MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1266 December 1991

Finding Junctions Using the Image Gradient

David J. Beymer

Abstract

Junctions, the intersection points of three or more intensity surfaces in an image, are useful low-level features in machine vision, with applications in recognition, motion, grouping, and 3D line interpretation. The popular edge detectors in use today, however, such as the Laplacian of the Gaussian and the second directional derivative, fragment edges at junctions, leaving these important features undetected. This paper analyzes why edges are fragmented at junctions by differential edge operators and proposes a method for detecting junctions based on this analysis. The analysis of edge fragmentation focuses on the properties of the gradient and zero crossings of the Laplacian and the second directional derivative operators. Fragmentation is caused by the intrinsic pairing of zero crossings at junctions and by a destructive interference of edge gradient vectors which increases sensitivity to noise and quantization. We propose a junction detector that works by filling in gaps at junctions in edge maps. It uses the image gradient to guide extensions of disconnected edges at junctions. A new representation for the gradient, the bow tie map, is used to implement the endpoint growing rules, which include following gradient ridges and using saddle points in the gradient magnitude. We demonstrate the junction detector on real imagery. Finally, the paper discusses previous approaches to junction detection.

Copyright © Massachusetts Institute of Technology, 1991

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence Research is provided in part by grant S1-801534-2 from Hughes Aircraft Company and by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124. The author was supported by a graduate fellowship from the Office of Naval Research while performing this research.

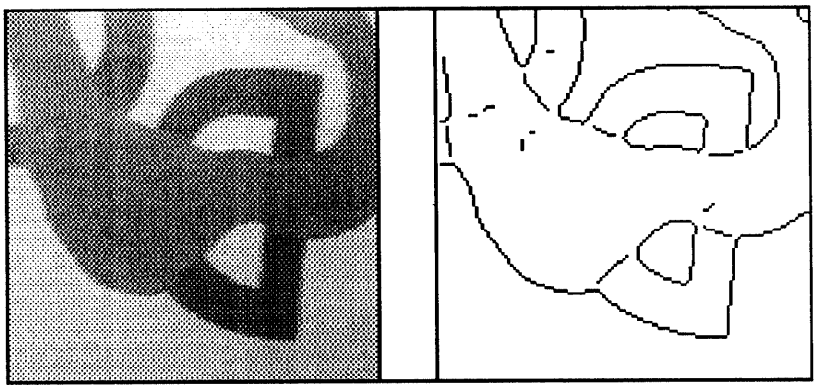


Figure 1: An image and its Canny edges. A threshold on the gradient magnitude cleaned up some of the edges due to noise. Edges are fragmented at the T junctions.

1 Introduction

One of the goals of low level computer vision is to provide stable image features for higher level analysis. Most work to date on feature detection has concentrated on finding edges, discontinuities in the image intensity surface. Working with edges simplifies the image considerably; edges correspond to such important image features as object boundaries, color changes, and illumination changes. A feature related to edges is the junction, which we define as the intersection of three or more intensity surfaces in the image. Junctions are usually located in images by looking for points of intersection of three or more intensity edges.

As point-based features, junctions have many useful applications in higher level computer vision. Model-based recognition (Bolles and Cain [6], Lamdan and Wolfson [23], Huttenlocher [19], Tucker [38]) and motion (Ullman [39]) have both used point-based features to drive their matching algorithms: recognition matches model to image and motion matches features between frames in time. Edge labelling schemes (Guzman [15], Huffman [18], Clowes [11], Waltz [40], Mackworth [27], Kanade [20], Chakravarty [9], Lee, *et al.* [25], Malik [28]) use junctions to build a three dimensional interpretation of an edge map. Finally, grouping, roughly defined as aggregating features coming from one object, can use junctions to group edges (Lowe [26], Beymer [4]).

This paper focuses on the problem of finding junctions in images. Detecting junctions is an open issue because gradient-based edge detectors, the popular edge detectors in use today, fail to detect junctions. One edge of the junction will often be detached from the junction, leaving a small gap. Generally speaking, the gap is caused by an interaction between edge gradients at the junction – some edges interfere with the detection of others. This gradient interaction is an unavoidable consequence of smoothing the image, which is introduced by camera optics and purposefully used to filter out noise. Figure 1 shows an example image and corresponding Canny [7] edges. Note the fragmentation at junctions.

This paper has two primary goals: to investigate why edge maps are fragmented at junctions and to propose a new method for finding junctions. Our investigation into the failure of gradient-based edge detectors will focus on properties of the zero crossings of the Laplacian and the second directional derivative, two popular edge operators. Edge fragmentation at junctions, as we will see, has two primary causes, the pairing of edges into groups of two and a sensitivity to noise and quantization. We will use an analytical model of an n-ary junction to provide a concrete example for analysis.

The second major focus of this paper is a new junction detection algorithm that works from edge maps and the gradient, using the gradient to fill gaps at junctions. An analysis of the gradient magnitude near junctions will drive the workings of our detection algorithm. We propose a new tool for describing peaks and valleys in the gradient, the bow tie map. Our junction detection algorithm uses the bow tie map to extend the endpoints of disconnected edges at junctions. Endpoints are grown by following ridges or saddle points in the gradient magnitude.

This paper is organized as follows. Section two discusses the zero crossings of the Laplacian and the second directional derivative, explaining why gradient-based edge operators fail at junctions. Section three analyzes the gradient near junctions and introduces the bow tie map. Section four proposes a new junction detection algorithm and shows results of the algorithm applied to real imagery. Finally, we close with a discussion of past work on junction detection.

2 Zero Crossings and the Gradient near Junctions

In this section, we will use an analysis of zero crossings and the image gradient to see why gradient-based edge detectors fragment junction edges. Gradient-based edge detection schemes are those that define edges as being local maximum of a first derivative operator, or, equivalently, a zero crossing of a second derivative operator.[1] Since gradient-based techniques take derivatives, we also assume that the image is smoothed with a regularizing filter [37]. This smoothing step blurs the structure of junctions and causes interaction between edge gradients, derailing edge detection there. In this paper, the function $f(x, y)$ will be the smoothed image intensity function.

To begin with, our analysis will look at the zero crossings of two second derivative operators, the Laplacian, $\nabla^2 f$, and the second directional derivative, $\frac{\partial^2 f}{\partial n^2}$. (For a good introduction to these operators and their use in edge detection, see Rosenfeld and Kak [34] or Ballard and Brown [1].) Zero crossings of the Laplacian is the scheme advocated by Marr and Hildreth [29]. The second directional derivative, defined as the second derivative of $f(x, y)$ taken in the direction of the gradient, characterizes edge detectors such as Canny [7] and Haralick [16]. Nonmaximum suppression in Canny eliminates some zero crossings of $\frac{\partial^2 f}{\partial n^2}$, as we will discuss later with phantom edges.

To analyze the properties of zero crossings near junctions, we need a model for junctions. As a generalization of the step edge model for edges, we chose to model junctions as the intersection of three or more surfaces of constant intensity. Even though intensity surfaces in real images are not constant (due to such factors as noise, shading, and specular highlights), approximating the intensity surface as constant should be reasonable since we are interested in its behavior near a point. We will use this model in two ways. First, following Poggio and Torre [37], we can qualitatively analyze zero crossings near a junction by using transversality theory. Similarly, we can use simple reasoning about the gradient as a vector field to explore its properties.

The second use of our model is quantitative: we find analytic equations for the gradient ∇f, $\nabla^2 f$, and $\frac{\partial^2 f}{\partial n^2}$, and numerically estimate these equations to run experiments on junctions. The analytic model we use is a general n-ary junction, the intersection of n regions with intensities r_0 through r_{n-1}. Shown in figure 2(a) for $n = 3$, our model n-ary junction has n incident edges e_0 through e_{n-1}, where e_i bounds the regions r_{i-1} and r_i and is at an angle θ_i with respect to the positive x axis. Using this model we can analyze any junction composed of straight edges (see figure 2(b) for examples). We shall focus most of our attention on trihedral junctions ($n = 3$), since they are the most common in imagery and are the easiest to analyze. This model is similar to the corner and trihedral junction models developed respectively by Berzins [3] and De Micheli *et al.* [31].

The smoothed image intensity function, $f(x, y)$, is obtained analytically by convolving the piecewise constant surface with a 2D Gaussian with width σ. Overall, the relevant parameters are σ and r_i and θ_i for $0 \leq i < n$. Appendix A gives the equations for the partial derivatives of $f(x, y)$, from which $|\nabla f|$, $\nabla^2 f$, and $\frac{\partial^2 f}{\partial n^2}$ are easily derived. 2D arrays of values of these equations, which require

[1] The zeros of a second derivative operator can also correspond to a local minimum in a first derivative operator. We will discuss this later when we talk about phantom edges.

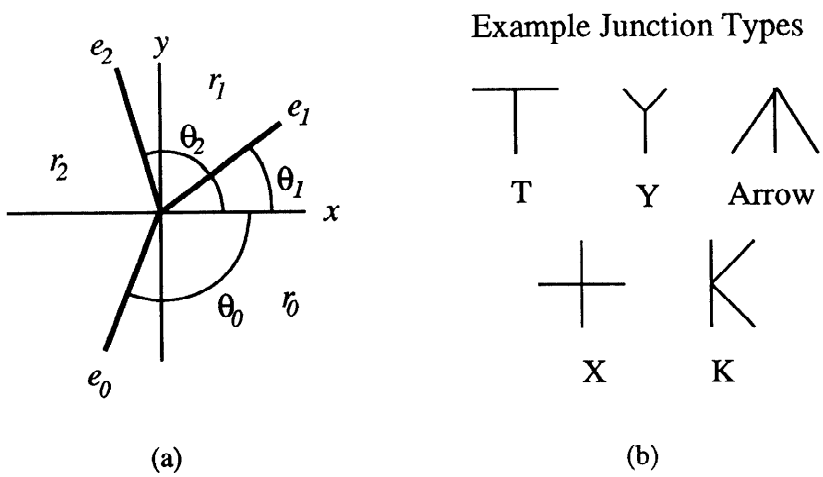


Figure 2: (a) Our model n-ary junction is the intersection of n regions of constant intensity (shown here for $n = 3$). (b) Some examples of junctions composed of straight edges, the type of junction we can model.

numerical integration, were computed using the Connection Machine.

Now that we have defined our model of a junction, let us consider why gradient-based edge detectors disconnect edges at junctions. The first cause we explore is that the edge operators $\nabla^2 f$ and $\frac{\partial^2 f}{\partial n^2}$ intrinsically pair zero crossings in twos.

2.1 The Pairing of Edges at Junctions

Poggio and Torre [37] have done a detailed analysis of the geometric properties of zero crossings. Using ideas from transversality theory and Morse functions, they showed that zero crossings always form closed contours or contours that terminate at the image boundary. This makes intuitive sense if you think of zero crossings as the boundary between positive and negative regions of $\nabla^2 f$. Thus, the edges incident to a junction will pair off in groups of two: two edges will be paired if they both bound the same positive-to-negative transition in $\nabla^2 f$ (see figure 3).

