAH04-95-1-0494, Center for Imaging Science (Washington University).

1 Introduction

1.1 Global information and compositional hierarchies

The problem of curve inference from a brightness image is of fundamental importance for image analysis. Curve detection and reconstruction is a difficult task since brightness data provides only uncertain and ambiguous information about curve location. A source of uncertainty is for instance the presence of "invisible curves", namely curves across which there is no brightness change (Kanizsa, 1979). Local information is not sufficient to resolve these uncertainties reliably 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 (Geman and Geman, 1984; Marroquin et al., 1987; Mumford and Shah, 1989; Nitzberg and Mumford, 1990; Nitzberg et al., 1993; Zhu et al., 1995) introduce global information by simply adding an appropriate term to the cost functional. These formulations are simple and compact but usually lead 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 (Richardson and Mitter, 1994). Curve evolution approaches, where the computed curves are defined to be the stationary solutions of some differential equation (Kass et al., 1988; Tannenbaum, 1995), can be computationally efficient but require an appropriate external initialization in order to converge to the desired solution. Iterative procedures, such as relaxation labeling, can produce good results but only at a high computational cost (Parent and Zucker, 1989; Hancock and Kittler, 1990).

A major difficulty is that in order to exploit global constraints it is necessary to use large "contextual neighborhoods" so that interaction between data from distant locations is possible. This typically causes a combinatorial explosion of the search space since the number of states of each neighborhood grows exponentially with its size. A strategy to avoid this combinatorial explosion is to increase the size of the contextual neighborhoods

in stages. This leads to a hierarchy of representations whose primitive elements have larger spatial extent and more internal structure as one moves up in the hierarchy. Large and more complex descriptors can act as efficient carriers of global information. The increase of the interaction distance and of the descriptors' complexity across two adjacent levels should be small enough so that computation is always "local" and therefore efficient.

This approach is similar to multiscale schemes such as wavelet analysis in that representations at a large scale are constructed efficiently from local data by using the appropriate number of intermediate levels. The main difference is that the dictionary of primitive elements used in wavelet-like multiscale approaches does not change across the levels, except for a dilation transformation. As a result, information is lost or simplified and representations become coarser at larger scales. On the contrary, we are interested in hierarchies where the higher levels contain *more* information than the lower levels. Thus the expressive power of the underlying dictionary has to increase when moving up in the hierarchy.

Whereas the basic transformation underlying wavelet and similar multiscale representations is a dilation applied to the domain of the raw data, the hierarchies considered here are based on a *compositional* transformation. That is, the models to be detected are decomposed recursively into simpler units, leading to a *hierarchy of models.* Computation is mostly a bottom-up process which detects and reconstructs these models by means of composition. Top-down feedback, whose laws are derived from the model hierarchy, is used to select groupings of primitive units which are consistent with higher level models. Pruning the exponentially large search space of all possible groupings is necessary to keep computation efficient and can be done by using top-down feedback. A *hierarchy of descriptions* obtained by recursive composition of descriptors is generated from the input data as a result of this process.

To achieve robustness and computational efficiency, one needs to design a "smooth" hierarchy. That is, any two consecutive levels of the hierarchy should be "close" to each

other so that each level contains all the information necessary to reconstruct the objects at the following level efficiently and robustly. Thus, to design the next level of the hierarchy (in a bottom-up design approach), one has to understand what composite objects can be formed efficiently and robustly from the parts present in the current level. This constraint limits the amount of expressive power which can be added from one level to the next and therefore determines how many levels in the hierarchy are needed to bridge the gap between the input data and the desired final representation. For instance, in Section 1.2 it will be argued that in edge detection curve singularities and invisible curves should *not* be included in the lowest level curve dictionary but only in higher level ones.

By using a hierarchical approach, it is easier to resolve uncertainties and ambiguities at the right level, when the necessary contextual information is available. Typically, it is impossible to eliminate efficiently all uncertainties in a single step. Then, one should eliminate part of the uncertainties first and then use the new, more informative representation to resolve more uncertainties. Those uncertainties which can not be resolved should be propagated to the higher levels, rather than being resolved arbitrarily. Also, at every level, only those decisions which are necessary to avoid a combinatorial explosion of the search space should be taken. Therefore, in general, intermediate representations are quite uncertain and ambiguous and may contain mutually inconsistent hypotheses.

1.2 Perceptual organization as a detection problem

Many authors have suggested that hierarchical algorithms can be useful to infer global structures efficiently (Dolan and Riseman, 1992). Computation of global descriptors from simpler ones is related to *perceptual organization* (Sarkar and Boyer, 1993; Mohan and Nevatia, 1992; Lowe, 1985). Perceptual organization, which can be repeated recursively and hierarchically, consists in grouping descriptors according to properties such as proximity, continuity, similarity, closure and symmetry (Kanizsa, 1979). Every grouping of

descriptors is then composed into a higher level, more global, descriptor. To assess the significance of groupings in a task independent fashion, principles such as non-accidentalness (Lowe, 1985) and minimum description length (Bienenstock and Geman, 1995) have been proposed. These principles provide a general framework to design all the grouping procedures in the hierarchy. However, they do not guarantee per se that particular classes of objects are detected and reconstructed correctly by these procedures. The ultimate goal of perceptual organization is to detect and represent explicitly all the relevant structures present in the data. Thus, perceptual organization can be viewed as a *detection* problem where the a-priori dictionary consists of all the higher level descriptors which may be used to represent compactly any groupings of low level descriptors. A detection algorithm is successful if it makes explicit all the object in the dictionary which are consistent with the set of low level descriptors in the input representation. For instance, the problem of grouping edge-points into curvilinear structures can be formulated as a problem of curve detection by defining a class of curves and their relationship with their constituent sub-components (e.g. points or tangents). One such model is proposed in Section 2.

In a hierarchical approach, perceptual organization is really a *hierarchy* of detection problems. Each problem consists in computing explicitly all the objects in the dictionary of that level which are consistent with the data at the previous level. Detection can be achieved by composition: object sub-components are detected first and then composed to reconstruct the object. This aggregation method is repeated at every level up to the top level which contains the desired global description.

To guarantee robust performance, the representation computed at each level must be *complete* with respect to the dictionary of that level. In other words, all the elements of the dictionary consistent with the data must be present in the computed representation. For instance, the set of regular curves computed by the algorithm proposed in the following sections contains an approximation to every regular curve which is consistent with the input set of tangent vectors. The computed description represents explicitly all the

important groupings of tangent vectors and is therefore complete.

1.3 A hierarchy of models for image segmentation

Let us now examine how this approach can be applied to the problem of edge detection and image segmentation. The goal of image segmentation is to detect the boundaries (edges) of the objects in the visual scene. The main source of information for image segmentation is that most of these edges generate sharp brightness discontinuities. The result can be represented as a set of curves or, at a still higher level, by a set of planar regions ordered by depth (see for instance the *2.1D sketch* proposed in (Nitzberg and Mumford, 1990)).

What is a hierarchy suitable for this task? Curves representing edges can be naturally decomposed into a set of tangent vectors plus a set of singularities (corners, junctions and self-intersections). Efficient methods exist to estimate these tangent vectors from the brightness data where the brightness discontinuity is large enough compared to the noise (Canny, 1986; Haralick, 1984; Perona and Malik, 1990). (Singularities can also be estimated directly from the brightness data but only at a high computational cost (Deriche and Blaszka, 1993; Rosenthaler et al., 1992; Rohr, 1992)). These tangent vectors can then be composed to yield a set of curves. What dictionary of curves should be used for the first step of curvilinear organization? According to the hierarchical principles discussed before, one should use a curve model simple enough to ensure that the resulting detection problem can be solved reliably and efficiently. The model proposed in Section 2 yields a detection problem which can be solved robustly in linear time. Only regular curves (namely curves with no singularities) are considered in this model. Its two basic assumptions are that the tangent vectors are aligned with the curves to be detected and their magnitude is maximum in a neighborhood of these curves (these assumptions will be relaxed to include noise in the model). Therefore, corners, junctions, and curves where the brightness change is too small (invisible curves) can not be recovered at this stage. Once the regular portions

of the visible curves have been detected, corners and junctions can be recovered more easily at the following stage. Finally, invisible curves and two dimensional descriptors (regions) can be detected. A possible hierarchy for the whole edge detection task is then composed of the following stages: *brightness data* $\rightarrow$ *tangent vectors (vector graph)* $\rightarrow$ *regular visible curves* $\rightarrow$ *visible curves with corners and junctions* $\rightarrow$ *closed partly invisible curves* $\rightarrow$ *overlapping regions ordered by depth (2.1 sketch).*

Notice that the whole detection problem has been broken down into a set of detection tasks where the "distance" between the input and output dictionaries is small.

1.4 Outline

This paper deals with the second stage of this hierarchy, namely detection of regular curves from a set of tangent vectors. The set of these vectors is endowed with a graph structure by putting an arc between every pair of vectors estimated in adjacent regions. Thus the data can be represented by a *vector graph*, namely a discrete vector field with a graph structure. Section 2 describes vector graphs in more details. In Section 3 the concept of stability in edge linking and the notion of stable graphs are introduced. It will be argued that some of the errors incurred by conventional linking algorithms are due to the instability of the underlying search graph.

The algorithm proposed to detect regular curves is composed of two parts. First, a stable graph is computed from the initial vector graph (Sections 4 and 5). It is proved that this stable graph preserves the relevant information about the regular curves to be detected. At the same time, the uncertainties which may cause instability are removed so that an approximation of every curve in Γ can be computed efficiently. Secondly, the remaining ambiguities are resolved by selecting the paths with minimum turn (Section 6). Section 7 discusses how to deal with cycles in the graph. The results of experiments are presented in Section 8. The Appendix contains the proofs of the theorems.

2 Vector graphs

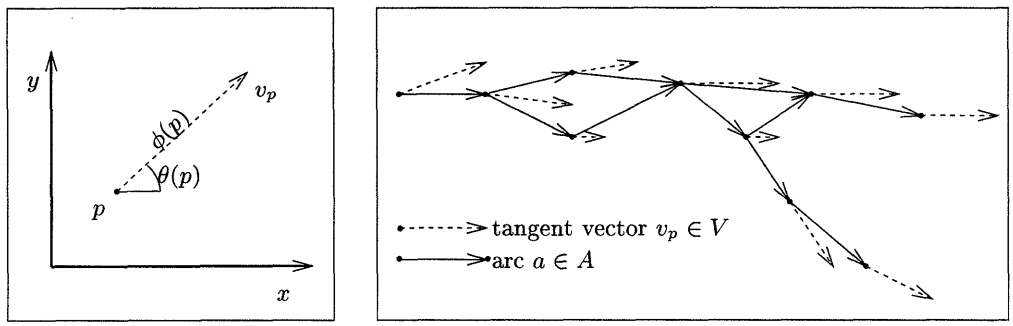
This paper addresses the problem of detecting and reconstructing a set of regular curves Γ from local and noisy information about these curves. This information is represented by a *vector graph*, namely a triple (P, V, A) where

- $P \subset \mathbb{R}^2$ is a finite set of candidate curve points;
- $V = \{v_p \ : \ p \in P\}$ is a discrete vector field with vertices P;
- $A \subset P \times P$ represents a set of directed arcs.

A directed arc between p_1 and p_2 is represented by a pair $(p_1, p_2) \in P \times P$. The pair (P, A) is a *directed graph* with arcs A and vertices P. Figure 1 shows an example of a vector graph and introduces some useful notation. The orientations $\theta(p)$ of the tangent vectors $v_p \in V$ represent estimates of the local orientation of the curves in Γ and the magnitude $\phi(p)$ of these vectors measures the likelihood that each candidate point $p \in P$ belongs to some curve. The vector graph (P, V, A) can be viewed as a noisy local projection of the curves in Γ onto small neighborhoods of the image. A model of the relationship between Γ and the vector graph (P, V, A) is proposed in Section 2.3.

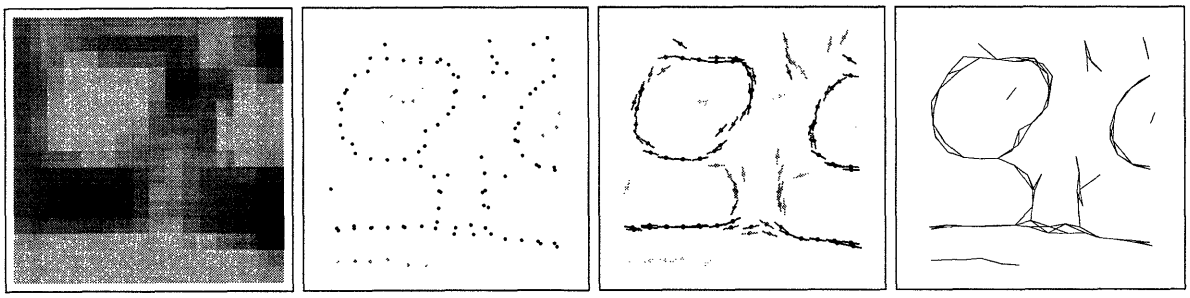
2.1 Computation of the vector graph

What method is used to compute the vector graph from the brightness image is irrelevant as long as the vector graph satisfies the three assumptions in Section 2.3. In our implementation, a fitting method similar to (Haralick, 1984) has been used. This method assumes that brightness changes significantly across the curves to be detected. Furthermore, it is assumed that the scale at which this change occurs is known. To compute a set of tangent vectors, the image is tiled with a set of overlapping regions. One tangent vector is computed from each region by means of the following steps:



(a) Tangent vector (b) Vector graph

Figure 1: (a): Notation. (b): Example of a vector graph. Solid segments represent arcs of the directed graph and dashed segment represent tangent vectors.