The only configuration where more than two zero crossings can meet at a junction is when the value of $\nabla^2 f$ or $\frac{\partial^2 f}{\partial n^2}$ at a saddle point is zero (see figure 4(a)). As Poggio and Torre point out, this hyperbolic point is structurally unstable; changing the parameters of our junction will cause the edges to break off in pairs as shown in figures 4(b) or 4(c). Since the configuration of 4(a) is unstable, we would not expect to see edges meeting this way often; the configurations of 4(b) and 4(c) are more commonplace. Interestingly enough, perturbing the junction parameters to force the junction edges to pair corresponds to moving the saddle point above (figure 4(c)) or below (figure 4(b)) the $z = 0$ plane. This suggests that one can try to find junctions by looking at *level* crossings of $\nabla^2 f$ or $\frac{\partial^2 f}{\partial n^2}$ instead of zero crossings. The level crossing that one needs to examine is the critical value

Figure 3: A junction and its zero crossings. The arrows show the gradient direction and the (+) and (-) signs show the sign of $\nabla^2 f$. Edges are paired if they border the same positive-to-negative transition in $\nabla^2 f$.

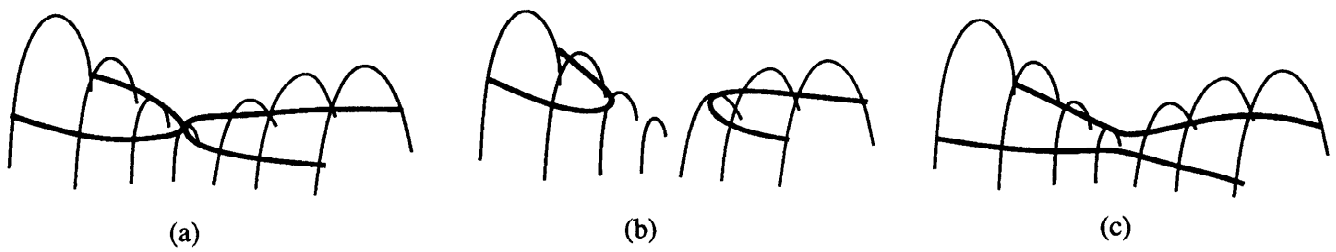


Figure 4: Hyperbolic points are structurally unstable: perturbing the junction that gave rise to (a) can cause the zero crossings to break apart as in (b) or (c).

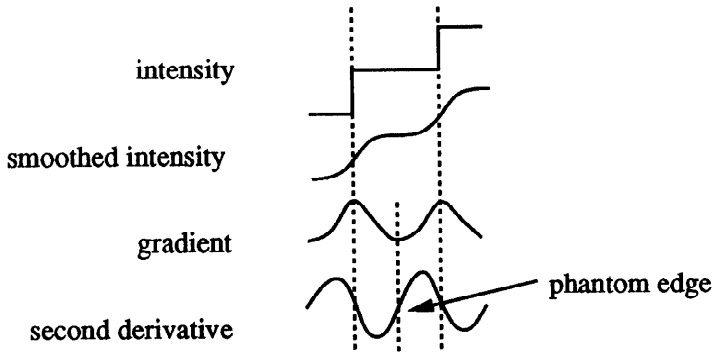


Figure 5: A double step in intensity can create phantom edges, which appear in the middle intensity region (figure adapted from Clark [10]).

of the junction's saddle point. We do not explore this potential technique further, as our junction detection scheme is based on the gradient magnitude.

Given that edges are grouped in pairs of two at junctions, what happens at junctions with an *odd* number of edges, such as our model trihedral junction? *Additional* edges, called phantom edges by Clark [10], will appear to force the total number of edges at a junction to be even. Phantom edges, roughly speaking, are zero crossings in the second derivative associated with local *minima*, rather than local maxima, in the first derivative. Clark introduces phantom edges as a result of a double step in intensity. Shown in figure 5, the double step contains two step edges that are close to one another relative to the width of the smoothing filter. The second derivative has two zero crossings at the two steps, but an additional "phantom" zero crossing appears in the region of intermediate intensity.

A similar analysis can be applied to junctions to predict the occurrence of phantom edges. A junction analogous to the 1D double step, shown in figure 6(a), consists of regions of low, medium, and high intensities. A phantom edge will appear in the region of middle intensity, leading to the zero crossings shown in figure 6(b). Simple reasoning with the edge gradients demonstrates why the phantom edge appears in the middle intensity region. Figure 6(b) shows gradient vectors and the sign of $\frac{\partial^2 f}{\partial n^2}$ associated with each edge. A single edge creates a gradient pointing from the "lower" intensity region r_- to the "greater" intensity region r_+. The second derivative is positive in r_-, zero at the edge, and negative in r_+. The second derivative in the medium intensity region, r_2, is thus bounded by a positive area at one edge and a negative area at the other. Somewhere in the medium intensity region there *must* be a zero crossing. Thus, in general, a phantom edge will occur in any region that is bounded by regions of greater and lesser intensity.

Now that we know how phantom edges are created at junctions, it is easy to see why phantom edges will force the number of incident zero crossings to be even. Imagine making a single circular pass around a junction, starting and finishing at the same point. Since the sign of the second derivative must be the same at the starting/ending point, obviously an even number of $+ \rightarrow -$ and

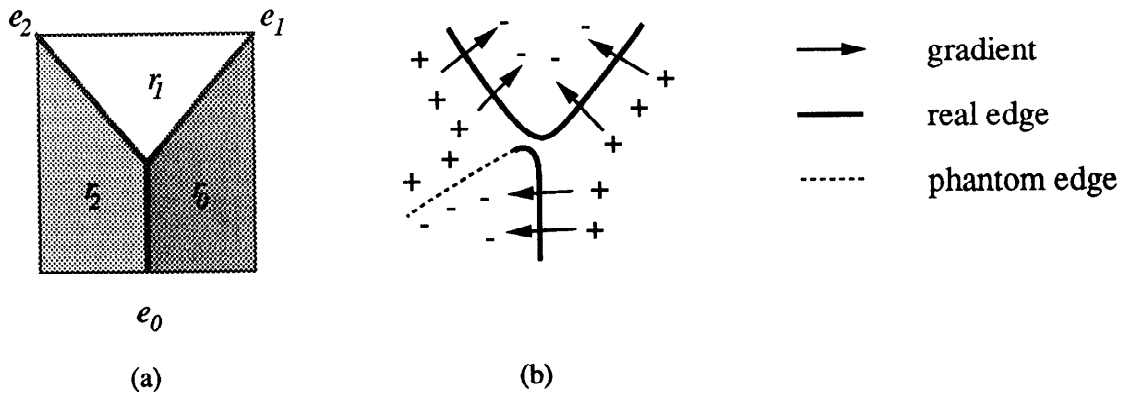


Figure 6: Because of the geometry of gradient vectors, the trihedral junction in (a) must have a phantom edge in r_2, the medium intensity region.

$- \rightarrow +$ transitions occurred while traveling around the junction. All real zero crossings are included among these transitions; phantom zero crossings make up the difference to force the total number to be even.

Phantom edges are important because they tell us in part how edges are disconnected at junctions. That is, an authentic edge paired with a phantom edge will be separated from other junction edges. This is because edge detectors usually employ methods for eliminating phantom edges. Nonmaximum suppression in Canny, for instance, will weed out phantom edges because it explicitly looks for local maxima in the first derivative. Zero crossings of $\nabla^2 f$, as first proposed by Marr and Hildreth, do detect phantom edges, but Clark [10] proposed an authentication measure to distinguish authentic and phantom edges. The authentication measure, basically the third directional derivative of $f(x, y)$, measures the concavity of the first derivative. Local maxima, corresponding to authentic edges, have a negative concavity and thus a negative authentication measure. Phantom edges, on the other hand, have a positive authentication measure. Both Canny edges and authenticated zero crossings of $\nabla^2 f$ are thus similar in character – they only differ by the second differential operator used. Figure 7 shows authenticated zero crossings of $\frac{\partial^2 f}{\partial n^2}$ where the authentication measure is negative in the dark region (authentic edges) and positive in the light region (phantom edges). The edge disconnected from the junction in these cases is the one paired with the phantom edge.

In a trihedral junction there is always one phantom edge to force the total number of edges to be four. As described earlier, the four edges break into two pairs, and the authentic edge paired with the phantom edge is disconnected from the junction. This naturally raises the issue of which edge the phantom edge pairs with. Referring back to figure 6, we know the phantom edge lies in region r_2, the medium intensity region. For simple geometrical reasons, the phantom edge must be paired with edges e_0 or e_2 – the only other legal configuration is that all edges meet in an unstable hyperbolic point. This immediately rules out the possibility that the phantom edge can be paired with edge e_1, the strongest[2] edge in the junction. The phantom edge must be paired with one of the

[2]We are measuring strength by gradient magnitude, which is proportional to the intensity difference at the edge.

Figure 7: Zero crossings of $\frac{\partial^2 f}{\partial n^2}$ for example T and Y junctions. The zero crossings that fall in the dark regions are authentic, and the zero crossings that fall in the light regions are phantom.

two weaker edges.

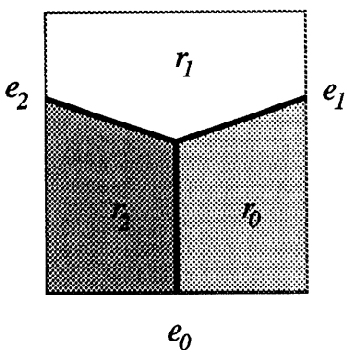
Empirically, we have found that the weakest edge at a trihedral junction is paired with the phantom edge and thus disconnected. A qualitative account of why this happens can be presented for the second directional derivative operator, $\partial^2 f/\partial n^2$, since a good "procedural description" of the operator exists. To obtain this operational description, we first note that the first derivative in the gradient direction is precisely the gradient magnitude:

$$\begin{aligned} \frac{\partial f}{\partial n} &= \nabla f \cdot \frac{\nabla f}{|\nabla f|} \\ &= |\nabla f| = \sqrt{f_x^2 + f_y^2} \end{aligned}$$

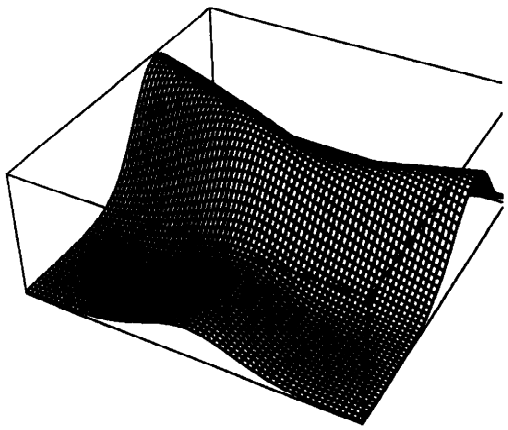
Authenticated zero crossings of the second directional derivative are thus the local maxima of the gradient magnitude taken in the *gradient direction.* The skewing effect of the stronger edge gradients on the weaker edge's gradient direction, as we will see, causes the weakest edge to be disconnected. We demonstrate by way of an example.

Consider the gradient magnitude and direction of the junction shown in figure 8(a), where we have deliberately made edge e_0 weak. Each edge contributes a "ridge" to the gradient magnitude (figure 8(b)), with the ridge height proportional to the intensity difference at the edge. To describe the gradient direction, we must look at the gradient as a vector field. Each edge contributes to the gradient a vector perpendicular to its tangent. Because of smoothing, though, these edge gradients interact at the junction, and, naturally, the stronger edges swamp out the weaker ones. As a result, the gradient direction at a junction is perpendicular to the stronger edges. As shown in figure 8(c), edges e_1 and e_2 dominate the gradient direction at the junction. The gradient direction along edge e_0 is skewed by the stronger gradients. This skewing effect disconnects the edge e_0 from the junction, because, at point P in figure 8(d), $\partial^2/\partial n^2$ will differentiate *up* the gradient ridges of edges e_1 and e_2 in direction v_a instead of perpendicular to edge e_0's gradient ridge, direction v_b.

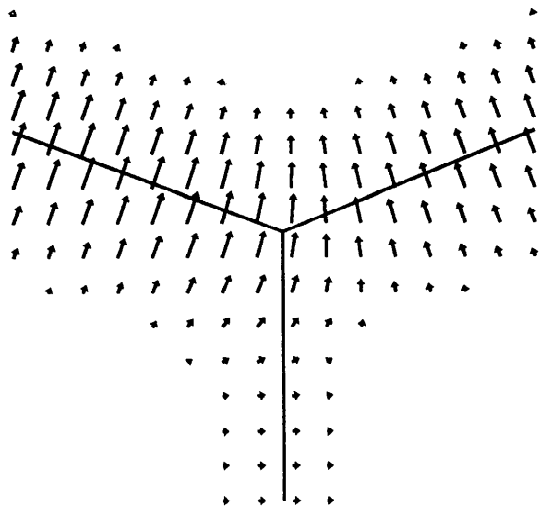
It is important to note the effect of changing the width of the Gaussian filter on zero crossings



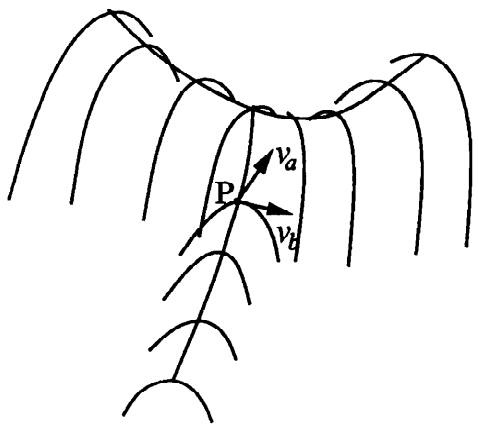
(a)



(b)



(c)



(d)

Figure 8: The junction shown in (a) produces the gradient magnitude (b) and a gradient vector field (c). The strong gradient along edges e_1 and e_2 swamp out the weak gradient along edge e_0 near the junction, which, in part (d), causes $\frac{\partial^2 f}{\partial n^2}$ to differentiate along v_a instead of the edge perpendicular v_b at point P.

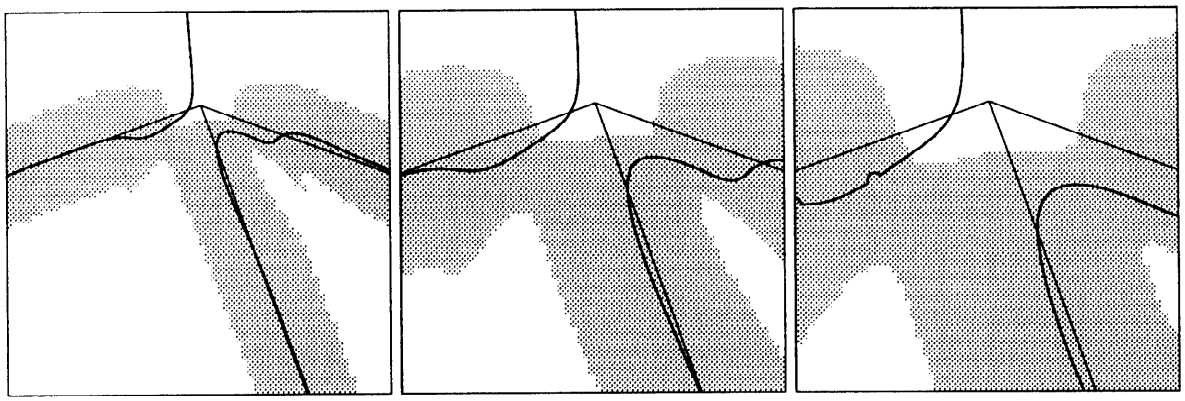


Figure 9: The effects of increasing the width of the smoothing Gaussian filter: zero crossings contract from the junction center as σ increases from 1 in (a) to 2 in (b) and 3 in (c). The darkened regions contain real edges – there is one phantom edge extending upward.

at junctions. From research on edge detection, there is a well-known trade-off between using a large σ to combat noise and a small σ to better localize edges; i.e. increasing σ eliminates noise at the cost of moving edges from their true locations. A similar breakdown in localization happens at junctions. As σ is increased, zero crossings not only wander from their true locations, but the gaps between edge pairs increase. As shown in figure 9, increasing σ from 1 to 3 causes the zero crossings to contract away from the junction center, increasing the spacing between them. Thus, junction detection becomes more problematic as σ is increased. For junction edges paired with phantom edges, the edge's endpoint will move away from the junction center as σ increases. One way of looking at the effect of raising σ is to think of smearing gradient ridges together. Since differentiation and convolution commute, we can think of finding zero crossings as first computing the gradient or Laplacian and then smoothing the result. Increasing σ just increases the effects of inter-edge interference, such as gradient skew discussed previously. Bergholm [2] has explored a technique called edge focusing to try to reap the benefits of both a high and low σ. High σ edges are tracked as σ is slowly decreased, bettering edge localization and reconstructing junctions.

So far our analysis has used a continuous model for junctions and differential operators. We are ultimately interested in discrete images – what qualifications must we make when speaking about images? While much of this depends on the properties of the image and the discrete differentiator chosen to implement ∇^2 and $\partial^2/\partial n^2$, some things may be said in general. Since edge points are spatially discretized, a gap at a junction in a real edge map might not show up if it falls between samples. The distance between the endpoint of the disconnected edge and the other junction edges needs to be on the order of one pixel before it appears in discrete edge maps. Thus, discretization actually assists in detecting junctions – some junctions that are disconnected in continuous space will be connected in discrete edge maps. But, as we will see in the next section, discretization can cause problems, too.

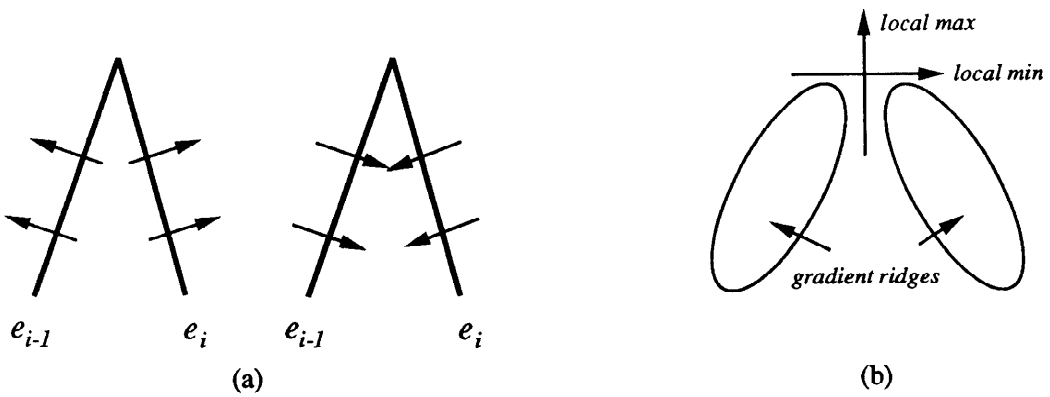


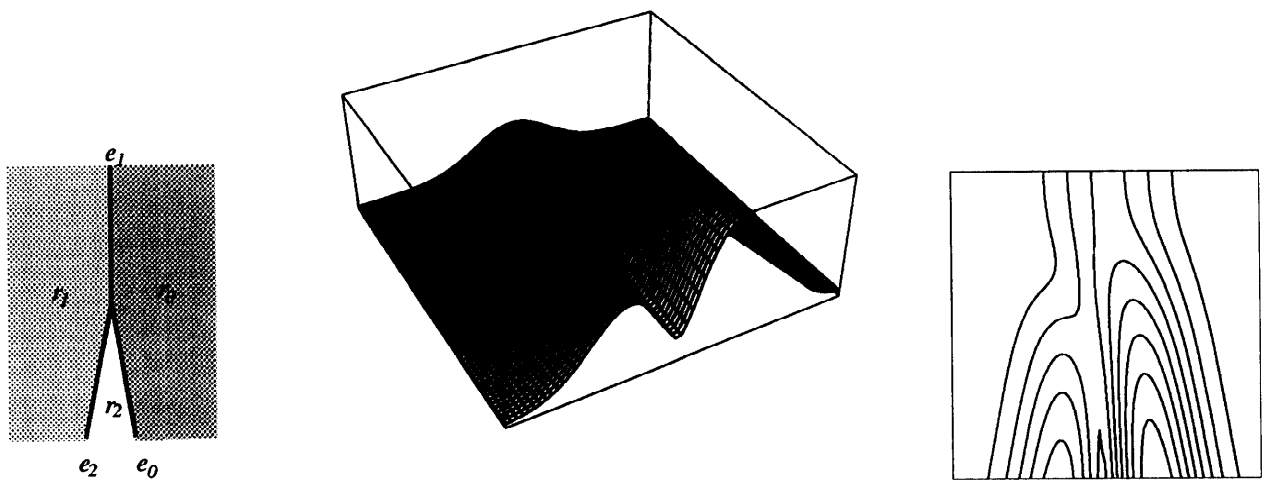
Figure 10: The gradient vector fields from e_{i-1} and e_i in (a) oppose one another, leading to a saddle point in the gradient magnitude near their intersection, (b).

To review, we have seen that the natural pairing of zero crossings at junctions causes gradient-based edge detectors to miss junctions. There is another factor that contributes to edge fragmentation at junctions, a factor that we call vanishing gradients.