(a) Brightness (b) Points P (c) Vectors V (d) Arcs A

Figure 2: The vector graph (P, V, A) computed from the brightness image in (a) by using the method described in Section 2.1. (b): The estimated curve points P. (c): The tangent vectors V. The magnitude of the tangent vectors is coded by the gray level of the segments. (d): The directed graph structure (the direction of the arcs is not shown).

- Estimate the brightness gradient in the region by fitting a third order polynomial to the brightness data.
- Locate to sub-pixel accuracy the point where the brightness gradient is locally maximum in the direction of the gradient and let p be this point.
- Let $\theta(p)$ be the direction orthogonal to the gradient at p. This is an estimate of the orientation of the curve passing through p.
- Let $\phi(p)$ be the gradient magnitude at p. In general, $\phi(p)$ is some positive quantity depending on the gradient magnitude (and maybe also on the fitting error) which represents some sort of feedback from the brightness image. This feedback evaluates the likelihood that there exists indeed a curve with orientation $\theta(p)$ passing through p.
- Let v_p be the tangent vector with foot p, orientation $\theta(p)$, and magnitude $\phi(p)$.

Then, let P be the set of all the estimated points p and let V be the set of all the tangent vectors v_p, $p \in P$. V is a *discrete vector field.* The set of arcs A is then given by the set of all pairs $(p_1, p_2) \in P \times P$ estimated from adjacent regions and aligned with v_{p_1} (namely such that $(p_2 - p_1) \cdot v_{p_1} \geq 0$). A *path* in this graph is a sequence $\pi = (p_1, \ldots, p_n)$ such that $(p_i, p_{i+1}) \in A$, $i = 1, \ldots, n-1$. The corresponding sequence of vectors $(v_{p_1}, \ldots, v_{p_n})$ will also be called a path.

2.2 Relationship between the vector graph and Γ

Γ denotes the set of regular curves to be detected from the vector graph (P, V, A). What assumptions are appropriate to model the relationship between Γ and (P, V, A)? In the ideal case, when no noise is present, the following assumptions are quite natural:

- The vector graph contains a connected sampling of the tangent bundle of each curve $\gamma \in \Gamma$.

- The vector field V is locally maximum on every curve $\gamma \in \Gamma$.

Recall that the tangent bundle of a curve is the set of all its tangents. Thus, the first condition requires that for every curve $\gamma \in \Gamma$, the vector graph (P, V, A) contains a path whose vertices are tangents to γ. This is called the approximating path of γ. In order for this path to be a good approximation of γ, the length of the arcs in the graph has to be small enough.

The second condition is necessary because the graph may contain tangent vectors other than those belonging to curve-approximating paths. Thus, ideally, the magnitude of these spurious vectors is always smaller than nearby vectors belonging to a true curve.

When noise is present, the following distortions may be present:

- The vertices of the approximating paths are not exactly on the curve $\gamma \in \Gamma$. Let δ_0 be an upper bound on the distance of these vertices from γ.

- The tangent vectors v_p of the approximating path are not perfectly aligned with the curve tangents of the approximated curve γ. Let Θ_1 be an upper bound on this deviation.

- The magnitude of the vector field V may achieve the local maximum at some distance away from γ. Let δ_1 be an upper bound on this distance.

2.3 Formal assumptions

We now proceed to state the three assumptions which define the curve model in terms of the vector graph (P, V, A). These assumptions define rigorously the three distortion parameters above. For simplicity, the term "curve", which usually means a continuous mapping from an interval to $\mathbb{R}^2$, will be used to denote the image of the curve, which is a subset of $\mathbb{R}^2$. Thus, if γ is a curve, then $\gamma \subset \mathbb{R}^2$.

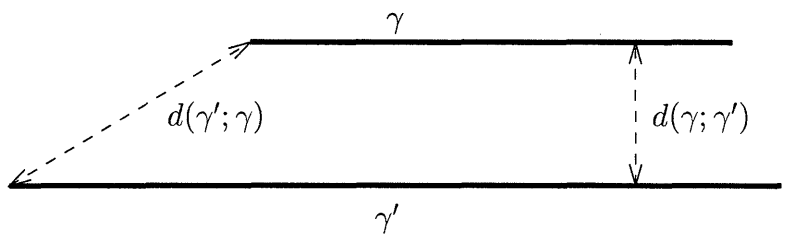


Figure 3: The asymmetric Hausdorff distance of γ from γ', denoted $d(\gamma;\gamma')$ is the maximum distance of a point in γ from the set γ'. Notice that $d(\gamma;\gamma') \neq d(\gamma';\gamma)$.

The polygonal curve defined by the path π is denoted $\sigma(\pi)$. It is given by the union of all the straight line segments $\sigma(a)$ on the path. Here $\sigma(a)$, $a = (p_1, p_2)$, denotes the set of points lying on the straight line segment with end-points p_1 and p_2.

Let S_1, S_2 be closed paths. The *asymmetric Hausdorff distance* of S_1 from S_2 is defined as

$$d(S_1;S_2) = \max_{p_1 \in S_1} d(p_1;S_2) = \max_{p_1 \in S_1} \min_{p_2 \in S_2} ||p_1 - p_2||$$

where $d(p_1;S_2)$ is the distance of the point p_1 from the set S_2. See Figure 3.

The first assumption requires that every curve $\gamma \in \Gamma$ has an approximating path in (P, A) with error bounded by some constant δ_0.

Covering condition: *The graph (P, A) covers Γ with distance δ_0. That is, for every $\gamma \in \Gamma$ there exists a path π in (P, A) such that $d(\gamma;\sigma(\pi)) < \delta_0$.*

The other two conditions involve only the vector field V and Γ. For simplicity, these constraints are formulated only for unbounded curves with zero curvature (namely infinite straight lines).

Let γ be any curve in Γ. The decay condition (see Figure 4) requires that the vector field V decays at a distance δ_1 from γ. More precisely, let δ_1, δ_2 be two scale parameters such that $0 < \delta_0 \leq \delta_1 < \delta_2$ (where δ_0 is the parameter used in the covering condition).

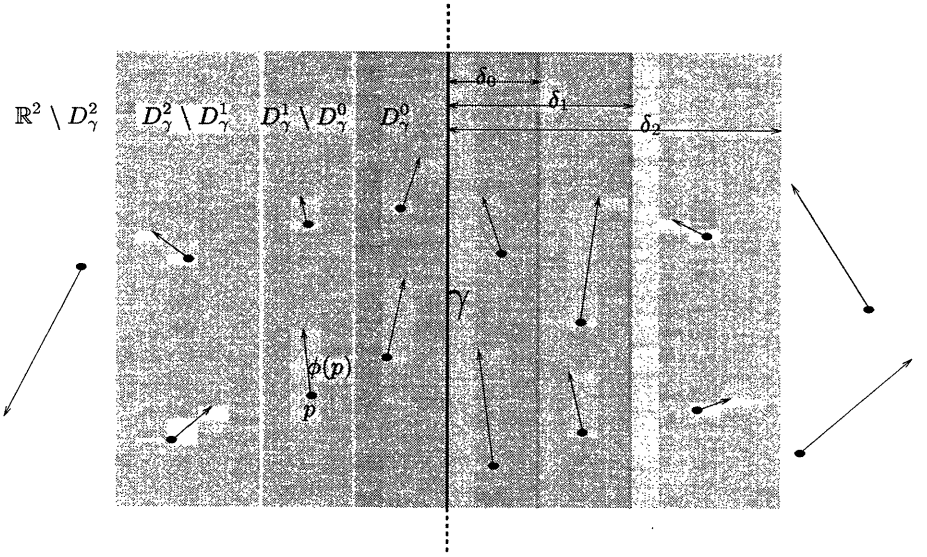


Figure 4: Constraints on the vector field in the vicinity of a curve. The magnitude $\phi(p)$ of the tangent vectors (length of arrows) is larger in D^0_γ than in $D^2_\gamma \setminus D^1_\gamma$. The magnitude $\phi(p)$ is arbitrary in $D^1_\gamma \setminus D^0_\gamma$. The angle formed by a tangent vector in D^1_γ with respect to the orientation of the curve is less than Θ_1.

Let D^i_γ, $i = 0, 1, 2$, be the neighborhoods of γ given by

$$D^i_\gamma = \left\{ p \in \mathbb{R}^2 \ : \ d(p;\gamma) < \delta_i \right\} \qquad i = 0, 1, 2$$

Decay condition:

$$p_1 \in D^0_\gamma;\ p_2 \in D^2_\gamma \setminus D^1_\gamma \quad \Longrightarrow \quad \phi(p_1) > \phi(p_2) \tag{1}$$

The parameter δ_2 is the distance up to which γ constrains the vector field V. Notice that the magnitude of V in $D^1_\gamma \setminus D^0_\gamma$ is arbitrary. This is to model the fact that, due to noise, the local maxima of $\phi(p)$ may be displaced from the inner neighborhood D^0_γ.

Finally, the alignment condition requires that the orientation of the vector field V at a distance from γ less than δ_1 does not deviate by more that Θ_1 from the orientation of γ. That is, if θ_γ denotes the orientation of γ,

Alignment condition:

$$p \in D_\gamma^1 \quad \Longrightarrow \quad ||\theta(p) - \theta_\gamma|| < \Theta_1 \tag{2}$$

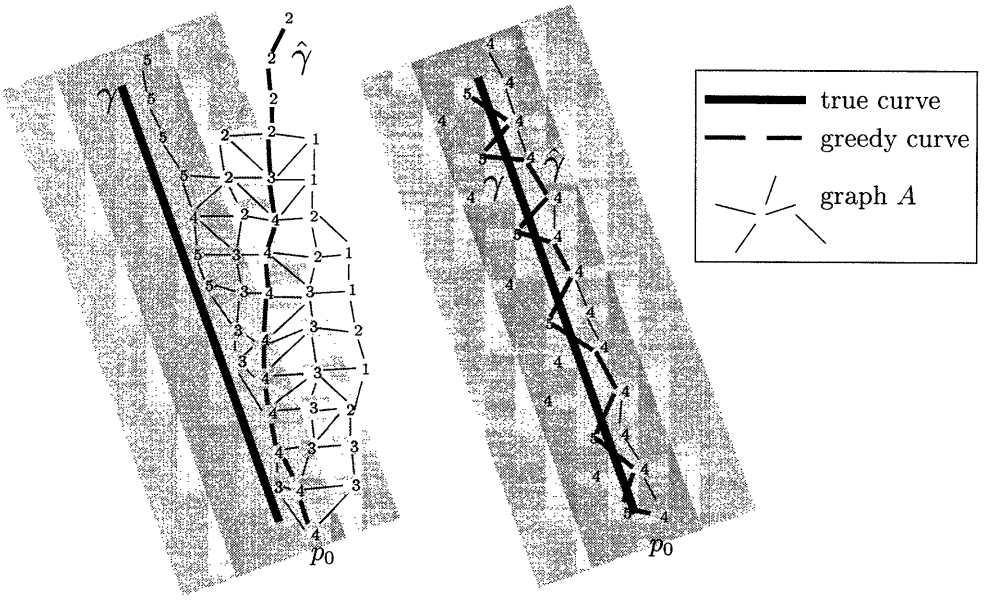
Definition 1 *Let Γ be a set of curve. The vector graph (P, V, A) is said to be a* projection *of Γ with admissible deviations δ_0, δ_1, δ_2 and Θ_1 if it satisfies the covering, decay and alignment conditions on Γ.*

3 Stability

The goal of the algorithm is to compute a set of disjoint curves $\hat{\Gamma}$ which approximate every curve in Γ. Attaining robust performance in the presence of the uncertainties implicit in the model described in Section 2.3 is potentially an intractable problem. In fact, interference due to nearby curves and uncertainty in curve location and orientation generate ambiguities as to how candidate points should be linked together to form a curve. These ambiguities result in bifurcations in the vector graph. The number of possible paths can be exponentially large and it might be impossible to explore efficiently all of them. On the other hand, to obtain a complete representation every plausible path must be somehow taken into account.

3.1 Inadequacy of greedy linking methods

Figure 5 illustrates the inadequacy of straightforward linking methods in the presence of uncertainties. Let's assume that there is just one curve γ to be detected, namely $\Gamma = \{\gamma\}$, and that γ satisfies the decay condition. Also, let's assume that $\delta_2 = \infty$. The vector field in the inner neighborhood D_γ^0 (dark shaded area) is larger than the vector field in the outer region $\mathbb{R}^2 \setminus D_\gamma^1$ (white area) but it is not necessarily larger than the vector field in the intermediate areas $D_\gamma^1 \setminus D_\gamma^0$ (the light shaded areas). Notice that in the example of



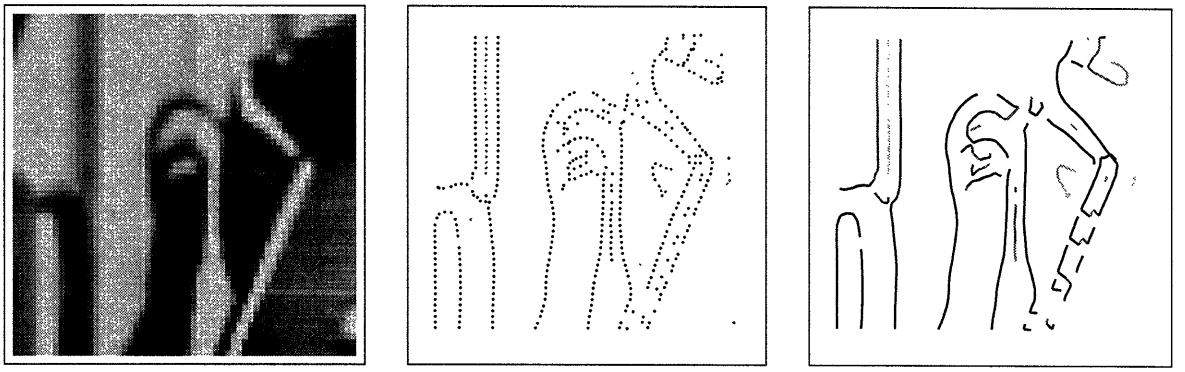
(a) Instability (b) Wiggly curve

Figure 5: Noise can cause instability and wiggly curves in greedy tracking algorithms. (a): The model assumes that the vector field in the inner neighborhood D_γ^0 (dark area) is larger than in the outer region $\mathbb{R}^2 \setminus D_\gamma^1$ (white area). Instability in curve tracking occurs because the maxima of $\phi(p)$ leak from D_γ^0 into $D_\gamma^1 \setminus D_\gamma^0$ (light gray areas). (b): Notice that noise can cause greedy linking to follow a wiggling path rather than the smoother path on the right.