2.2 Vanishing Gradients

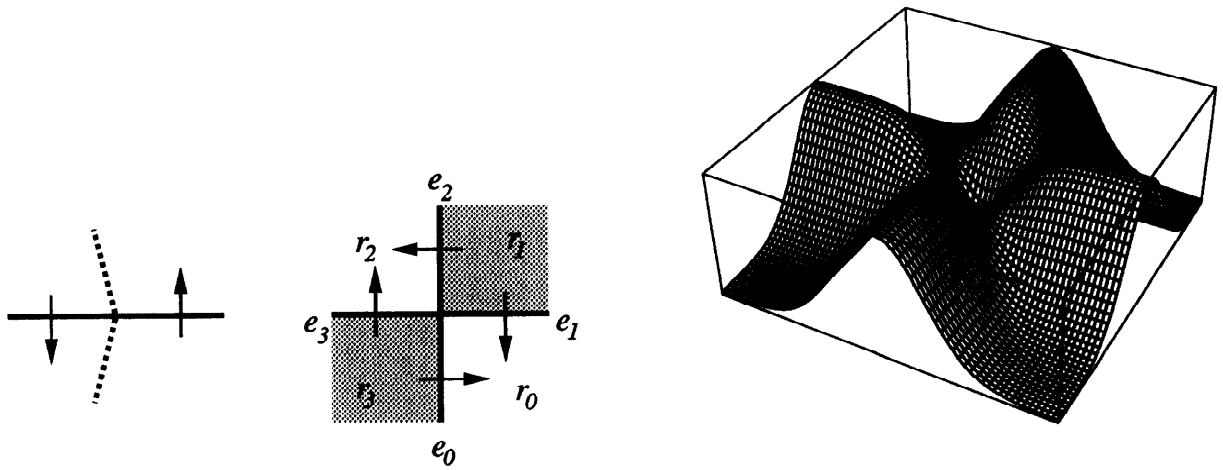
The gradient vector fields of two edges can interfere destructively with one another at a junction, contributing to edge fragmentation. This happens whenever the gradient vector fields from two edges point in opposite directions. For instance, in a junction where two edges meet at an acute angle, the gradient vectors can be made to oppose one another (see figure 10(a)). Because of smoothing, the gradient vectors destructively interfere, causing the gradient magnitude to actually decrease where the two edges meet. This creates a saddle point in the gradient magnitude at the junction, as shown in figures 11(b) and (c) for the junction of figure 11(a). At the saddle, the local maximum is in a direction that roughly bisects the acute angle and the local minimum "bridges" the two gradient ridges belonging to the two edges (see figure 10(b)). How pronounced the saddle point is, or how much the gradient magnitude dips near the junction depends on how acute the angle is. The smaller the angle, the more opposed the gradient vectors are, so the gradient dips more.

A second type of edge geometry where gradients combine destructively is pictured in figure 12(a). This case is not guaranteed to generate a saddle point; all that can be said for sure is that the destructive interference tends to decrease the gradient magnitude at the junction center. A particularly extreme example of this kind of gradient interaction is shown in figure 12(b), where e_0 opposes e_2 and e_1 opposes e_3. One can show analytically that the gradient at the center is zero – the opposing gradient vectors from the four edges completely annihilate one another. The gradient magnitude, shown in 12(c) has four saddle points, one between each of the edge pairs e_0-e_1, e_1-e_2, e_2-e_3, and e_3-e_0. As in the acute angle case, the minimum direction in each saddle bridges the gradient ridges between two edges. In general, when vanishing gradients form a saddle point between the gradient



(a) (b) (c)

Figure 11: The destructive interference between gradients at edges e_0 and e_2 in (a) creates a saddle point in the gradient magnitude near their intersection. The saddle point is clearly visible in the mesh plot of the gradient magnitude (b) and the contour plot (c).



(a) (b) (c)

Figure 12: The junction edges in (a) have opposing gradients, but there may not necessarily be a saddle point in $|\nabla f|$. The junction in (b), an extreme example of (a), produces four saddles in $|\nabla f|$ shown in (c).

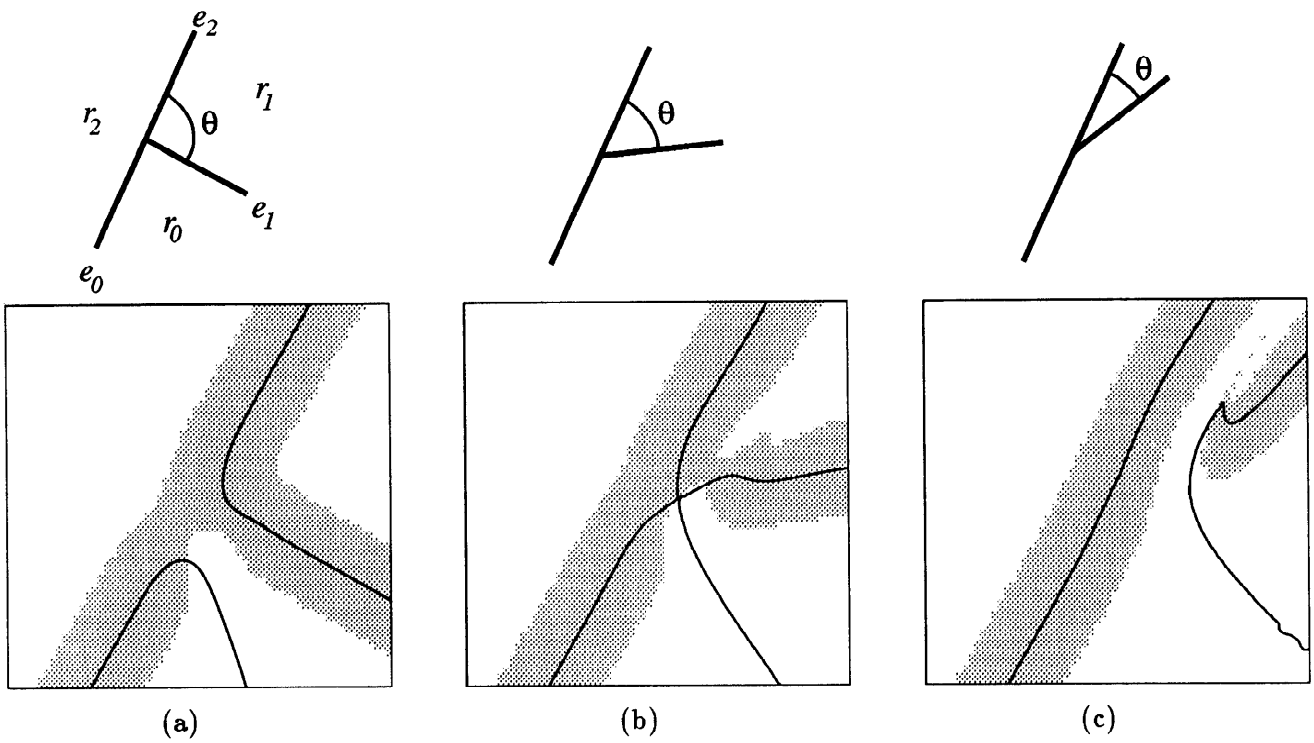


Figure 13: Decreasing the angle θ in (a) increases the effect of vanishing gradients between e_1 and e_2. Vanishing gradients eventually so weakens edge e_1 that it is paired with the phantom edge (part (c)), even though it is stronger than e_0 in (a). Part (b) shows an intermediate case where all edges meet in a hyperbolic point.

ridges of two edges, the minimum direction will link the two edges.

Vanishing gradients affects edge fragmentation in two ways, one affecting the way zero crossings pair, and the other deals with the practical issues of noise and quantization. Recall that for a trihedral junction, the phantom edge pairs with one of the two weaker edges. Empirically, we found that the weakest edge is paired with the phantom edge, but vanishing gradients can change this. That is, vanishing gradients can "weaken" the second strongest edge to such an extent that the phantom edge pairs with it instead. In figure 13, we demonstrate how increasing the effect of vanishing gradients changes the edge the phantom edge pairs with. When $\theta = 90°$ (figure 13(a)), no gradients oppose one another, so the weakest edge is paired. When θ reaches 20° (figure 13(c)), though, vanishing gradients have weakened edge e_1 to the point that it is now paired with the phantom edge. It is interesting to note that in between these two extremes, there is an angle in which all four edges meet in a hyperbolic point (figure 13(b)).

Besides changing which edge pairs with the phantom edge, vanishing gradients can cause additional edge fragmentation at junctions, fragmentation independent of edge pairing. These additional sources of fragmentation deal with the practical issues of real images, as opposed to the ideal con-

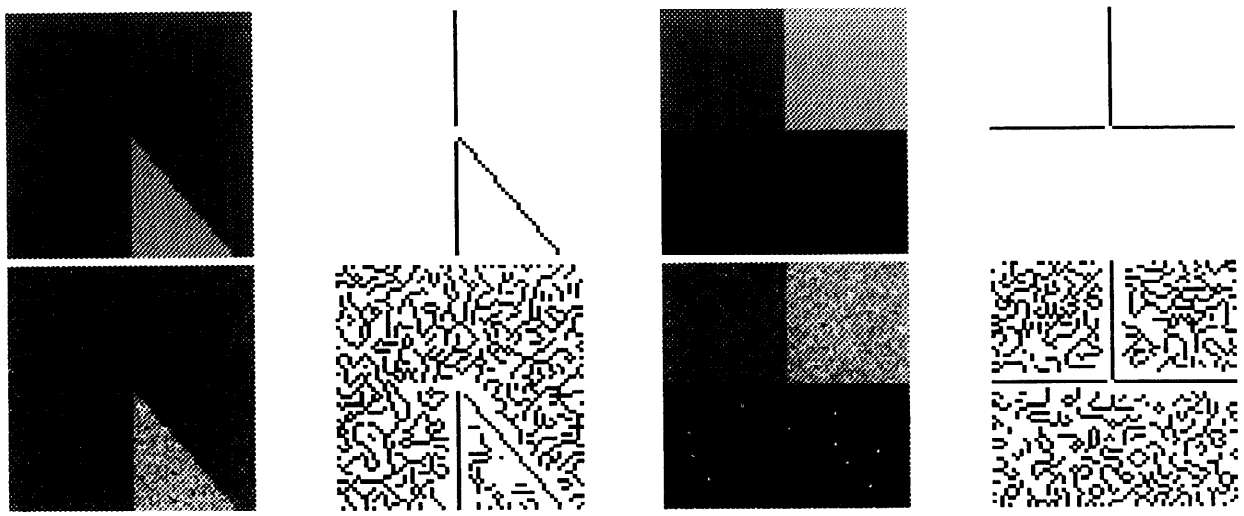


Figure 14: For the junction in figures (a), vanishing gradients increase $\frac{\partial^2}{\partial n^2}$'s sensitivity to noise, as seen in the Canny edges to the immediate right – notice the gap introduced at the junction when the noise is added. Vanishing gradients do not affect the junction on the right (b), so adding noise does not add any new gaps in the edge map.

tinuous analysis that led to edge pairing. The first issue we explore is noise: vanishing gradients make edge detection at junctions more susceptible to noise. The strength of an edge, or its "signal", is usually chosen to be the gradient magnitude at the discontinuity. As we have seen, the gradient magnitude dips where two edges with interfering gradients meet. Thus, the edge "signal" decreases at the intersection of the two edges. If we assume that the noise variance is constant across the image, then the signal to noise ratio decreases at the edge intersection. Thus, as shown in figure 14, noise may disconnect the two edges while the edges themselves remain intact.

Noise and quantization also affect the gradient direction, which has an impact on our $\partial^2/\partial n^2$ operator. The skewing effect of noise and quantization is particularly harmful at the saddle point in gradient magnitude caused by vanishing gradients. At the saddle, the gradient direction points down the valley[3] between the two gradient ridges belonging to the edges involved. The tolerable range of angles for which the saddle point is a local maximum is small compared with normal edge points along a gradient ridge. This is because the gradient valley can be very narrow, and also because the saddle is a local minimum perpendicular to the gradient. Thus, a small change in the gradient direction can cause the gradient to point *up* one of the gradient ridges rather than down the valley. Thus, noise or quantization error can disconnect edges found by $\partial^2/\partial n^2$ by changing the gradient direction from what it would be in the continuous ideal case. It is interesting to note that this is why sharp corners are disconnected in the Canny edge detector.

[3]It could also point 180° in the opposite direction, but the analysis for this case is similar.

In review, we have explored two basic reasons why edges are fragmented at junctions. The first, more theoretical in nature, is that zero crossings intrinsically pair off in twos. Secondly, vanishing gradients at junctions makes edge detection more problematic given the practical issues of noise and quantization. Now that we know the causes of edge fragmentation at junctions, we ask how one might be able to group disconnected edges given information about the gradient magnitude.

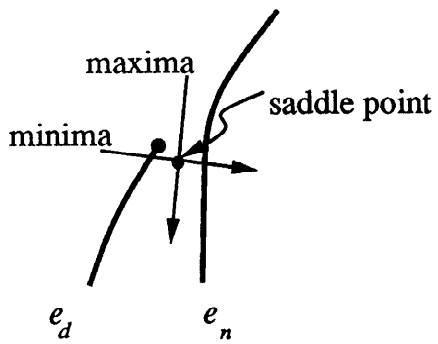


Figure 15: One may group the edge e_d to the junction by following the minimum direction from the saddle point between e_d and e_n.

3 Using the Gradient to Group Edges at Junctions

The previous section discussed the reasons why edges from gradient-based edge detectors are broken apart at junctions. If a real edge does not pair off with another real edge, then it will be disconnected from the junction, separated by a small gap. As a result, edge maps from gradient-based edge detectors are filled with nearly complete junctions: just filling in the small gap between the endpoint of the disconnected edge and its junction will enable a system to detect junctions. This section explores methods for using the gradient to fill these gaps and introduces the bow tie map as a tool for implementing those methods. The techniques developed here form the basis of our junction detector described in the next section.

3.1 Properties of the Gradient at Disconnected Endpoints

One can use the analysis of the previous section to find local properties of the gradient that enable the grouping of a disconnected edge with its junction. When vanishing gradients break off an edge, e_d, there will be a saddle point in the gradient magnitude near e_d's endpoint. The gradient vector field generated by e_d combines destructively with that of another edge, e_n, resulting in a saddle point where the two edges meet (see figures 10 and 11). As shown in figure 15, the minimum of the saddle point is in a direction that links the endpoint of e_d to e_n and the junction center. Thus, one way to fill the gap between e_d and the junction is to follow the minimum direction away from e_d until e_n is found. This rule is related to a similar idea by Korn [21] for filling in gaps at junctions by looking for local minima in the gradient magnitude.

Another method for grouping edges must be found for junctions with no saddle points in $|\nabla f|$. Such cases will arise when no edge gradient opposes any other or the opposition is too small to be detected. The disconnected edge e_d will be one of the weaker edges in the junction; stronger edge gradients nearby have skewed its gradient direction, as discussed in the last section. The edge e_d

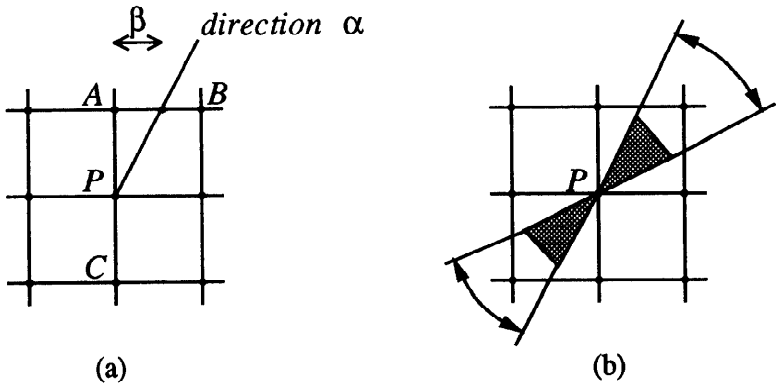


Figure 16: (a) We use linear interpolation between A and B to calculate the value of the gradient magnitude in direction α: the interpolated value $= \beta B + (1-\beta)A$. For the max bow tie in (b), $|\nabla f|$ at P is bigger than the indicated interpolated values.

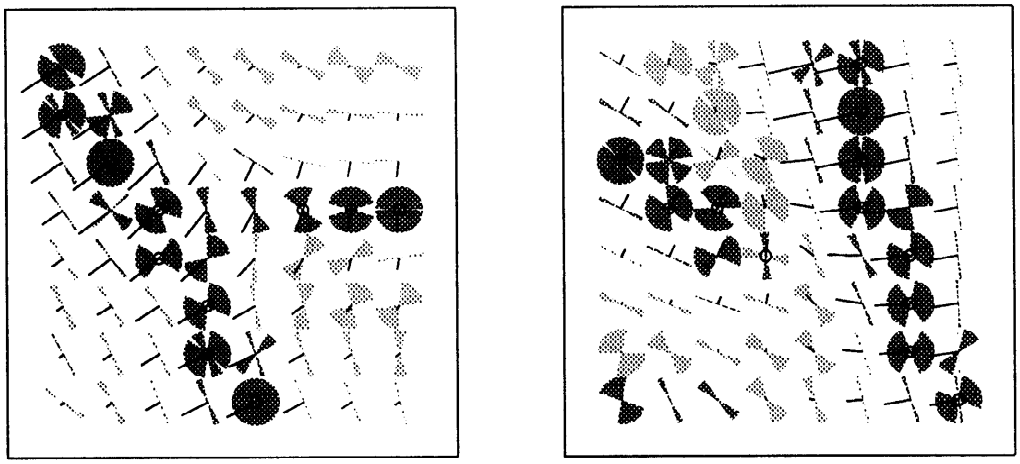
will be paired with a phantom edge; the endpoint of e_d is the location where e_d turns from being authentic to phantom. The gradient magnitude ridge for e_d will not dip in value before it combines with the stronger gradient ridges of the nearby stronger edges. Thus, the gradient magnitude ridge serves as a link between the endpoint of e_d and the junction. We can fill the gap by following e_d's gradient ridge. Refer back to figure 8, looking at 8(b) for how e_0's gradient ridge continues uninterrupted to the other gradient ridges.

3.2 The Bow Tie Map: A Tool for Representing the Gradient

From our discussion about filling in gaps in edge maps at junctions, we would like a representation for $|\nabla f|$ that would expose gradient ridges and saddle points. We introduce the bow tie map as a way of doing this. The bow tie map captures the local geometry of $|\nabla f|$ by comparing the gradient magnitude of a point P to each of its eight connected neighbors. For instance, we could easily measure whether P is a local maximum along the y-axis in $|\nabla f|$ by comparing the value of $|\nabla f|$ at P to those values of $|\nabla f|$ at P's vertical neighbors, A and C in figure 16(a). More generally, P is a local maximum in direction α if $|\nabla f|$ at P is bigger than $|\nabla f|$ sampled in directions α and $-\alpha$[4]. We can find the gradient magnitude in any direction α by linearly interpolating between P's eight connected neighbors. For instance, for directions α in the range 45° to 90° from the x-axis, we interpolate $|\nabla f|$ between points A and B (see figure 16(a)).

The bow tie map records at each point two ranges of directions: one range of directions for which the point is a local maximum in $|\nabla f|$, and a second range for which the point is a local minimum. The former range is called a max bow tie and the latter a min bow tie. These ranges can be easily computed in closed form because of the fact that we are using linear interpolation. We call the direction ranges bow ties because we draw them by filling in directions in two wedges that look like a bow tie (see figure 16(b)).

[4]The direction $-\alpha$ is the opposite direction of α.



(a) (b)

Figure 17: Bow tie maps at a junction affected by gradient skew (a) and by vanishing gradients (b). In (a), gap points arise in the edge map where the gradient direction moves outside the max bow tie. In (b), the saddle point can be used to help fill the gap in the edge map.

One of the better qualities of this representation in helping us find junctions is that it is independent of the gradient direction, and thus not susceptible to gradient skew. This is evident in the bow tie map of figure 17(a), a good example of a junction affected by gradient skew. In the figure, edge points detected by the Canny edge detector are shown with unfilled circles (some points that look like they should be edge points have been thinned away for curve tracking purposes). Max bow ties are dark; min bow ties, light. The gradient is shown as a vector field – the vector at each point shows the gradient direction and magnitude. This figure shows how the gradient direction along the disconnected edge, e_d, moves *out* of the max bow tie. The gap points where this has happened are not picked up as edge points by the Canny edge detector. The edge e_d's gradient ridge is *still intact* in the gap region, as shown by the max bow ties there.

This leads us to the first use of the bow tie map: detecting gradient ridges. A gradient ridge will be a sequence of max bow ties with similar orientation. In the gradient skew case[5], a gradient ridge will be present linking the endpoint of e_d to the other junction edges. From e_d's endpoint, we try to extend e_d by following a path of max bow ties with fairly constant orientation. This orientation tends to be perpendicular to the extension. Since a weaker edge is being reconnected to stronger edges, the gradient magnitude should *increase* along the path of max bow ties – we are ascending the gradient ridges of the stronger edges. Furthermore, since the weaker gradient ridge is disappearing into the stronger ones, the max bow ties tend to get thinner as the junction is neared (this happens in figure 17(a)).

For disconnected edges caused by vanishing gradients, we may use the saddle point in $|\nabla f|$ to help reconnect the edge. A saddle point will appear in the bow tie map as a point with both a min

[5] Assuming no or negligible vanishing gradients.