Figure 5 the values of the vector field in the inner/intermediate/outer areas are $3, 4, 5$ / $2, 3, 4$ / $1, 2$ respectively.

Now suppose that a polygonal curve $\hat{\gamma}$ is constructed from the point p_0 by a greedy linking procedure. That is, the current point is linked to its strongest neighbor and then the procedure is restarted from the new point. Notice that the tracked path exits first the inner neighborhood, then the outer neighborhood and then it diverges arbitrarily from γ. Thus, this simple "greedy" procedure is *unstable* because a small deviation of the tracked path from the approximating path can lead to an arbitrarily large deviation between the two paths. This type of errors occur frequently in real images if greedy linking is used. An example of these errors is shown in Figure 6, which shows the result of an implementation

of Canny's edge detection algorithm followed by greedy edge linking (Perona, 1995).



(a) Brightness image (b) Edge-points (c) Greedy linking

Figure 6: Result of Canny's edge detection with greedy linking on the image shown in (a). The set of points found by Canny's algorithm is shown in (b). The polygonal curves obtained by linking each point to one of its neighbors are shown in (c). When ambiguities are present, a "best" neighbor is determined by minimizing a cost function which depends on the brightness gradient and on the orientation similarity between the linked points. Notice that these ambiguities, which are usually caused by multiple responses to the same edge, can disrupt the tracking process and lead to instability. That is, the reconstructed path can diverge significantly from the true edge. This is particularly evident for the two parallel edges of the bright thin long line on the right of the image.

3.2 Definition of stability

An important definition in this paper is that of a stable graph. Roughly speaking, a graph is stable if every path in the graph "attracts" nearby paths. Section 4 describes an algorithm which stabilizes a projection (P, V, A) of Γ. A weak definition of stability is given below and a stronger one will be given in Section 5. Both definitions depend on a positive constant w, which is the scale parameter used by the stabilization algorithm.

For $p \in \mathbb{R}^2$, $w > 0$, let $\mathbf{B}_w(p)$ be the ball centered at p with radius w:

$$\mathbf{B}_w(p) = \left\{ p' \in \mathbb{R}^2 \ : \ ||p - p'|| < w \right\}.$$

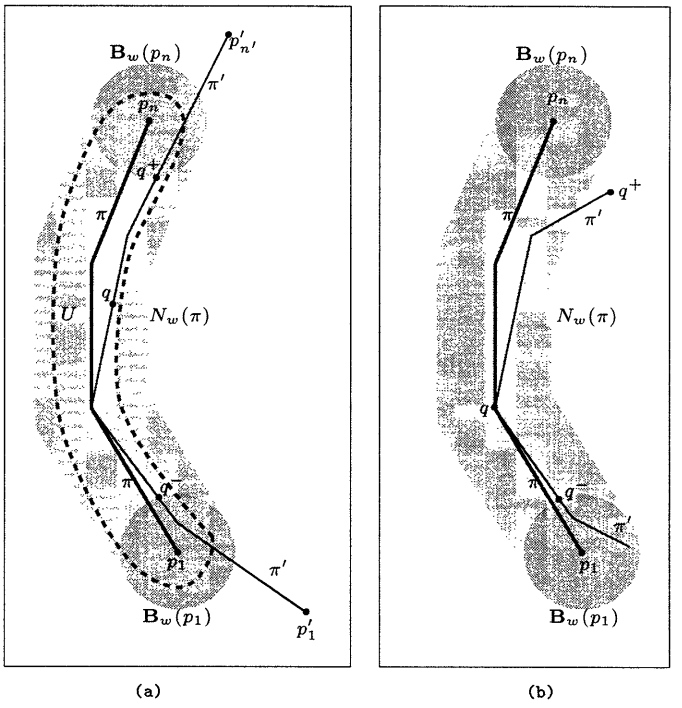


Figure 7: The path π in (a) is an attractor because the curve $\sigma_q(\pi')$, namely the curve between q^- and q^+, lies in some neighborhood U of $\sigma(\pi)$ contained in $N_w(\pi)$. The path π in (b) is not an attractor because there is no such neighborhood U of π. In fact, $\sigma_q(\pi')$ diverges (laterally) from π, that is, the subcurve $q \rightarrow q^+$ exits the neighborhood $N_w(\pi)$ without intersecting $\mathbf{B}_w(p_n)$.

The w-neighborhood of a path π is the set of points in $\mathbb{R}^2$ with distance from $\sigma(\pi)$ less than w. It is given by:

$$N_w(\pi) = \bigcup_{p \in \sigma(\pi)} \mathbf{B}_w(p).$$

Let $\pi = (p_1, \ldots, p_n)$ and $\pi' = (p'_1, \ldots, p'_{n'})$ be two paths and let $q \in \sigma(\pi')$. Let $\sigma_q(\pi')$ be the largest connected subcurve of $\sigma(\pi')$ which contains q and such that (see Figure 7)

$$\sigma_q(\pi') \cap \mathbf{B}_w(p_1) = \sigma_q(\pi') \cap \mathbf{B}_w(p_n) = \emptyset.$$

In Figure 7, $\sigma_q(\pi')$ is the curve between q^- and q^+.

Definition 2 *A path π in A is a* w-attractor *if there is a neighborhood U of $\sigma(\pi)$, $U \subset$*

$N_w(\pi)$, such that $\sigma_q(\pi') \subset U$ for every path π' in A and every $q \in \sigma(\pi') \cap U$. The set U is called an attraction basin *for π. A graph (P, A) is* w-stable *if every path in it is a w-attractor.*

4 Stabilization of a complete graph

This section describes an algorithm which computes a stable graph from an arbitrary vector graph (P, V, A). Moreover, if (P, V, A) is a projection of Γ then the result is also a projection of Γ. The set of arcs in the computed graph is denoted $S^w(V, A)$ where w is the scale parameter. This parameter is related to the constants of the curve model by means of the bounds in Theorem 2. Ultimately, w depends on the amount of noise in the brightness image and on the sharpness of the brightness discontinuity across edges.

To obtain a stable graph, the algorithm ensures that every path $\pi = (p_1, \ldots, p_n)$ in $S^w(V, A)$ has an attraction basin $U_w(\pi)$ contained in $N_w(\pi)$. The boundary of $U_w(\pi)$, denoted $\beta_w(\pi)$, is a polygonal curve constructed as follows. For every $p \in P$ let p^+ and p^- be the points lying w away from p in the direction orthogonal to the vector field at p. That is,

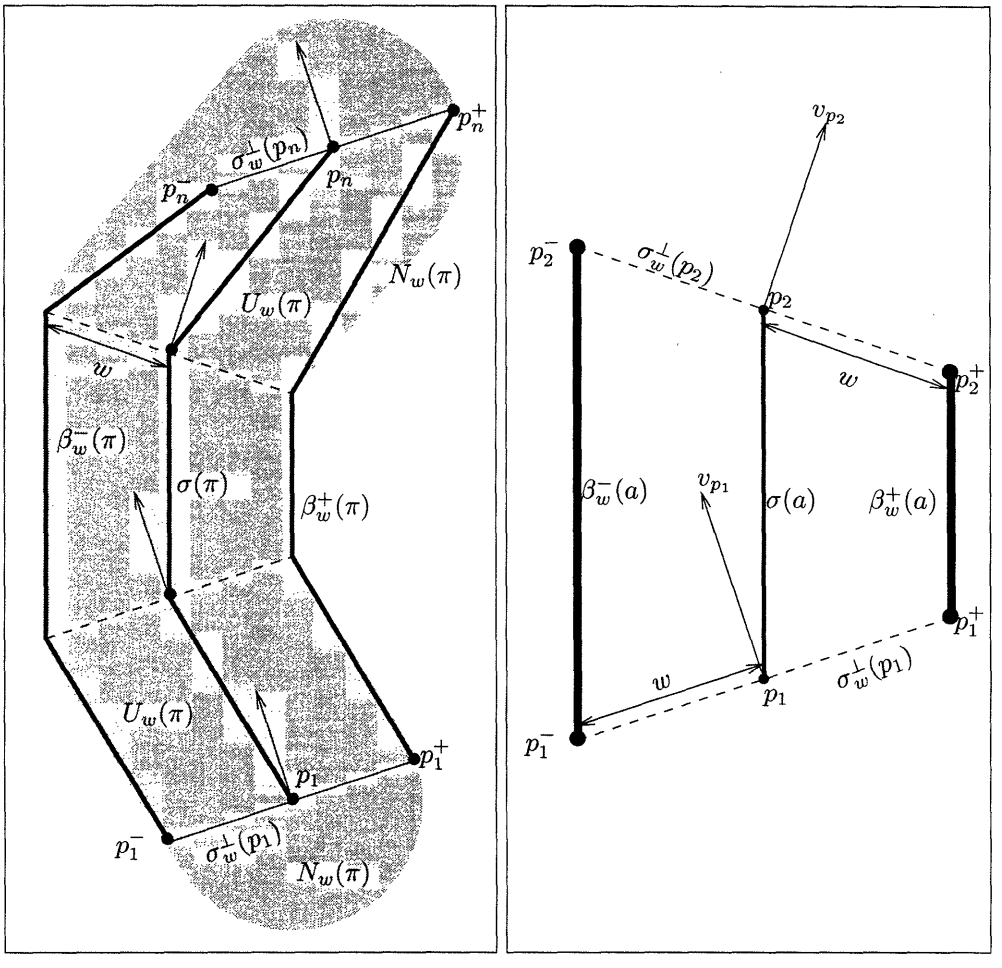
$$p^+ = p + wu_\perp(p) \tag{3}$$

$$p^- = p - wu_\perp(p) \tag{4}$$

where $u_\perp(p)$ is the unit vector perpendicular to the direction of the vector field at p, $u_\perp(p) = (\sin\theta(p), -\cos\theta(p))$. The boundary of $U_w(\pi)$ is then the polygonal curve with vertices (see Figure 8(a)):

$$p_1^+, \ldots, p_n^+, p_n^-, \ldots, p_1^-, p_1^+$$

The algorithm can be described as follows. Let $\beta_w^-(a)$, $\beta_w^+(a)$ be the *lateral segments* of



(a) Lateral boundaries of path π (b) Lateral segments of arc a

Figure 8: (a): The attraction basin $U_w(\pi)$ is the polygon with vertices $p_1^+, \ldots, p_n^+, p_n^-, \ldots, p_1^-, p_1^+$. Its perimeter $\beta_w(\pi)$ is composed of four parts $\beta_w(\pi) = \beta_w^-(\pi) \cup \sigma_w^\perp(p_n) \cup \beta_w^+(\pi) \cup \sigma_w^\perp(p_1)$ where $\sigma_w^\perp(p_i) = \sigma(p_i^-, p_i^+)$; $\beta_w^\pm(\pi) = \cup_{a \in A_\pi} \beta_w^\pm(a)$; A_π are the arcs of π; and $\beta_w^\pm(a)$ are the lateral segments of a shown in (b). Notice that each point in $U_w(\pi)$ has distance from $\sigma(\pi)$ less than w, that is, $U_w(\pi) \subset N_w(\pi)$.

the arc $a = (p_1, p_2) \in A$ defined by (see Figure 8(b)):

$$\beta_w^-(a) = \sigma(p_1^-, p_2^-) \tag{5}$$

$$\beta_w^+(a) = \sigma(p_1^+, p_2^+) \tag{6}$$

Let a_1, a_2 be arcs in A. If a_1 intersects one of the two lateral segments of a_2 or vice-versa then we say that (a_1, a_2) is an *incompatible* pair. Let's define a boolean function $\psi_w : A \times A \to \{\texttt{false}, \texttt{true}\}$ such that $\psi_w(a_1, a_2) = \texttt{true}$ if (a_1, a_2) is incompatible and $\psi_w(a_1, a_2) = \texttt{false}$ otherwise. Thus we have

$$\begin{aligned} \psi_w(a_1, a_2) = \quad & \sigma(a_1) \cap \beta_w^-(a_2) \neq \emptyset \quad \vee \quad \sigma(a_1) \cap \beta_w^+(a_2) \neq \emptyset \quad \vee \\ & \sigma(a_2) \cap \beta_w^-(a_1) \neq \emptyset \quad \vee \quad \sigma(a_2) \cap \beta_w^+(a_1) \neq \emptyset \end{aligned} \tag{7}$$

where $\vee$ denotes the "or" operator. Let I_w be the set of incompatible pairs of arcs in A. For every $(a_1, a_2) \in I_w$ let