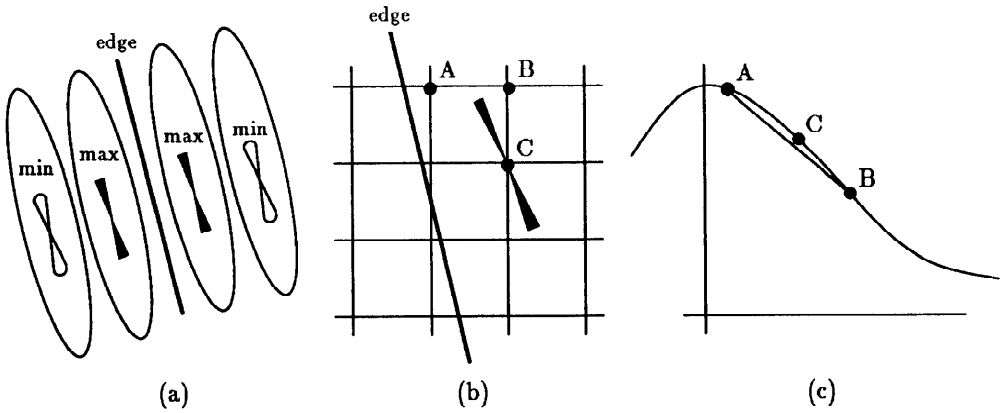


Figure 18: (a) The layers of min and max bow ties surrounding an edge; (b) The max bow tie at C depends on the linear interpolation between A and B; (c) The interpolated points between A and B fall below the Gaussian in its concave down section.

and a max bow tie. As already discussed, the minimum direction of this saddle point will point towards the other junction edges. From e_d's endpoint, we simply search for the nearby saddle point and extend the edge along the minimum direction. Figure 17(b) shows the bow tie map of a junction affected by vanishing gradients. Notice that the endpoint of the disconnected edge is a saddle point. Also, noise, quantization, and the weakening effect of vanishing gradients have broken the path of max bow ties from the disconnected edge to the other junction edges. Thus, it is necessary to have extension rules that look at both gradient ridges and saddle points.

Looking at the gradient ridges along the edges in figures 17(a) and 17(b), one notices very thin max and min bow ties oriented parallel to the edges. Although not immediately evident from the figures, a more thorough examination of these thin bow ties near gradient ridges reveals that there are two "layers", a max layer near the edge, right next to the edge points, and a min layer beyond that (see figure 18(a)). There should only be a maximum condition *across* the ridge, not a max or min condition *along* the ridge. So, what is going on? We believe that the explanation has to do with the fact that we are using a linear interpolation of the gradient after a Gaussian filter has been applied. Consider the max bow tie at point C in figure 18(b). We are interested in the computation of the bow tie in the direction parallel to the edge, which involves an interpolation between points A and B. We claim that the max bow tie range is made larger by the linear interpolation process. We can view the gradient magnitudes at points A, B, and C as being sampled on a Gaussian at a point determined by how far they are from the edge, as shown in figure 18(c). When "close" to the edge, the Gaussian is concave down, so the interpolation will be *less* than the actual value along the Gaussian. Thus, the interpolated points are artificially low, so point C is effectively increased in value. This tends to create a max bow tie oriented parallel to the edge. A symmetric argument occurs for points in the min bow tie layer. Only this time, we are in the concave up section of the Gaussian, so interpolated points are artificially high. This will create min bow ties. Overall, because we combine a linear interpolation with a Gaussian filter, we get an artifact of max and min bow

ties oriented parallel to the edges. Empirically, these bow ties are very thin and the gradient points perpendicular to the bow tie, so a system might be able to distinguish them from "real" bow ties.

In summary, we have introduced the bow tie map as a tool for finding gradient ridges and saddle points in $|\nabla f|$. Two techniques for extending disconnected edges have been discussed: following gradient ridges and following the minimum direction of saddle points. We have built a junction detector based on these two techniques; we present it in the next section.

4 A Junction Detection Algorithm

We have developed an algorithm for detecting junctions that works by filling in the gaps at junctions in edge maps. A postprocessing step to edge detection, the algorithm extends disconnected edges, using the techniques of the last section to "grow" edges from their endpoints. In keeping with our analysis of gradient-based edge detection schemes, we chose to use the Canny edge detector [7] as the preliminary edge detection step. We favored Canny edges over other gradient-based schemes because of its good localization of edges and immunity to noise. After finding Canny edges and identifying edge endpoints, the junction detection algorithm works in two phases. First, an edge grouping phase, using gradient information, determines which edges to group together to form a junction. Then a second phase refines this grouping by using geometrical information to actually extend the disconnected edge. After filling in the gaps at junctions, we are left with a "network" of edges, an edge map with both edges *and* junctions.

4.1 Edge Grouping

In the edge grouping phase, disconnected edges are grouped with their fellow junction edges using the gradient information in the bow tie map. The endpoints of disconnected edges, extended using the ridge and saddle minimum rules, are grouped with the edges intersecting the extension. We now examine how the ridge and saddle minimum following rules are implemented.

As explained in the previous two sections, we can use saddle points in the gradient magnitude to group disconnected junction edges via the saddle's minimum direction. We basically want to extend the disconnected edge in a direction aligned with a nearby saddle point's minimum direction. First, from the disconnected edge's endpoint P, we search P's neighbors for a saddle point, a point with both a min and max bow tie. In the search for saddle points, we not only include the usual pixels at integral coordinates, but also at locations offset by half a pixel; i.e. pixel location (12.5, 15.5). Gradient values at these "half pixel" coordinates are estimated by bilinear interpolation. Next, the saddle's minimum direction is determined by the average direction of the min bow tie. Finally, we simply linearly extend the disconnected edge from the saddle point. Along the path from the saddle point to the neighboring junction edges we expect to be climbing a gradient ridge; i.e. the gradient magnitude should be increasing since we are traveling in the minimum direction. Thus, we measure $|\nabla f|$ at the saddle, $|\nabla f|_{critical}$, and we make sure that each point along the linear extension has a gradient magnitude at least as large as this value. (To allow for noise, we test for only 90 percent of $|\nabla f|_{critical}$.) If $|\nabla f|$ at some point along the extension dips below $|\nabla f|_{critical}$, then the extension is stopped and the disconnected edge left ungrouped. The extension stops successfully when it intersects other edges.

The second method for using the bow tie map to group disconnected edges is to follow gradient ridges from the endpoint of the disconnected edge. We use this rule when there is no saddle point in $|\nabla f|$ nearby, which indicates that vanishing gradients have not weakened or destroyed the gradient ridge linking the endpoint to the other junction edges. The goal here is to follow a path of similarly

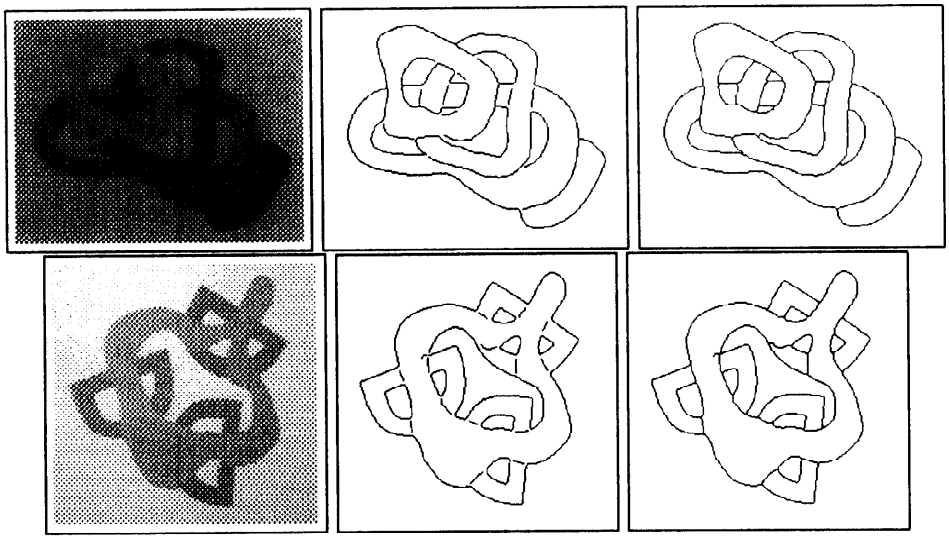


Figure 19: Grouping phase results: real grey level images of construction paper cutouts (left), Canny edges (middle) and edge extensions using gradient information (right).

oriented max bow ties from the endpoint to the other junction edges. We "grow" the endpoint along the gradient ridge by recursively applying the following technique. Rank the endpoint's "forward neighbors" using the scoring function

$$\text{score} = \frac{|\nabla f|}{|\text{mean angle}(\text{max bow tie}_{endpoint}) - \text{mean angle}(\text{max bow tie}_{forward\ neighbor})|}$$

and continue the edge to the highest ranking point. This tends to favor those endpoint neighbors with big gradients and max bow ties similarly oriented to that of the endpoint. We define the "forward neighbors" of an endpoint P as the three eight connected neighbors that lie most closely along the edge's extension. Additionally, as with the saddle point case, we expect the extension to be rising up a gradient ridge. To enforce this constraint, we measure $|\nabla f|$ at the endpoint, $|\nabla f|_{endpoint}$, and require that all extension points to have $|\nabla f|$ greater than 90 percent of $|\nabla f|_{endpoint}$.

We show some edge grouping results in figure 19. The original grey levels are on the left, Canny edges are in the middle, and edges with gaps filled in at junctions are on the right. One of the difficulties the edge grouping system encounters, though not evident in figure 19, is with short "noise edges" that exist in the Canny edge map when gradient thresholds are not properly chosen. When the noise edges are close to one of the real edges, they often attach themselves to the stronger edge, creating a false positive junction.

What can be done to help eliminate these false positive junctions? We have tried to create an "edge saliency" measure to distinguish noise edges from real edges. Our edge saliency measure ranks edges by their average gradient, length, and smoothness. Since the measure favors long and smooth edges with high contrast, short noise edges will naturally score poorly and can be filtered out. Ideally only the real edges will remain to be extended. We use a threshold on edge saliency

in the final results (see figures 21 and 22) to eliminate some noise edges from consideration. As we have implemented it here, edge saliency is a generalization of the technique for using a threshold on $|\nabla f|$ to distinguish edge points from noise. More generally, edge saliency can be used as a tool for filling in gaps and selecting important edges (see Sha'ashua and Ullman [36]).

As a technique for eliminating false positives, edge saliency is not entirely satisfactory for two reasons. First, the edge saliency measure does not necessarily completely separate the real and "noise" edges; i.e. the lowest scoring real edges may not be more salient then the highest scoring noise edge. Second, there seems to be no principled way to choose a saliency threshold. What we have done in our test cases was to deliberately select a low, conservative threshold so as to avoid ignoring real edges but still eliminating many noise edges.

Another way to eliminate false positive junctions is to require all edges to be either closed or joined with other edges at both endpoints after junction detection. This eliminates the false positive junctions where a noise edge is dangling from a real edge and has one free endpoint. The resulting edge junction graph would consist of closed cycles, a natural condition for defining image "regions". This technique is applicable when the real edges *do* form a network of closed cycles – a missing edge could make this heuristic eliminate some true junctions. We study the usefulness of this technique in the discussion section, but did not use it in the final results of figures 21 and 22.

The results of figure 19 demonstrate the motivation for having a geometry-driven stage for refining the extensions. A few of the edge extensions have tangent discontinuities with the original edge or do not intersect the junction center properly. The edge grouping may be correct, but the extension and the resulting junction center are not well localized. This problem seems to be caused by the interaction of strong and weak gradients brought on by smoothing. If a weak edge's gradient is absorbed into a stronger edge's ridge far from the junction (> 3 pixels), then the ensuing gap will be large. When the growth process from the disconnected edge starts climbing the stronger edge's gradient ridge (and the influence of the weaker edge is negligible), then the growth process naturally takes the fastest path up the ridge, which is perpendicular to the stronger edge. This path may not be a nice continuation of the disconnected edge. This is why we emphasize that the techniques presented so far are *grouping* techniques – we will use geometrical information to fill the gaps between grouped edges.

4.2 Gap Filling

Unlike the edge grouping phase, the gap filling phase does not use gradient information. As such, it is independent of the analysis of sections 2 and 3 and is not at the heart of the junction detector. This is a *refinement* step, using geometrical information to improve the gap filling of the edge grouping phase. The goal of this refinement step is to find a smooth continuation of the disconnected edge to the edge it was grouped with. There are two hypotheses that we are testing: either the disconnected edge e_d continues well with one of the other junction edges (figure 20(a)), or it meets the other curve with a tangent discontinuity (figure 20(c)). Key to this analysis is the

Figure 20: In the gap filling phase, we assume that the disconnected edge either continues well with one of the other edges (a), or meets them in a discontinuity, (c). Figures (b) and (d) show the cubics we fit to edge portions to decide between the two cases.

fitting of cubic polynomials to portions of junction edges. We will use the second derivative (the second derivative is used as a measure of curvature) of these cubics to evaluate the extensions of e_d under different hypotheses.

First we test the hypothesis that e_d continues well with one of the other junction edges. Assume that e_d was grouped with the pair of edges e_1 and e_2, as shown in figure 20(a). Here we are assuming that e_1 and e_2 were not disconnected in the original edge map. The edge e_d can continue well with either e_1 or e_2. Both possibilities are tested by fitting two cubic polynomials c_1 and c_2 (see appendix B for the details of cubic fitting) to the two edge pairs e_d-e_1 and e_d-e_2 respectively (see figure 20(b)). We judge the continuity of two edges by a second derivative measure of the fitted cubics, taken to be the second derivative sampled at several points along the cubic. If the second derivative measure of either c_1 or c_2 is below a threshold, then the implied edge grouping is formed and the gap is filled using the cubic. If both c_1 and c_2 have a measure below the threshold, the cubic with the lowest measure wins.

If e_d does not continue well with one of the edges at the junction, then it must meet the other junction edges with a discontinuity. In this case, we need to determine which point along edge e_1 (see figure 20(d)) e_d extends to. We simply test many points along e_1 centered around the point P, the intersection of the grouping pass extension with e_1. We choose the point along e_1 that generates the fitted cubic with the smallest second derivative measure. Again, the gap is filled with the "winning" cubic.

Now that the description of the junction detector is complete, let us look at some results. Figures 21 and 22 show image "triplets" consisting of the original grey level image, Canny edges, and Canny edges with junctions. Notice how the gaps are filled in at junctions. We used $\sigma = 1.0$ for the Gaussian smoothing step applied before Canny. The hysteresis parameters fed to Canny are: low = 2 and hi-to-low = 2 or 3, where the units are in terms of the noise estimate. The noise estimate is chosen to be the 30th percentile of the gradient magnitude histogram. We chose a threshold on the edge saliency measure to prevent the extension of some of the "noise edges". We now close this

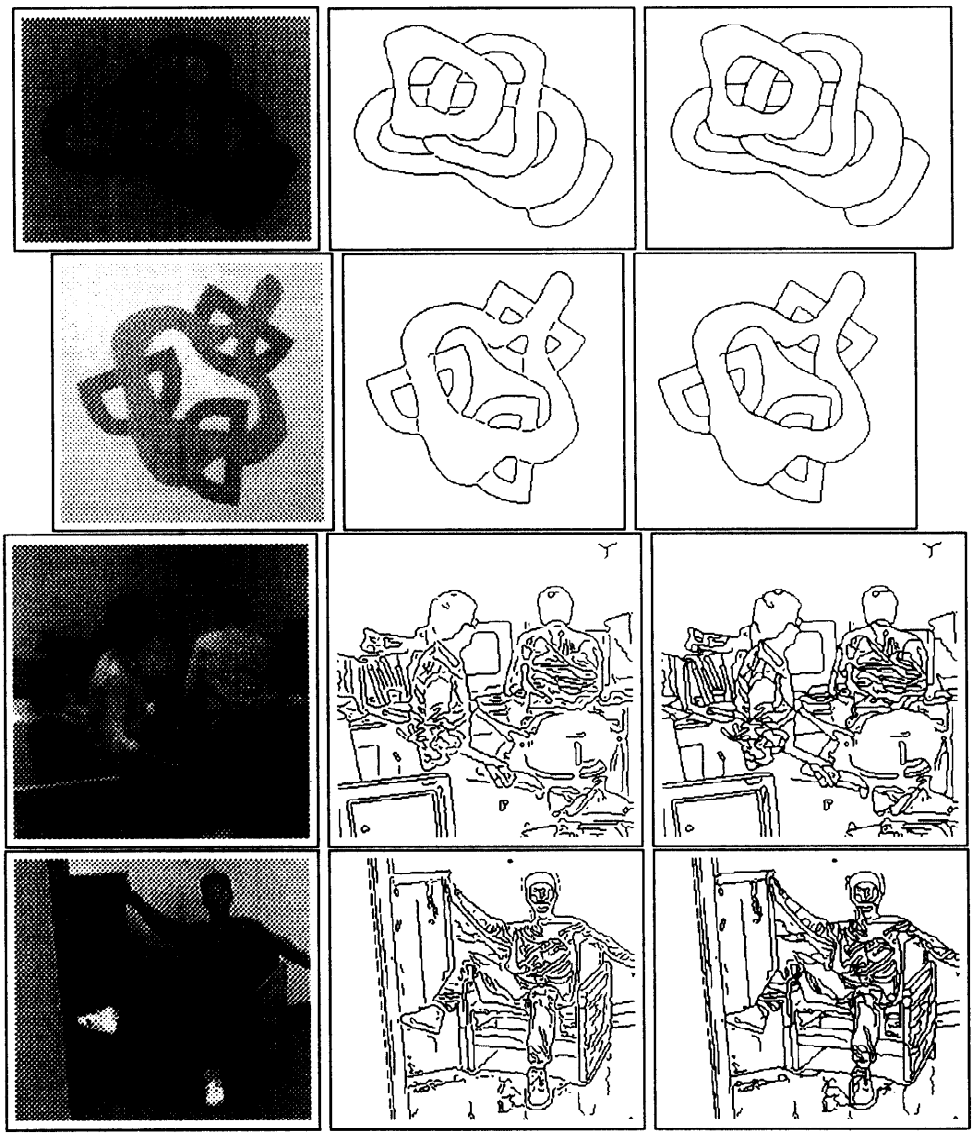


Figure 21: Final junction results: grey level images (`rings`, `d-occlude`, `two-guys`, and `sung`), Canny edges and edge extensions using gradient information.

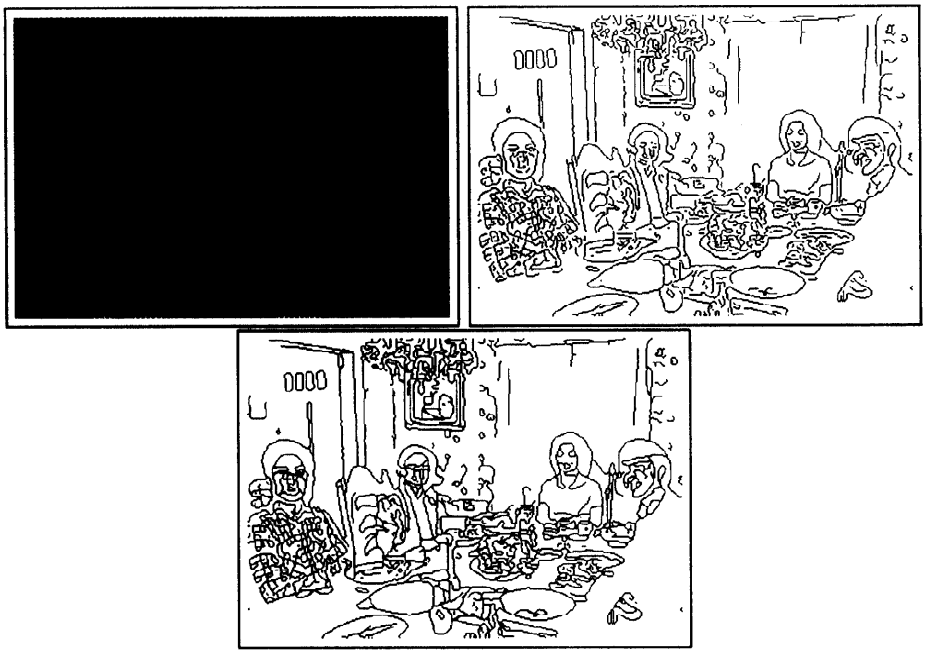


Figure 22: More final junction results: grey level image **Thanksgiving**, Canny edges and edge extensions using gradient information.

section with an evaluation of the junction detector.

4.3 Discussion

In this section we evaluate the junction detector in terms of its computational cost, ability to detect true junctions, and its false positive rate. We will also discuss cases where the junction detector is known to fail.

4.3.1 Timing of our Junction Detector

The amount of time taken by the grouping and cubic fitting stages grows linearly with the number of endpoints in the edge map. As edges are extended from their endpoints, the bow tie map is computed in a 3x3 neighborhood about the endpoint of the extension. Computing the bow tie map itself is inexpensive since it involves up to 16 subtractions and 8 divisions per point. The smoothed gradient values from which the bow tie map is constructed are already available from the edge detection step. We show some timings for the images of figures 21 and 22 in table 1. The gap filling stage, the most expensive step, is only a refinement step – one can get a first cut at the junctions and their locations with only the grouping step. The timings were done on a SPARC IPC and the system was written in Common Lisp.

Comparing the computational expense of our junction detector to that of junction preserving

image	saliency	edge grouping	cubic fitting
rings	1.6	1.5	2
d-occlude	1.4	2	3
two-guys	5	30	45
sung	6	33	47
Thanksgiving	12	48	76

Table 1: Timings of the junction detector, in seconds (SPARC IPC, Common Lisp implementation).