- $E(a_1, a_2)$ be the four end-points of a_1 and a_2:

$$E(a_1, a_2) = \{p_1(a_1), p_2(a_1), p_1(a_2), p_2(a_2)\}$$

- $P_{\min}(a_1, a_2)$ be the set of points minimizing ϕ in $E(a_1, a_2)$:

$$P_{\min}(a_1, a_2) = \{p \in E(a_1, a_2) \ : \ \phi(p) = \phi_0\}, \qquad \phi_0 = \min_{p \in E(a_1, a_2)} \phi(p) \tag{8}$$

If ϕ takes different values on the elements of $E(a_1, a_2)$, then $P_{\min}(a_1, a_2)$ is a singleton. Let $\tilde{P}_w(V, A)$ be the union of all these minimum-achieving points over all pairs of incompatible arcs:

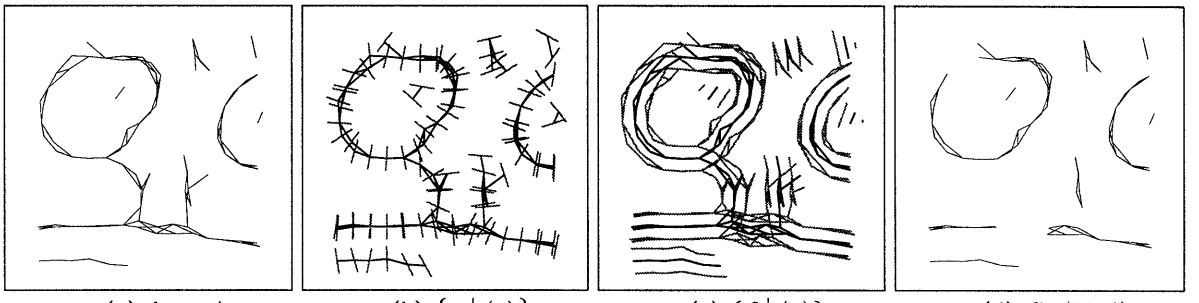
$$\tilde{P}_w(V, A) = \bigcup_{(a_1, a_2) \in I_w} P_{\min}(a_1, a_2) \tag{9}$$

The set of vertices of the computed graph is $P \setminus \tilde{P}_w(V, A)$ and

$$S^w(V, A) = \left\{(p_1, p_2) \in A \ : \ p_1, p_2 \notin \tilde{P}_w(V, A)\right\} . \tag{10}$$

1 For every $(a_1, a_2) \in A \times A$
2 compute $\psi_w(a_1, a_2)$, as given by equation (7)
3 For every $a_1, a_2 \in A$ such that $\psi_w(a_1, a_2) = \texttt{true}$
4 compute $P_{\min}(a_1, a_2)$ as given by equation (8)
5 $\tilde{P}_w(V, A) = \cup_{(a_1,a_2)\in I_w} P_{\min}(a_1, a_2)$
6 Return $A - \tilde{P}_w(V, A)$

Table 1: The stabilization algorithm. Steps 2 and 4 need not be carried out over all pairs of arcs. In fact, for each arc, it is enough to consider all the arcs in a fixed neighborhood around its midpoint. If we further assume that the density of arcs in the image is upper bounded, then the complexity of the algorithm is linear in the number of arcs.



(a) Arcs A (b) $\{\sigma_w^{\perp}(p)\}$ (c) $\{\beta_w^{\pm}(a)\}$ (d) $S^w(V, A)$

Figure 9: Stabilization of the graph in (a). (b): The segments $\sigma_w^{\perp}(p)$, $p \in P$. (c): The lateral boundaries. (d): The result of the stabilization algorithm.

By using the following notation

$$A - P' := \{(p_1, p_2) \in A \ : \ p_1, p_2 \notin P'\} \tag{11}$$

for any $P' \subset P$, equation (10) can be rewritten as $S^w(V, A) = A - \tilde{P}_w(V, A)$. The proofs of the following theorems are in the Appendix. The result of the stabilization algorithm on the graph of Figure 9(a) is shown in Figure 9(c).

Theorem 1 *For any vector graph (P, V, A) and $w > 0$, the graph with arcs $S^w(V, A)$ generated by the stabilization algorithm is w-stable with attraction basins $U_w(\pi)$.*

Let $l^{\max}(A)$ be the maximum arc length of the graph (P, A),

$$l^{\max}(A) = \max\{||p_1 - p_2|| \ : \ (p_1, p_2) \in A\}$$

Theorem 2 *Let Γ be a set of curves with bounded curvature and let (P, V, A) be a projection of Γ with admissible deviations δ_0, δ_1, δ_2 and Θ_1. If*

$$\frac{2\delta_1}{\cos\Theta_1} < w \cdot (1 - \epsilon_1(\kappa)), \tag{12}$$

$$\delta_2 - \delta_1 > \max\{w, l^{\max}(A)\} \cdot (1 + \epsilon_2(\kappa)) \tag{13}$$

then the graph $S^w(V, A)$ covers Γ with distance δ_0. Namely, for every $\gamma \in \Gamma$ $S^w(V, A)$ contains a path π such that $d(\gamma; \sigma(\pi)) < \delta_0$.

In Theorem 2, κ denotes the maximum curvature of the curves Γ and $\epsilon_1(\kappa)$, $\epsilon_2(\kappa)$ are positive functions such that $\epsilon_1(0) = \epsilon_2(0) = 0$. As a corollary of Theorems 1 and 2, notice that the vector graph with arcs $S^w(V, A)$ is a stable projection of Γ.

5 Invalid end-points and strong stability

In a stable graph, the tracked path is guaranteed to remain close to the true curve if this path contains a point sufficiently close to the curve. Thus, errors such as those in Figure 5(a) can not occur in a stable graph. However, it is not guaranteed that all the paths near the curve are long enough to cover the curve completely. Therefore, the tracked path might terminate before reaching the end of the curve (see Figure 10).

To avoid this problem, invalid end-points such as p_3 in Figure 10 are connected to some collateral path by adding an arc (e.g. (p_3, p_5)) to the graph. To describe how this is done, a few definitions are needed. First of all, let us assume for simplicity that the relationship

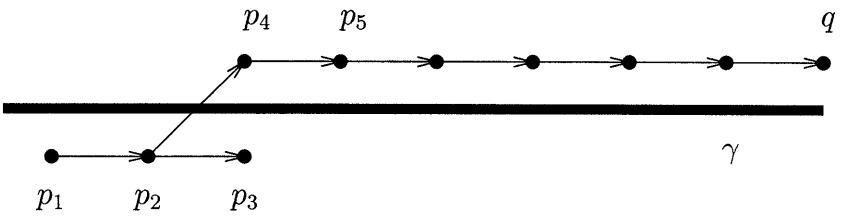


Figure 10: The path $(p_1, p_2, p_4, \ldots, q)$ covers γ whereas (p_1, p_2, p_3) does not (both paths are maximal). p_3 is said to be an "invalid" end-point. If during curve tracking p_3 is chosen instead of p_4 at the bifurcation point p_2 (this occurs if the most collinear arc with (p_1, p_2) is chosen), then the resulting path does not cover γ completely.

between the points in P and the other geometric entities is "generic", that is,

$$\sigma(p_1, p_2) \cap P = \{p_1, p_2\} \tag{14}$$

$$\sigma_w^{\perp}(p) \cap P = \{p\} \tag{15}$$

$$\beta_w(\pi) \cap P = \{p_1, p_n\} \tag{16}$$

where $\pi = (p_1, \ldots, p_n)$ and, as defined before, $\sigma_w^{\perp}(p) = \sigma(p^-, p^+)$. Also, without loss of generality, let us assume that (P, A) does not contain *isolated* vertices, namely every vertex in the graph (P, A) is connected to at least one other vertex. Let A_p^{in} and A_p^{out} be the set of in-arcs and out-arcs from p, namely $A_p^{\text{in}} = \{a \in A \,:\, p_2(a) = p\}$, $A_p^{\text{out}} = \{a \in A \,:\, p_1(a) = p\}$. A path $\pi = (p_1, \ldots, p_n)$ is said to be *maximal in* A if $A_{p_1}^{\text{in}} = A_{p_n}^{\text{out}} = \emptyset$. Let (see Figure 11(a))

$$\dot{\sigma}_w^{\perp}(p) = \sigma_w^{\perp}(p) \setminus \{p\}$$

Definition 3 *A point $p \in P$ is an* end-point *if $A_p^{\text{out}} = \emptyset$ or $A_p^{\text{in}} = \emptyset$. An end-point is* invalid *if $\sigma(p_1, p_2) \cap \dot{\sigma}_w^{\perp}(p) \neq \emptyset$ for some $(p_1, p_2) \in A$.*

Notice that if π is a path terminating at an invalid end-point p, then there might be a collateral path of π which "prolongs" it. This longer path contains the arc (p_1, p_2) such that $\sigma(p_1, p_2) \cap \dot{\sigma}_w^{\perp}(p) \neq \emptyset$.

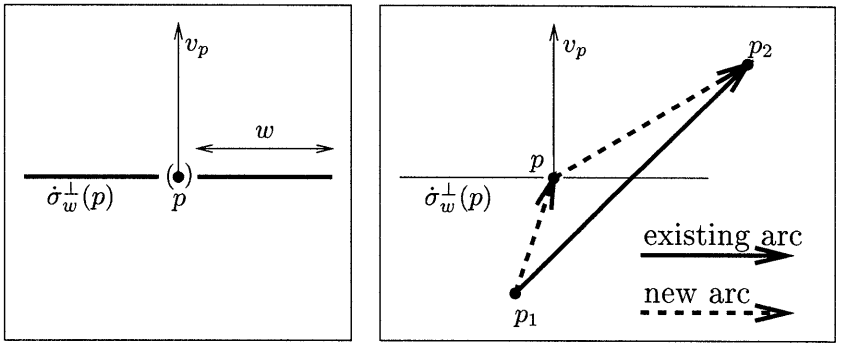


Figure 11: (a): Definition of $\dot{\sigma}_w^{\perp}(p)$. (b): To deal with invalid end-points, p_1 is connected to p and p to p_2 whenever $\sigma(p_1, p_2)$ intersects $\dot{\sigma}_w^{\perp}(p)$.

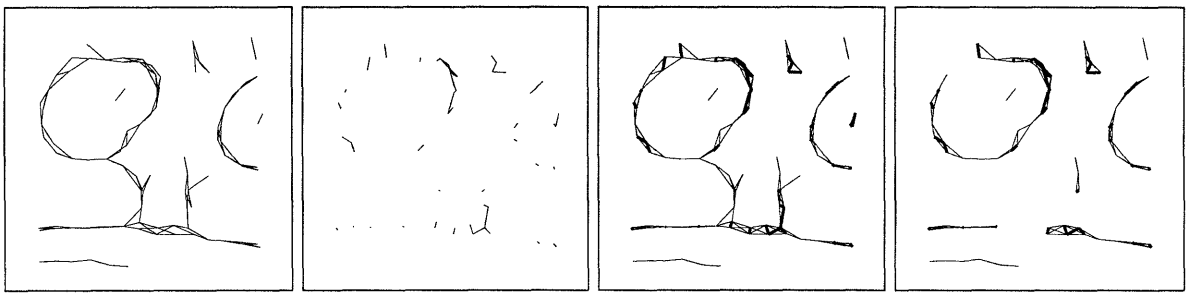
To ensure that tracking does not get stuck at an invalid-end point, a new graph with arcs $\overline{A} \supset A$ is constructed as follows. Initially, let $\overline{A} = A$. Then, for any p such that $\sigma(p_1, p_2) \cap \dot{\sigma}_w^{\perp}(p) \neq \emptyset$, $(p_1, p_2) \in A$, add the arcs (p_1, p) and (p, p_2) to $\overline{A}$. The new graph $\overline{A}$ obtained from the graph in Figure 12(a) is shown in Figure 12(c). By stabilizing the graph with arcs $\overline{A}$ one obtains a graph which possesses the following strong stability property. Let $A, \hat{A}$ be sets of arcs.

Definition 4 *A path π in $\hat{A}$ is a* strong attractor with respect to A *if there is a neighborhood U of $\sigma(\pi)$, $U \subset N_w(\pi)$, such that $\sigma(\pi') \cap U \neq \emptyset$ implies $\sigma(\pi') \subset \overline{U}$ for every path π' in $\hat{A} \cap A$. $\hat{A}$ is* strongly w-stable with respect to A *if every maximal path in $\hat{A}$ is a strong attractor with respect to A.*

Here, $\overline{U}$ denotes the closure of U.

Theorem 3 *For any vector graph (P, V, A) and $w > 0$, the graph with arcs $S^w(V, \overline{A})$ is strongly w-stable with respect to A with attraction basins $U_w(\pi)$.*

A result similar to Theorem 2 holds also for $S^w(V, \overline{A})$. The only difference is that the lower bound on $\delta_2 - \delta_1$ is now proportional to $w + l^{\max}(A)$.



(a) Arcs A (b) New arcs (c) $\overline{A}$ (d) $S^w(V,\overline{A})$

Figure 12: Computation and stabilization of $\overline{A}$. (a): Initial graph. (b): The arcs needed to connect invalid end-points to a collateral path. (c): The graph $\overline{A}$ with the new arcs shown in bold. (d): The graph obtained by stabilizing $\overline{A}$. This graph is strongly stable.

Theorem 4 *Let Γ be a set of curves with bounded curvature and (P, V, A) a projection of Γ with admissible deviations δ_0, δ_1, δ_2 and Θ_1. If*

$$\frac{2\delta_1}{\cos\Theta_1} < w \cdot (1 - \epsilon_1(\kappa)), \tag{17}$$

$$\delta_2 - \delta_1 > (w + l^{\max}(A)) \cdot (1 + \epsilon_2(\kappa)) \tag{18}$$

then, the graph $S^w(V,\overline{A}) \cap A$ covers Γ. Namely, for every $\gamma \in \Gamma$, $S^w(V,\overline{A}) \cap A$ contains a path π such that $d(\gamma; \sigma(\pi)) < \delta_0$.

Notice that, as a corollary of Theorems 3 and 4, $S^w(V,\overline{A}) \cap A$ is a strongly stable projection of Γ.

6 Computing a path covering

In the previous sections a strongly stable projection of Γ with arcs $S^w(V,\overline{A})$ was constructed. This section addresses the problem of computing a family of disjoint paths $\{\pi_1, \ldots, \pi_N\}$ in $S^w(V,\overline{A})$ which covers Γ. It is assumed for now that $S^w(V,\overline{A})$ does not contain any cycles. The more general case where cycle can be present is treated in the next section.

1	$A_1 = S^w(V, \overline{A})$
2	$j = 1$
3	Do until $A_j = \emptyset$
4	pick a source s_j in A_j
5	$\pi_j = maximalPath(s_j, A_j)$
6	$Q_j = P_{A_j} \cap U_w(\pi_j)$
7	$A_{j+1} = A_j - Q_j$
8	$j = j + 1$
9	end do

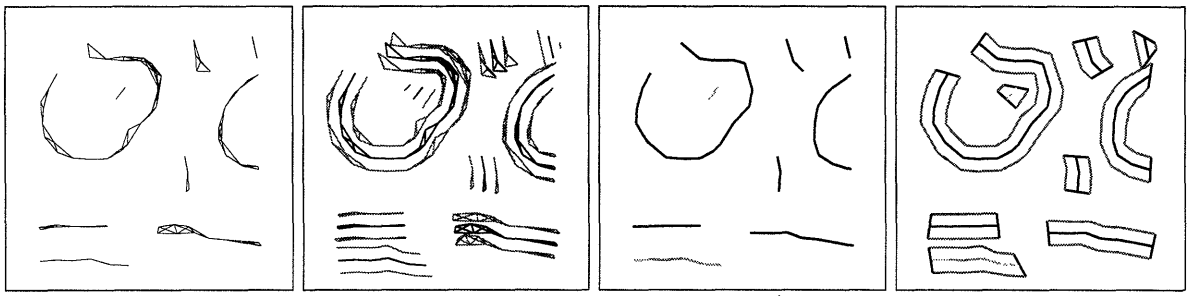
Table 2: The algorithm to compute regular curves from $S^w(V, \overline{A})$. A *source* in A_j is a vertex with no in-arcs. In line 6, P_{A_j} denotes the set of vertices of the graph A_j. The procedure $maximalPath(s_j, A_j)$ returns a maximal path in A_j with starting point s_j.

Recall that the length of an arc in A is less or equal to $l^{\max}(A)$ whereas arcs in $S^w(V, \overline{A})$ have lengths less or equal to $l^{\max}(A) + w$. Thus, elements in $S^w(V, \overline{A}) \cap A$ will be called *short arcs* and the other elements of $S^w(V, \overline{A})$ will be called *long arcs*. Notice that the subgraph of short arcs is sufficient to cover Γ. Long arcs have been added to ensure that all maximal paths are sufficiently extended to cover the tracked curve.

The paths $\pi_1, \ldots, \pi_N$ are computed one at a time by an iterative procedure. This procedure extracts a maximal path π_j from the current search graph A_j and then defines the new search graph $A_{j+1} \subset A_j$ by eliminating all arcs with a vertex in the neighborhood $U_w(\pi_j)$. This is continued until the search graph is empty. Existence of a maximal path in A is guaranteed by the assumption that the graph does not contain cycles. The algorithm is shown in Table 2.

Since $U_w(\pi_j)$ is a neighborhood of $\sigma(\pi_j)$, all the vertices of π_j belong to $U_w(\pi_j)$, except for the two end-points (which belong to the boundary of $U_w(\pi_j)$). Thus, if π_j has at least three vertices, then no arc of π_j belongs to A_{j+1} because every arc of π_j has at least one vertex in $U_w(\pi_j)$ (see lines 6 and 7 of Table 2). That is, we have

$$A_{\pi_j} \cap A_{j+1} = \emptyset \tag{19}$$



(a) $S^w(V,\overline{A})$ (b) $\{\beta_w^{\pm}(a)\}$ (c) $\hat{\Gamma}$ (d) $\{U_w(\pi_j)\}$

Figure 13: (a): Computation of regular curves from the strongly stable graph $S^w(V,\overline{A})$. (b): The lateral boundaries formed by the lateral segments $\{\beta_w^{\pm}(a)\}$. (c): The regular curves computed by the algorithm. (d): The neighborhoods $\{U_w(\pi_j)\}$.

where A_{π_j} denotes the arcs belonging the the path π_j. If π_j has just two vertices, then one needs to modify slightly the algorithm shown in Table 2 so that (19) is still true. We omit these details for the sake of simplicity. From (19) (and $A_{\pi_j} \subset A_j$) we have that A_{j+1} is a proper subset of A_j. Since A_1 is finite, this implies that the algorithm terminates after a finite number of steps. Moreover, it implies that the paths π_j are arc-disjoint, that is,

$$A_{\pi_j} \cap A_{\pi_k} = \emptyset, \qquad j \neq k \tag{20}$$

Let $\hat{\Gamma}$ be the set of polygonal curves $\sigma(\pi_j)$: $\hat{\Gamma} = \{\sigma(\pi_j) \ : \ 1 \leq j \leq N\}$.

Theorem 5 *Let Γ be a set of curves with bounded curvature and let (P,V,A) be projection of Γ with admissible deviations δ_0, δ_1, δ_2 and Θ_1. If*

$$\frac{2\delta_1}{\cos\Theta_1} < w \cdot (1 - \epsilon_1(\kappa)), \tag{21}$$

$$\delta_2 - \delta_1 > (w + l^{\max}(A)) \cdot (1 + \epsilon_2(\kappa)) \tag{22}$$

then for every $\gamma \in \Gamma$ there exists $\hat{\gamma} \in \hat{\Gamma}$ such that $d(\gamma;\hat{\gamma}) < w + \delta_0$.

Notice that, in the zero curvature case, and if w is set to the lower bound given by equation (21), then the localization error is

$$\delta_0 + 2\frac{\delta_1}{\cos\Theta_1} \ .$$

The parameters δ_0 and δ_1 represent two different types of localization uncertainty in the local data whereas Θ_1 is an upper bound on orientation uncertainty. It is not clear how tight this bound is. Probably, the factor 2 can be reduced but it is unlikely that it can be made smaller than 1.

The parameter δ_2, which represents the minimum separation distance between two curves, has to be at least $3\delta_1 + l^{\max}(A)$ (by letting $\kappa = \Theta_1 = 0$). The $l^{\max}(A)$ part can be made arbitrarily small by breaking down the arcs of the vector graph into smaller pieces. This entails a linear slowdown of the algorithm. As for the other term, $3\delta_1$, we do not know how tight it is.

6.1 Computing the optimal path

The algorithm in Table 2 contains the procedure $maximalPath(s_j, A_j)$ which returns a path $\pi_j \in M_j(s_j)$, where $M_j(s_j)$ denotes the set of maximal paths in A_j with first element s_j. Notice that Theorem 5 holds for any choice of a path in $M_j(s_j)$. Thus, this path can be determined according to an arbitrarily chosen cost function $c(\pi)$. If this cost is additive,

$$c(\pi) = \sum_{i=1}^{n} c(p_i), \qquad \pi = (p_1, \ldots, p_n)$$

then the minimizing path can be computed by using an efficient dynamic programming algorithm which runs in linear time in the number of arcs in the graph. In fact, let $c^\star(p)$, $p \in P_{A_j}$, be the optimal cost of a maximal path starting at p. Then, the following Bellman equation holds

$$c^\star(p) = c(p) + \min_{p' \in F(p)} c^\star(p'), \qquad p \in P_{A_j}$$

where $F(p)$ denotes the set of vertices p' such that $(p, p') \in A_j$. The recursive algorithm in Table 3 can be used to compute $c^\star(p)$ for all $p \in P_{A_j}$ (assuming the graph does not contain cycles). The optimal path starting from p is then

$$maximalPath(s_j, A_j) = (s_j, \nu(s_j), \nu(\nu(s_j)), \ldots, \nu^n(s_j)) \tag{23}$$

```
1    optimize(p)
2        if F(p) = ∅
3            c*(p) = c(p)
4        else
5            for every p' ∈ F(p)
6                optimize(p')
7            c*(p) = c(p) + min {c*(p') : p' ∈ F(p)}
8    return
```

Table 3: The recursive algorithm to compute optimal maximal paths.

where $\nu(p)$ is the minimizer of $c^\star(p')$, $p' \in F(p)$.

Notice that this method can be generalized to cost functions of the form

$$c(\pi) = \sum_{i=1}^{n-k} c\left(p_i, p_{i+1}, \ldots, p_{i+k}\right).$$

To do this, one needs to construct a graph whose nodes are all possible $(k+1)-$paths $(q_1, \ldots, q_k)$ in the original graph and whose arcs are all the pairs $((q_1, \ldots, q_k), (q_1', \ldots, q_k'))$ where $q_i' = q_{i+1}$, $i = 2, \ldots, k-1$ (see also (Casadei, 1995)).

This approach can be used to compute paths with minimum total turn. In fact, the total turn of a path $\pi = (p_1, \ldots, p_n)$ is given by

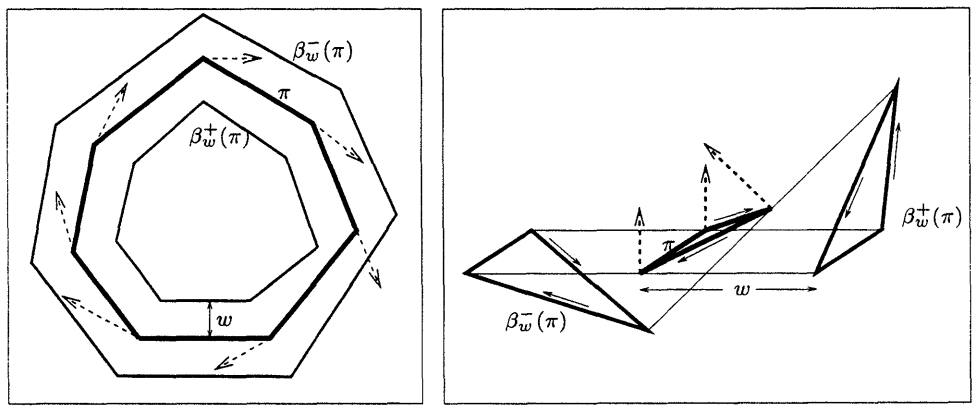
$$c(\pi) = \sum_{i=2}^{n-1} c\left(p_{i-1}, p_i, p_{i+1}\right)$$

where $c(p_{i-1}, p_i, p_{i+1})$ is the absolute value of the orientation differences between the arcs (p_{i-1}, p_i) and (p_i, p_{i+1}).

A dynamic programming approach for optimal curve detection was also proposed in (Subirana-Vilanova and Sung, 1992; Sha'ashua and Ullman, 1988). Their method minimizes a cost function which penalizes curvature and favors the total length of the curve. Our approach is simpler in that only curvature appears in the cost function. This simplification is possible because all the curves which can be constructed from a given point cover

each other, i.e. they have the same "length". This is a consequence of the strong stability of the graph. Due to this simplification, the optimal algorithm needs to pass through each point only once and therefore has linear time complexity.

7 Classification and detection of cycles



(a) Regular cycle (b) Irregular cycle (looplet)

Figure 14: (a): A Regular cycle. (b): A looplet.

The curve extraction procedure of Section 6 needs some adjustment to deal with cycles in the vector graph. Recall that a cycle is a path $\pi = (p_1, \ldots, p_n)$ such that $p_1 = p_n$. The structure of the stabilized graph $S^w(V, \overline{A})$ makes it possible to classify cycles into two classes, *regular* cycles and *irregular* cycles (or *looplet*). To do this, notice first of all that the two lateral boundaries $\beta_w^+(\pi)$ and $\beta_w^-(\pi)$ of a cycles are closed curves (see Figure 14). Since $S^w(V, \overline{A})$ is the output of the stabilization algorithm, the closed polygonal curve $\sigma(\pi)$ is disjoint from $\beta_w^+(\pi)$ and $\beta_w^-(\pi)$ (see Proposition 2 in the Appendix). Thus, two cases are possible. In the first case, $\sigma(\pi)$ encloses either $\beta_w^+(\pi)$ or $\beta_w^-(\pi)$. Then, the other lateral boundary encloses the other two closed curves (see Figure 14(a)). A cycle of this type is said to be *regular*. In the second case, neither $\beta_w^+(\pi)$ nor $\beta_w^-(\pi)$ encloses $\sigma(\pi)$ and the interiors of the three cycles are all disjoint (Figure 14(b)). A cycle of this kind will be called a *looplet*, or irregular cycle.

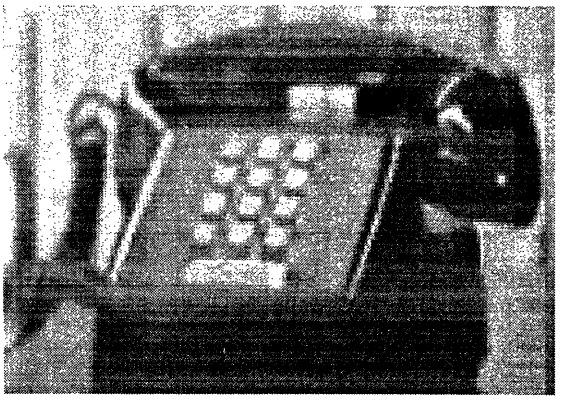
The procedure *optimize*(p) in Table 3 can be modified slightly so that cycles are detected and handled appropriately. Let *optimize'* be the modified procedure. Every point p is assigned a status variable which can take one of three possible values: "new" (which is the initial state), "active" and "done". When a point is opened for the first time, namely *optimize'* is called with argument p, then the status of p is changed from "new" to "active". Then, when the procedure *optimize'*(p) returns, the status of p is changed from "active" to "done". A cycle occurs whenever *optimize'* opens a point which is already "active".

To check whether the cycle is regular it is sufficient to pick one of its vertices and verify whether it is enclosed by either $\beta_w^+(\pi)$ or $\beta_w^-(\pi)$. If the cycle is regular then the path $maximalPath(s_j, A_j)$ in (23) is set equal to this cycle and lines 7,8 of the procedure in Table 2 are applied to it. If the cycle is a looplet then all its nodes are coalesced into a "super-node" and the procedure *optimize'* continues from this new super-node.

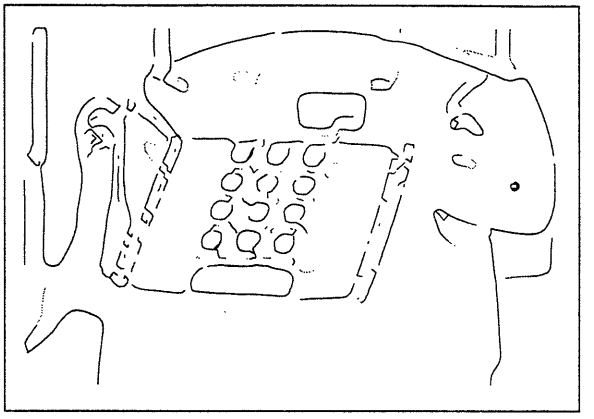
8 Experiments

The result of the algorithm on two images are shown in Figures 15 and 16. The vector graph was computed by using the method described in Section 2.1. Rectangular regions with width 4 and height 3 were used to compute the cubic polynomial fit and the tangent vectors. Some thresholds were used to eliminate some of these tangent vectors. The parameter w was set to 0.75 for all the experiments.

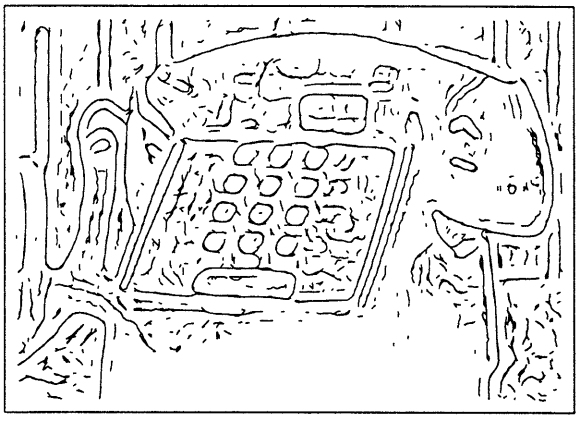
The results are compared to Canny's algorithm with sub-pixel accuracy followed by greedy edge linking, as implemented in (Perona, 1995). The performance of the two algorithms is comparable on edges which are isolated, with low curvature and with significant brightness change. However, when the topology of the edges is more complex, the advantages of a more robust approach become evident. In fact, greedy linking connects points with very little knowledge about the semi-local curvilinear structure. If an unstable bi-



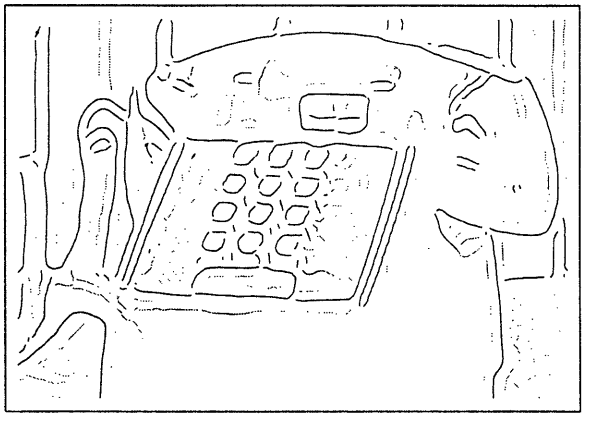
(a) Brightness image



(b) Canny's algorithm



(c) Vector graph

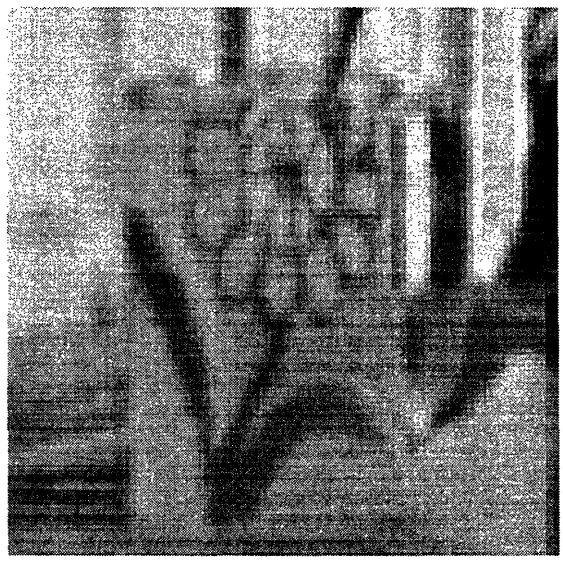


(d) Regular curves

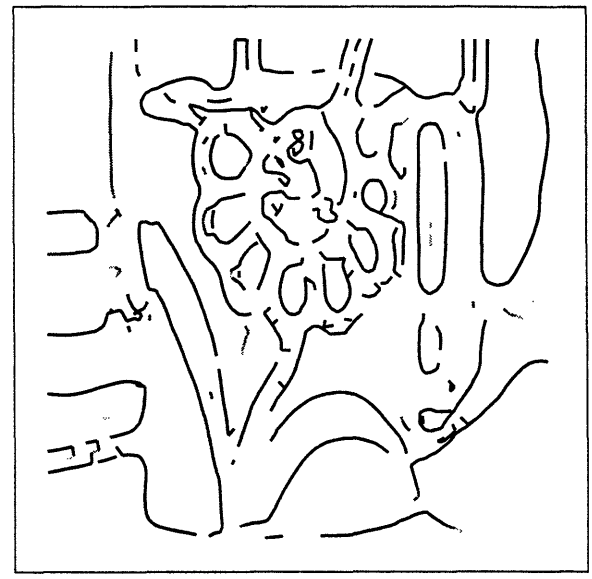
Figure 15: Telephone image. The final result is shown in (d).

furcation is present, then greedy linking makes a blind choice which may lead to a curve diverging from the true edge. For instance, the two parallel edges on the left of telephone keyboard are disconnected and merged together in 15(b). The contour of the telephone keys and of the flower petals are often confused with nearby edges. Notice that the contours obtained by the proposed algorithm, even though disconnected, remain close to the true edges and do not diverge from them.

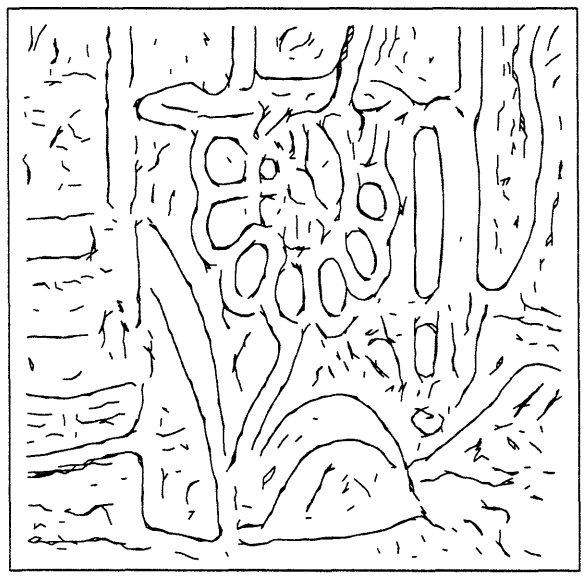
The limits of the proposed algorithm lie in the assumptions which are necessary for



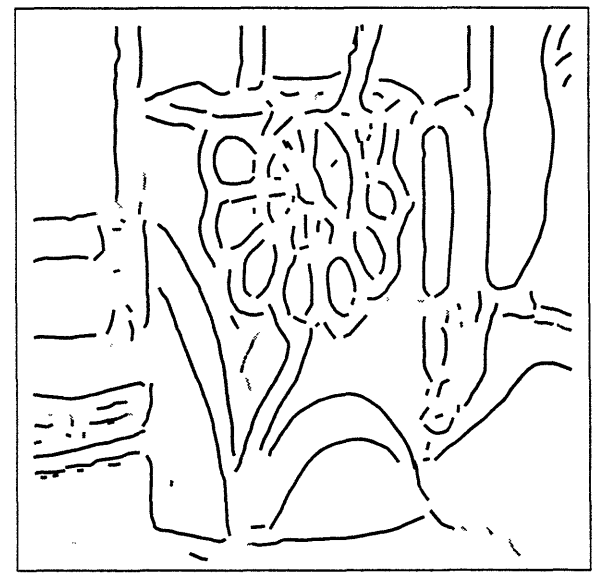
(a) Brightness image



(b) Canny's algorithm



(c) Vector graph



(d) Regular curves

Figure 16: Flower image. The final result is shown in (d).

a curve to be recovered without disconnections. If a curve in the image violates one of these assumptions, then the curve might be disconnected at the point where the violation occurs. For instance, notice that edges are broken at points of high curvature and at curve singularities such as T-junctions. Also, since the asymmetric Hausdorff distance is used to evaluate the match between the true curve and the reconstructed one, curve end-points may not be recovered correctly. Sometimes, curves are extended a little beyond the ideal end-point.

These limitations can be dealt with at higher levels of the edge detection hierarchy by using more powerful edge models. The regular curves extracted at the current level can be used to construct efficient feedback loops to activate the appropriate high level models. For instance, this method was used in (Casadei and Mitter, 1996) to bridge small gaps in the curves due to high curvature and T-junctions.

9 Conclusions

A hierarchical and compositional strategy, based on a hierarchy of contour models, can be useful to compute efficiently a global representation of the contours in an image. Computation in this hierarchy consists of two types of processes: bottom-up composition processes, whereby complex descriptors are constructed from simple ones, and top-down feedback loops which guide and constrain the search for relevant groupings to avoid combinatorial explosion of the search space. Having enough many intermediate levels in the hierarchy is necessary to ensure that enough top-down feedback is used so that uncertainties are neither resolved arbitrarily nor causing exponential computational complexity.

When applied to the problem of composing local edge information into curve descriptors, this requirement suggests that the curve dictionary used in the first composition stage should be simple enough to ensure that the elements in this dictionary can be computed efficiently and reliably. This leads to three natural assumptions on the initial curve mod-

els, namely the covering, decay and alignment conditions on the input vector graph. The corresponding detection problem, which consists in detecting the visible and bounded-curvature components of the contours, can be solved robustly in linear time by assuming upper bounds on the deviations from the ideal (noiseless) model. The analytical upper bound on the localization error is quite reasonable and vanish linearly when the deviations from the ideal model vanish.

A notion of stability is introduced and it is argued that instabilities due to uncertainties and ambiguities in the local edge information, such as multiple responses to the same edge, need to be addressed systematically at this stage.

9.1 Future work

The higher levels of the hierarchy, which are currently being developed on top of the representation described in this paper, use regular curves as vehicles to efficiently include global information into edge detection. To do this, compositional models of complex descriptors (such as partially invisible curves with singularities) are being constructed. These models relate complex descriptors to a set of simpler parts (such as regular visible curves) that are computed by the lower levels of the hierarchy. Every model has to provide *validation criterion* which relies on all the information at the lower levels to determine when enough evidence exist to hypothesize a high level descriptor.

A first step consists in recovering curve singularities (e.g. corners and junctions) and small curve gaps. It turns out that by using short-range semi-local information it is possible to recover correctly many of these features (Casadei and Mitter, 1996). The next higher dictionary is based on closed curves. The properties of the regions inside these curves provide a very powerful validation tool to be used as a feedback mechanism. Closed curves are computed by searching cycles in a *curve graph* ((Elder and Zucker, 1996) also propose to compute closed edges as cycles in a graph). Curve graphs, whose nodes are regular visible curves, are a high-level generalization of vector graphs. Initially,

only curves with nearby terminators are connected by an arc in the curve graph. At a later stage, long-range arcs can be inserted in the graph so that invisible curves can be represented. To control the number of long-range arcs and therefore computational complexity, information about the shape of the curves and feedback from the region-based validation mechanism will be used. At a later stage, edges will be represented as a set of regions ordered by depth, as proposed in (Nitzberg and Mumford, 1990). This will make possible to exploit occlusion and other higher dimensional constraints.

A Proofs

A.1 Proof of Theorem 1

The proofs of the main results will be preceded by some useful propositions.

Definition 5 *A vector graph (P, V, A) is said to be* arc-compatible *if $\psi_w(a_1, a_2) = \texttt{false}$ for every pair of arcs $(a_1, a_2) \in A \times A$. The set A will also be said to be arc-compatible.*

Proposition 1 *For any vector graph (P, V, A) and $w > 0$, $S^w(V, A)$ is arc-compatible.*

Proof. Let $\psi_w(a_1, a_2) = \texttt{true}$ for some $a_1, a_2 \in A$. Then, $\tilde{P}_w(V, A)$ contains at least one of the points in $E(a_1, a_2)$. Hence either $a_1 \notin A - \tilde{P}_w(V, A)$ or $a_1 \notin A - \tilde{P}_w(V, A)$ so that $S^w(V, A)$ can not contain both a_1 and a_2. □

Proposition 2 *Let A be arc-compatible. Then for any paths π, π' in A:*

$$\sigma(\pi) \cap \beta_w^-(\pi') = \sigma(\pi) \cap \beta_w^+(\pi') = \emptyset \tag{24}$$

Proof. Let A_π be the set of arcs in the path π. Notice that

$$\sigma(\pi) = \bigcup_{a \in A_\pi} \sigma(a), \qquad \beta_w^+(\pi') = \bigcup_{a' \in A_{\pi'}} \beta_w^+(a'), \qquad \beta_w^-(\pi') = \bigcup_{a' \in A_{\pi'}} \beta_w^-(a')$$

The result follows then from Proposition 1. □

Proposition 3 *Let (P, V, A) be a vector graph and $w > 0$. Then $U_w(\pi) \subset N_w(\pi)$ for any path π in the graph (P, A).*

Proof. The result follows directly from the construction of the boundary of $U_w(\pi)$ (see Figure 8(a)). □

Proof of Theorem 1 (page 21). Let $\pi = (p_1, \ldots, p_n)$ be a path in $S^w(V, A)$. From Proposition 3 we have $U_w(\pi) \subset N_w(\pi)$. We have to prove that $U_w(\pi)$ is an attraction basin for π (see Definition 2). Recall that the boundary of $U_w(\pi)$ is

$$\partial U_w(\pi) = \beta_w(\pi) = \beta_w^-(\pi) \cup \sigma_w^\perp(p_n) \cup \beta_w^+(\pi) \cup \sigma_w^\perp(p_1)$$

Let $q \in U_w(\pi)$ and let π' be a path such that $\sigma(\pi')$ contains q. From Proposition 2 it follows that

$$\sigma(\pi') \cap \beta_w^-(\pi) = \sigma(\pi') \cap \beta_w^+(\pi) = \emptyset$$

so that $\sigma(\pi')$ can intersect $\beta_w(\pi)$ only at its extremities, $\sigma_w^\perp(p_1)$ and $\sigma_w^\perp(p_n)$, which are contained in $\mathbf{B}_w(p_1) \cup \mathbf{B}_w(p_n)$:

$$(\sigma(\pi') \cap \beta_w(\pi)) \subset \left(\sigma_w^\perp(p_1) \cup \sigma_w^\perp(p_n)\right) \subset (\mathbf{B}_w(p_1) \cup \mathbf{B}_w(p_n))$$

Therefore, $\sigma(\pi')$ can intersect $\partial U_w(\pi)$ only inside $\mathbf{B}_w(p_1) \cup \mathbf{B}_w(p_n)$. Hence $\sigma_q(\pi')$ — namely the largest connected subcurve of $\sigma(\pi')$ disjoint from $\mathbf{B}_w(p_1) \cup \mathbf{B}_w(p_n)$ — is contained in $U_w(\pi)$. Thus, $U_w(\pi)$ is an attraction basin for π. □

A.2 Proof of Theorem 2

Lemma 1 *Let γ be an unbounded straight line and let (P, V, A) be a vector graph satisfying the alignment condition on γ. Let $a \in A$. If $\sigma(a) \subset D^1_\gamma$ and*

$$\frac{2\delta_1}{\cos\Theta_1} < w, \tag{25}$$

$$\delta_2 - \delta_1 > w \tag{26}$$

then $\beta^+_w(a) \subset D^2_\gamma \setminus D^1_\gamma$ and $\beta^-_w(a) \subset D^2_\gamma \setminus D^1_\gamma$.

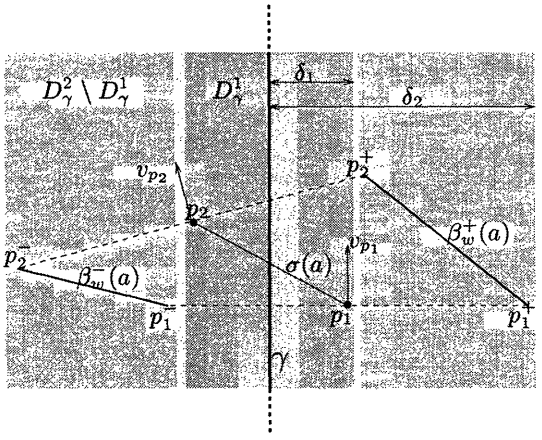


Figure 17: Proof of Lemma 1.

Proof. Let us assume without loss of generality that γ coincides with the y-axis. For any $p \in \mathbb{R}^2$ let $x(p)$ be the x-coordinate of p so that $d(p;\gamma) = |x(p)|$ (see figure 17). Let p_1^-, p_2^- be the end-points of $\beta^-_w(a)$ and p_1^+, p_2^+ the end-points of $\beta^+_w(a)$. Recall that

$$p_i^\pm = p_i \pm w u_\perp(p_i), \qquad i = 1, 2$$

where $u_\perp(p_i)$ is the unit vector perpendicular to v_{p_i}. Then,

$$x(p_i^\pm) = x(p_i) \pm w\cos\theta_i, \qquad i = 1, 2$$

where θ_i is the angle between v_{p_i} and γ. The alignment condition requires $\theta_i < \Theta_1$ so that

$$|x(p_i^{\pm}) - x(p_i)| \geq w \cos \Theta_1, \qquad i = 1, 2$$

Therefore, since $|x(p_i)| < \delta_1$ and by using (25),

$$|x(p_i^{\pm})| > w \cos \Theta_1 - |x(p_i)| > w \cos \Theta_1 - \delta_1 > \delta_1$$

Also, by using (26),

$$|x(p_i^{\pm})| \leq |x(p_i)| + w < \delta_1 + w < \delta_2$$

Thus $p_i^{\pm} \in D_\gamma^2 \setminus D_\gamma^1$ and the result follows from the convexity of $D_\gamma^2 \setminus D_\gamma^1$. □

Proposition 4 *Let γ be an unbounded straight line. Let (P, V, A) satisfy the decay and alignment conditions on γ. Suppose equations (25) and (26) hold and $\delta_2 - \delta_1 > l^{\max}(A)$. Let $p \in P$. Then*

$$p \in D_\gamma^0 \implies p \notin \tilde{P}_w(V, A)$$

Proof. Let A_p be the set of all arcs in A which have p as one of its two end-points. Let $a = (p, \overline{p}) \in A_p$ and $a' = (q_1, q_2) \in A$. We must prove that either a and a' are compatible or $\phi(p) > \phi_0$, where

$$\phi_0 = \min \{p, \overline{p}, q_1, q_2\}$$

Let $B_\gamma^2 = D_\gamma^2 \setminus D_\gamma^1$. Since $||p - \overline{p}|| \leq l^{\max}(A) < \delta_2 - \delta_1$ and $d(p; \gamma) < \delta_1$ we have

$$d(\overline{p}; \gamma) < d(p; \gamma) + ||p - \overline{p}|| < \delta_1 + (\delta_2 - \delta_1) < \delta_2,$$

that is $\overline{p} \in D_\gamma^1 \cup B_\gamma^2$. If $\overline{p} \in B_\gamma^2$ then $\phi(p) > \phi_0$, because $p \in D_\gamma^0$ and $\phi(p) > \phi(\overline{p})$ from the decay condition. Let us assume then that $\overline{p} \in D_\gamma^1$. This implies that $\sigma(a) \subset D_\gamma^1$ and, from Lemma 1,

$$\beta_w^+(a) \cup \beta_w^-(a) \subset B_\gamma^2 \tag{27}$$

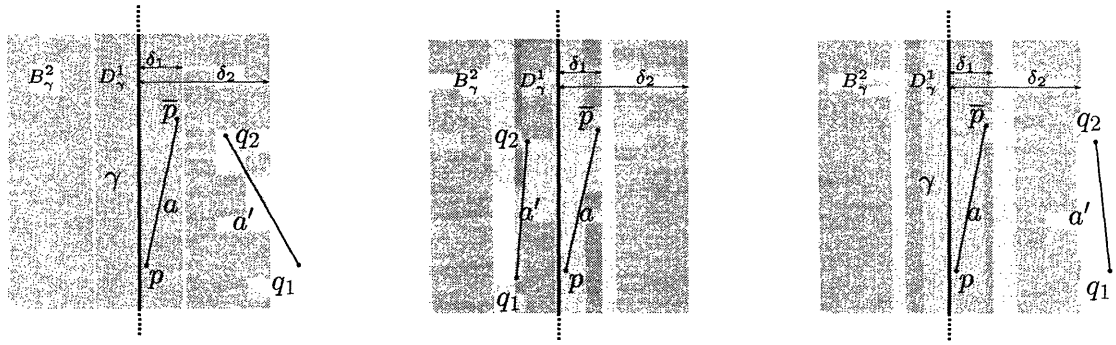


Figure 18: The three possible cases for a'.

The geometric relationship between $a' = (q_1, q_2)$ and γ can fall into one of three possible cases (see Figure 18):

(i) At least one of q_1, q_2 belongs to $B^2_\gamma = D^2_\gamma \setminus D^1_\gamma$.

(ii) $q_1, q_2 \in D^1_\gamma$.

(iii) $q_1, q_2 \in \mathbb{R}^2 \setminus D^2_\gamma$.

The case where one of q_1, q_2 is in D^1_γ and the other is in $\mathbb{R}^2 \setminus D^2_\gamma$ can not occur because $||q_1 - q_2|| \leq l^{\max}(A) < \delta_2 - \delta_1$.

Case (i). From the decay condition it follows $\phi(p) > \phi(q_1)$ or $\phi(p) > \phi(q_2)$ and therefore $\phi(p) > \phi_0$.

Case (ii). Notice that $\sigma(a') = \sigma(q_1, q_2) \subset D^1_\gamma$. From this and from equation (27) we have

$$\sigma(a') \cap \left(\left(\beta^-_w(a) \cup \beta^+_w(a)\right)\right) = \emptyset.$$

Similarly, from $\sigma(a) \subset D^1_\gamma$ and from Lemma 1 applied to a',

$$\sigma(a) \cap \left(\left(\beta^-_w(a') \cup \beta^+_w(a')\right)\right) = \emptyset.$$

Hence $\psi_w(a, a') = \texttt{false}$ and (a, a') is a compatible pair.

Case (iii). Since $\delta_2 - \delta_1 > w$, it follows that

$$D^1_\gamma \cap \left(\left(\beta^-_w(a') \cup \beta^+_w(a')\right)\right) = \emptyset$$

and therefore, from $\sigma(a) \subset D^1_\gamma$,

$$\sigma(a) \cap \left(\left(\beta^-_w(a') \cup \beta^+_w(a')\right)\right) = \emptyset.$$

Since $\sigma(a') \subset B^3_\gamma$, from (27) we have

$$\sigma(a') \cap \left(\left(\beta^-_w(a) \cup \beta^+_w(a)\right)\right) = \emptyset$$

and therefore $\psi_w(a, a') = \texttt{false}$. □

Proof of Theorem 2 (page 22). Let $\gamma \in \Gamma$ and let us assume that γ is an unbounded straight line (The proof is given for this case only[1]). Since (P, V, A) is a projection of Γ, $\gamma \in \Gamma$ satisfies the covering, decay and alignment conditions. From the covering condition, there exists a path π in A such that $d(\gamma; \sigma(\pi)) < \delta_0$. Notice that the vertices of π belong to D^0_γ so that, from Proposition 4, they do not belong to $\tilde{P}_w(V, A)$. Therefore, π is also a path in $S^w(V, A) = A - \tilde{P}_w(V, A)$.

A.3 Proofs of Theorems 3 and 4

Definition 6 *Let A, $\hat{A}$ be sets of arcs and let $\hat{P}$ be the set of vertices of $\hat{A}$. The set $\hat{A}$ is said to be* $\sigma^\perp_w$-connected with respect to A *if*

$$\sigma(p_1, p_2) \cap \dot{\sigma}^\perp_w(p) \neq \emptyset \implies (p_1, p) \in \hat{A},\ (p, p_2) \in \hat{A} \tag{28}$$

for every $(p_1, p_2) \in \hat{A} \cap A$ and every $p \in \hat{P}$.

[1]The proof for finite curves requires a somewhat complicated generalization of the decay and alignment conditions to constrain the vector field in the vicinity of the curve end-points. The generalization to curves with bounded curvature is trivial, given the arbitrariness of the functions $\epsilon_1(\kappa)$ and $\epsilon_2(\kappa)$.

Proposition 5 *The set $\overline{A}$ constructed in Section 5 is $\sigma_w^{\perp}$-connected with respect to A.*

Proof. The result follows directly from the definition of $\overline{A}$. □

The following Lemma ensures that a graph remains $\sigma_w^{\perp}$-connected if an arbitrary set of vertices is suppressed.

Lemma 2 *Let A, $\hat{A}$ be sets of arcs whose vertices belong to P. Let $P' \subset P$. If $\hat{A}$ is $\sigma_w^{\perp}$-connected with respect to A then $\hat{A} - P'$ is also $\sigma_w^{\perp}$-connected with respect to A.*

Proof. Let p be a vertex in $\hat{A} - P'$ and (p_1, p_2) an arc in $(\hat{A} - P') \cap A$ such that $\sigma(p_1, p_2) \cap \dot{\sigma}_w^{\perp}(p) \neq \emptyset$. We have to prove that $(p_1, p) \in \hat{A} - P'$ and $(p, p_2) \in \hat{A} - P'$. Notice that since p is a vertex in $\hat{A} - P'$ it is also a vertex in $\hat{A}$. Also, from $(p_1, p_2) \in (\hat{A} - P') \cap A$ we have $(p_1, p_2) \in A$. Therefore, since $\hat{A}$ is $\sigma_w^{\perp}$-connected w.r.t. A,

$$(p_1, p) \in \hat{A}, \qquad (p, p_2) \in \hat{A} \tag{29}$$

Since p is a vertex in $\hat{A} - P'$ we have $p \notin P'$. Also, from $(p_1, p_2) \in (\hat{A} - P') \cap A$ we have $p_1, p_2 \notin P'$. Hence, from (29) it follows that $(p_1, p) \in \hat{A} - P'$ and $(p, p_2) \in \hat{A} - P'$. □

Proposition 6 *For any vector graph (P, V, A), $S^w(V, \overline{A})$ is $\sigma_w^{\perp}$-connected with respect to A.*

Proof. Notice that $S^w(V, \overline{A}) = \overline{A} - \tilde{P}_w(V, \overline{A})$. Therefore, since $\overline{A}$ is $\sigma_w^{\perp}$-connected w.r.t. A, the result follows from Lemma 2. □

Proposition 7 *Let $\hat{A}$ be $\sigma_w^{\perp}$-connected with respect to A, and let $\pi = (p_1, \ldots, p_n)$ be a maximal path in $\hat{A}$. Then, for every $(q_1, q_2) \in \hat{A} \cap A$,*

$$\sigma(q_1, q_2) \cap \dot{\sigma}_w^{\perp}(p_1) = \sigma(q_1, q_2) \cap \dot{\sigma}_w^{\perp}(p_n) = \emptyset$$

Proof. For the purpose of contradiction, let $(q_1, q_2) \in \hat{A} \cap A$ be such that

$$\sigma(q_1, q_2) \cap \dot{\sigma}_w^{\perp}(q) \neq \emptyset \tag{30}$$

where q is either p_1 or p_n. Since $\hat{A}$ is $\sigma_w^{\perp}$-connected, we have that (q, q_2) and (q_1, q) belong to $\hat{A}$ and therefore q has at least one out-arc and one in-arc. This contradicts the fact that π is maximal in $\hat{A}$.

Proposition 8 *Let $\hat{A}$ be $\sigma_w^{\perp}$-connected with respect to A and arc-compatible.*

- *For every maximal path $\pi = (p_1, \ldots, p_n)$ in $\hat{A}$ and every path π' in $\hat{A} \cap A$,*

$$\beta_w(\pi) \cap \sigma(\pi') \subset \{p_1, p_n\} \tag{31}$$

- *$\hat{A}$ is strongly w-stable with respect to A with attraction basins $U_w(\pi)$.*

Proof. Let $\pi = (p_1, \ldots, p_n)$ be a maximal path in $\hat{A}$ and π' a path in $\hat{A} \cap A$. Since $\hat{A}$ is arc-compatible we have from Proposition 2,

$$\sigma(\pi') \cap \beta_w^-(\pi) = \sigma(\pi') \cap \beta_w^+(\pi) = \emptyset$$

Since $\hat{A}$ is $\sigma_w^{\perp}$-connected and π is maximal in $\hat{A}$ we have from Proposition 7

$$\sigma(\pi') \cap \dot{\sigma}_w^{\perp}(p_1) = \sigma(\pi') \cap \dot{\sigma}_w^{\perp}(p_n) = \emptyset$$

Thus, since $\sigma_w^{\perp}(p) = \dot{\sigma}_w^{\perp}(p) \cup \{p\}$,

$$\sigma(\pi') \cap \beta_w(\pi) = \sigma(\pi') \cap \left(\beta_w^-(\pi) \cup \sigma_w^{\perp}(p_n) \cup \beta_w^+(\pi) \cup \sigma_w^{\perp}(p_1)\right) \subset \{p_1, p_n\} \tag{32}$$

which proves the first part. Let now π' be a path in $\hat{A} \cap A$ such that $\sigma(\pi') \cap U_w(\pi) \neq \emptyset$. To prove that $\hat{A}$ is strongly w-stable, one has to show that $\sigma(\pi') \subset \overline{U}_w(\pi)$. From the

assumptions (14)-(16) and from $A^{\text{in}}_{p_1} = A^{\text{out}}_{p_n} = \emptyset$ we have that $\sigma(\pi')$ can not exit $\overline{U}_w(\pi)$ through the points p_1, p_n. This, together with 32, yields the result.

□

Proof of Theorem 3 (page 24). From Proposition 1, $S^w(V, \overline{A})$ is arc-compatible because it is the output of the stabilization algorithm on the vector graph $(P, V, \overline{A})$. Moreover, $S^w(V, \overline{A})$ is $\sigma^{\perp}_w$-connected from Proposition 6. The result then follows from Proposition 8.
□

Proof of Theorem 4 (page 25). The proof is similar the that of Theorem 2. Let $\gamma \in \Gamma$ and, as in the proof of Theorem 2, let us assume that γ is an unbounded straight line. Since (P, V, A) is a projection of Γ, γ satisfies the covering, decay and alignment conditions. From the covering condition, there exists a path π in A such that $d(\gamma; \sigma(\pi)) < \delta_0$. Notice that the length of the segments added to A to construct $\overline{A}$ is at most $w + l^{\max}(A)$. Therefore, $l^{\max}(\overline{A}) \leq w + l^{\max}(A)$. Since the vertices of π belong to D^0_γ, by using Proposition 4 with A replaced by $\overline{A}$, we have that none of the vertices of π belongs to $\tilde{P}_w(V, \overline{A})$. Hence, since $S^w(V, \overline{A}) = \overline{A} - \tilde{P}_w(V, A)$ and $A \subset \overline{A}$, π is also a path in $S^w(V, \overline{A}) \cap A$. □

A.4 Proof of Theorem 5

Proposition 9 *A_j is strongly w-stable with attraction basins $U_w(\pi)$.*

Proof. Notice that

$$A_j = S^w(V, \overline{A}) - \left(\bigcup_{k<j} Q_k \right)$$

Thus, from Lemma 2 and Proposition 6, we have that A_j is $\sigma^{\perp}_w$-connected. Also, A_j is arc-compatible because it is a subset of $S^w(V, \overline{A})$ which is arc-compatible. The result then

follows from Proposition 8. □

Proposition 10 *Let π be a path in $S^w(V, \overline{A}) \cap A$. Then there exists a π_j such that $d(\sigma(\pi); \sigma(\pi_j)) \leq w$.*

Proof. As a first step we construct a partition of $A_1 = S^w(V, \overline{A})$ into N sets B_j, $j = 1, \ldots, N$. These sets are given by:

$$B_j = \{(p_1, p_2) \in A_j \ : \ p_1 \in U_j \ \vee \ p_2 \in U_j\} \tag{33}$$

where $U_j = U_w(\pi_j)$. Notice that $B_j \subset A_j$. The following must be proven:

$$\bigcup_{j=1}^{N} B_j = S^w(V, \overline{A}) = A_1 \tag{34}$$

$$B_j \cap B_k = \emptyset, \qquad k > j \tag{35}$$

From the recursive definition of A_j (lines 6 and 7 of Table 2) we have

$$A_{j+1} = A_j - Q_j = A_j \setminus \{(p_1, p_2) \in A_j \ : \ p_1 \in U_j \ \vee \ p_2 \in U_j\}$$

and therefore

$$A_{j+1} = A_j \setminus B_j \tag{36}$$

From this it follows that $B_j \cap A_{j+1} = \emptyset$. Thus, if $k > j$ we have $B_j \cap B_k = \emptyset$ because $B_k \subset A_k \subset A_{j+1}$. This proves (35). Notice that by iterating (36) one obtains:

$$A_j = A_1 \setminus \bigcup_{k<j} B_k \tag{37}$$

Let N be the number of steps done by the procedure before terminating. That is, from line 3 of Table 2,

$$A_{N+1} = \emptyset$$

By using (37) one gets

$$A_{N+1} = A_1 \setminus \bigcup_{k \leq N} B_k = \emptyset$$

from which (34) follows. Notice that from (37) and (34) we have

$$A_j = A_1 \setminus \bigcup_{k<j} B_k = \bigcup_{k \geq j} B_k \tag{38}$$

Now let k be the smallest j such that B_j contains at least one arc of the path π:

$$A_\pi \cap B_j = \emptyset, \qquad j < k \tag{39}$$

$$A_\pi \cap B_k \neq \emptyset \tag{40}$$

where A_π denotes the arcs of π. From (34), (39) and (38) we have

$$A_\pi = \bigcup_{1 \leq j \leq N} (A_\pi \cap B_j) = \bigcup_{j \geq k} (A_\pi \cap B_j) \subset \bigcup_{j \geq k} B_j = A_k$$

Thus, since π is path in $A_1 \cap A$ by assumption, π is a path in $A_k \cap A$. From $A_\pi \cap B_k \neq \emptyset$ and from (33) we have $\sigma(\pi) \cap U_k \neq \emptyset$. Then, since (from Proposition 9) A_k is strongly w-stable, we have $\sigma(\pi) \subset \overline{U_k} \subset \overline{N}_w(\pi_k)$. From this it follows that $d(\sigma(\pi); \sigma(\pi_k)) \leq w$. □

Proof of Theorem 5. Let $\gamma \in \Gamma$. From Theorem 4 we have that there exists a path π in $S^w(V, \overline{A}) \cap A$ such that $d(\gamma; \sigma(\pi)) < \delta_0$. From Proposition 10 there exists a path π_j such that $d(\sigma(\pi); \sigma(\pi_j)) \leq w$. Then, by the triangular inequality of the asymmetric Hausdorff distance one gets $d(\gamma; \sigma(\pi_j)) < w + \delta_0$. □

References

Bienenstock, E. and Geman, S. (1995). Compositionality in neural systems. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks.* MIT Press.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698.

Casadei, S. (1995). Model based detection of curves in images. Technical Report LIDS-TH-2306, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

Casadei, S. and Mitter, S. K. (1996). A hierarchical approach to high resolution edge contour reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Deriche, R. and Blaszka, T. (1993). Recovering and characterizing image features using an efficient model based approach. In *Proceedings of the IEEE Computer Society conference on Computer Vision and Pattern Recognition.*

Dolan, J. and Riseman, E. (1992). Computing curvilinear structure by token-based grouping. In *Conference on Computer Vision and Pattern Recognition.*

Elder, J. H. and Zucker, S. W. (1996). Computing contour closure. *Proceedings of the Fourth European Conference on Computer Vision*, pages 399–411.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Hancock, E. and Kittler, J. (1990). Edge-labeling using dictionary-based relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:165–181.

Haralick, R. (1984). Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:58–68.

Kanizsa, G. (1979). *Organization in vision : essays on gestalt perception.* Praeger.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331.

Lowe, D. (1985). *Perceptual Organization and Visual Recognition.* Kluwer.

Marroquin, J., Mitter, S. K., and Poggio, T. (1987). Probabilistic solution of ill-posed problems in computational vision. *Journal of American Statistical Ass.*, 82(397):76–89.

Mohan, R. and Nevatia, R. (1992). Perceptual organization for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14.

Mumford, D. and Shah, J. (1989). Optimal approximations of piecewise smooth functions and associated variational problems. *Comm. in Pure and Appl. Math.*, 42:577–685.

Nitzberg, M. and Mumford, D. (1990). The 2.1-d sketch. In *Proceedings of the Third International Conference of Computer Vision*, pages 138–144.

Nitzberg, M., Mumford, D., and Shiota, T. (1993). *Filtering, segmentation, and depth*, volume 662 of *Lecture notes in computer science.* Springer-Verlag.

Parent, P. and Zucker, S. (1989). Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11.

Perona, P. (1995). Lecture notes, Caltech.

Perona, P. and Malik, J. (1990). Detecting and localizing edges composed of steps, peaks and roofs. In *Proceedings of the Third International Conference of Computer Vision*, pages 52–57, Osaka.

Richardson, T. J. and Mitter, S. K. (1994). Approximation, computation, and distorsion in the variational formulation. In ter Haar Romeny, B., editor, *Geometry-Driven Diffusion in Computer Vision.* Kluwer.

Rohr, K. (1992). Recognizing corners by fitting parametric models. *International Journal of Computer Vision*, 9:3.

Rosenthaler, L., Heitger, F., Kubler, O., and von der Heydt, R. (1992). Detection of general edges and keypoints. In *Proceedings of the Second European Conference on Computer Vision.*

Sarkar, S. and Boyer, K. (1993). Perceptual organization in computer vision: A review and a proposal for a classificatory structure. *IEEE Trans. Systems, Man, and Cybernetics*, 23:382–399.

Sha'ashua, A. and Ullman, S. (1988). Structural saliency: the detection of globally salient structures using a locally connected network. In *Proceedings of the Second International Conference of Computer Vision*, pages 321–327.

Subirana-Vilanova, J. and Sung, K. (1992). Perceptual organization without edges. In *Image Understanding Workshop.*

Tannenbaum, A. (1995). Three snippets about curve evolution. Preprint.

Zhu, S., Lee, T., and Yuille, A. (1995). Region competition: unifying snakes, region growing, and bayes/mdl for multi-band image segmentation. Technical Report CICS-P-454, Center for Intelligent Control Systems.