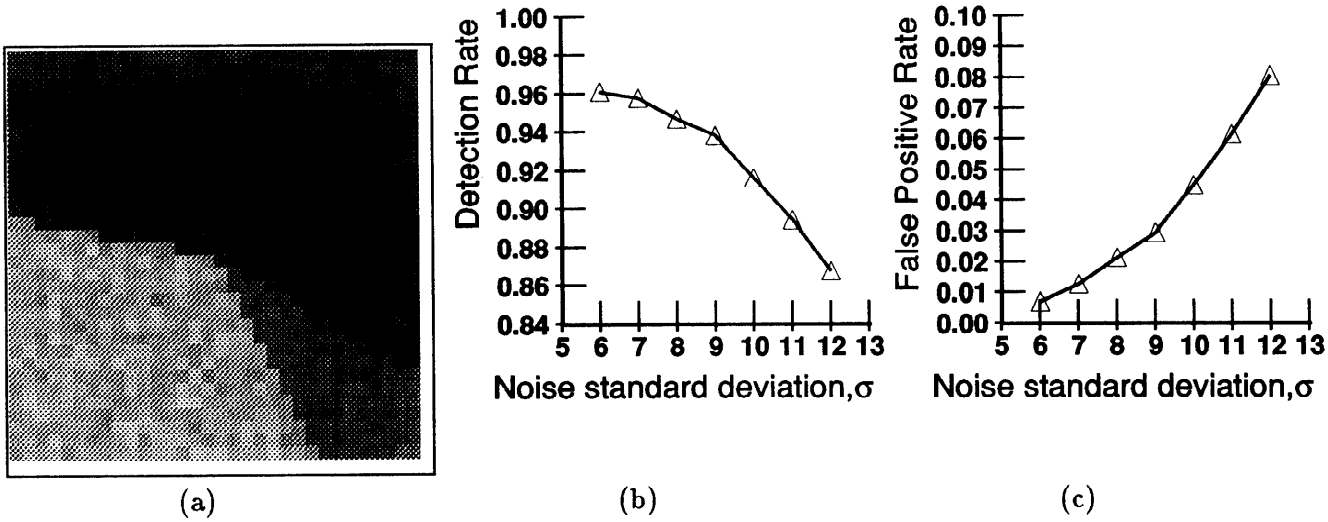
edge detectors and other junction detectors, our detector falls somewhere in the middle. It is cheaper than iterative methods that evolve the image over time, such as the weak membrane model or anisotropic diffusion (all the techniques referred to here will be discussed in the next section). It is more expensive than the simple gradient-based junction finders of Korn [21] and Lacroix [22], but more principled in the sense that we use an analysis of the gradient near junctions to build the junction detector.

4.3.2 Detection and False Positive Rates

One characteristic of the junction detector is that it is fairly aggressive about extending disconnected edges. Empirically, the detector successfully extends "true" edges. Gaps at real junctions are filled correctly most of the time. As previously noted, however, this aggressive extension strategy creates problems with false positive junctions – "noise" edges that latch onto real edges. This happens because the noise edges are interpreted as the "weaker edge" disconnected from a nearby stronger (real) edge. In this section, we look at the junction detector's ability to detect true junctions and reject false ones, both with and without the heuristics discussed before to eliminate false positives.

In order to quantitatively evaluate the junction detector's detection and false positive rates, we have tested it on synthetic trihedral junctions with additive Gaussian noise. The synthetic junctions were constructed from three randomly chosen intensity surfaces between 0 and 255. The boundaries between regions were straight lines with randomly chosen orientations. Intensities and orientations were constrained so that the minimum difference between intensities was 20, and between orientations, 20 degrees. See figure 23(a) for an example junction with additive Gaussian noise with $\sigma = 9$.

We examined the junction detection and false positive rates under varying amounts of noise. We used both the edge saliency and edge networking heuristics to reduce false positives. The results are presented in figures 23(b) and (c). Each point in the two plots represents 10,000 junction test cases. To explain how we computed the detection and false positive rates for each σ value, let us



(a) (b) (c)

Figure 23: (a) Example synthetic junction, σ of additive Gaussian noise is 9, (b) detection rate versus noise σ, (c) false positive rate versus noise σ.

first define some terms. Let

$$\begin{aligned} N &= \text{number of junctions} = 10,000 \\ CD &= \text{number of true detections, Canny} \\ BD &= \text{number of true detections, Bow tie detector} \\ BF &= \text{number of false detections, Bow tie detector.} \end{aligned}$$

Now we define the detection rate as the ratio of true detections by the bow tie detector to the total number of true junctions it *could* have found, which is the total number of junctions, N, minus the number of true junctions found by Canny. That is, the detection rate is defined as

$$\text{detection rate} = \frac{BD}{N - CD}.$$

The false positive rate is simply the fraction of false detections to total detections, or

$$\text{false positive rate} = \frac{BF}{BF + BD}.$$

How applicable are these results to real images? It depends on how closely we have modeled junctions in real images. On one hand, the noise level of $\sigma = 12$ is higher than we expect for real images, so the results for high σ are perhaps a little too pessimistic. On the other hand, the step edge model for junctions is simplistic. Some junctions may have edges that are, say, combinations of steps and impulses. This factor makes the results appear better than those we would expect for real images.

To get an idea on how much the edge saliency and edge networking heuristics add to our ability to reduce false positives, we ran some more experiments, this time keeping σ constant but varying

heuristics	false positive rate	detection rate
neither	.1946	.9213
edge network	.1161	.9370
edge saliency	.0602	.9329
both	.0295	.9384

Table 2: False positive and detection rates as we try different combinations of heuristics to reduce false positives. We used a noise σ of 9.

the number of heuristics used. Both heuristics were tried in isolation and one experiment used no heuristics at all. The results are shown in table 2, where the experiment using both heuristics is repeated for comparison. Clearly, the edge saliency measure is of more value than the edge networking heuristic and should make the false positive rate sufficiently low for most applications.

4.3.3 Known Failure Cases

In our approach to junction finding through extending endpoints, we have ignored the case where junctions break apart by the pairing off of *real* edges. In the example junction back in figure 3, four junction edges form two pairs – there is no endpoint from which to start a growth process. However, junctions such as this one are special cases; recall from section 2 that intermediate valued junction regions introduce phantom edges which pair off with real edges. Also, vanishing gradients can disconnect two real edges that should, in the ideal continuous case, be connected. These factors generate endpoints which give our algorithm a starting point.

To review, we have introduced a junction detector that works by extending the endpoints of disconnected edges at junctions. The extensions are guided by gradient ridges and saddle points. These extensions are refined by fitting cubics to portions of junction edges to find nice edge continuations. Having explored one method of junction detection in detail, let us now review previous work on junction detection by other researchers.

5 Previous Approaches

Our discussion of previous approaches to junction detection will focus on two areas. The first of these is edge detection techniques that do not inherently break apart junction edges. Secondly, we will look at systems that find junctions by modifying an edge operator or doing postprocessing after edge detection. Our system falls into this latter category.

5.1 Junction Preserving Edge Detectors

In the last couple decades of research in machine vision, a multitude of edge detectors have been proposed, each detecting junctions with varying degrees of success. We have already discussed how gradient-based detectors, such as the Laplacian of the Gaussian and the second directional derivative, perform poorly at junctions. There have been, however, edge detectors that do detect junctions well, and in this section we review three such approaches proposed recently, the weak membrane, anisotropic diffusion, and morphological edge detection. These edge detection techniques perform well at junctions because the structure of junctions is not destroyed by a uniform smoothing process, as is the case for gradient-based detectors.

5.1.1 The Weak Membrane Model

In Blake and Zisserman's weak membrane model for edge detection [5], which is an approximation to MRF models with line processes (see Geman and Geman [12], Marroquin *et al.* [30]), the intensity surface of the image is reconstructed by modeling it as a weak membrane. In general, a weak membrane can bend but it cannot crease, so at discontinuities, the membrane "breaks", or tears. The surface is reconstructed by minimizing an energy functional of the following form

$$E = D + S + P.$$

D is the closeness of the reconstructed surface to the data, given by $\int (u - d) dA$, where u is the height of the reconstructed surface and d is the original data. S is a regularization term that forces the reconstructed surface to be smooth; it is of the form $\lambda^2 \int \nabla u^2 dA$, basically favoring those solutions that have small first derivatives. P is a penalty term that discourages the formation of discontinuities, or tears, in the surface. Blake and Zisserman use a penalty function based on the length of the tear, tending to form edge contours of minimal length.

After an iterative process that finds the reconstructed surface, edges, the tears in the surface, are located simply by finding points where local differences in surface values fall above a particular threshold. In general, this process works well at junctions. One reason why it works so well is that the original data are never blurred by smoothing with a Gaussian. The reconstruction does have a smoothing effect, but the original data values are never forgotten and a high discrepancy between data and surface shows up as a high energy contribution in the D term. Another reason for its performance at junctions is that an edge hysteresis stage is "built into" the energy functional. This

hysteresis arises from an interaction between the penalty term, P, and the other terms, D and S, in a gap area between two discontinuities. In terms of energy, it costs less to close the gap and incur the extra penalty than to keep the gap and have an area of high data to surface disparity and high ∇u. The result is that tears in the membrane tend to propagate, continuing broken edges. Thus, we would not expect to see edges disconnected from junctions as often as we see in gradient-based detectors.

The weak membrane model does, however, have an inherent difficulty with what is called a "gradient limit". One parameter to the algorithm, the sensitivity parameter, measures the minimum height of a step edge that it can detect; when a step edge has an intensity gradient many times this sensitivity, many breaks will occur in the membrane. This results in a "fat" edge and decreases the membrane's edge localization ability. Interestingly enough, our junction detector has a similar localization problem introduced by smoothing gradient ridges into one another. Recall that this was the motivation for the geometrically driven gap filling stage.

5.1.2 Anisotropic Diffusion

Perona and Malik [33] have recently used anisotropic diffusion to perform a multiple scale analysis of an image, finding both edges and junctions. Using the initial image as a starting point, anisotropic diffusion "evolves" the image over time by smoothing intensity surfaces within regions while avoiding smoothing over edges. The later "times" compare to the coarser scales in more traditional scale-based approaches. Unlike the case with scale-based approaches, however, the localization of edges at later times in anisotropic diffusion does not deteriorate because smoothing between neighboring regions is avoided. Recall that smoothing across regions is how Gaussian filtering destroys the structure of junctions and smears gradients. Interregion smoothing is avoided by allowing the conductance to vary spatially across the image; the conductance is chosen to be a monotonically decreasing function of gradient magnitude. Thus, conductance is high and smoothing occurs where the gradient is small. There is a gradient threshold below which smoothing occurs and above which edges are enhanced. Junctions are preserved in the piecewise smooth image that evolves over time because interregion smoothing that would otherwise distort junctions is avoided.

5.1.3 Morphological Edge Detection

In morphological edge detection, pixels in the image are treated as elements of sets. Special operators are constructed by creating what is called a structuring element, somewhat like the kernels used in convolutional operators. Figure 24 shows a sample structuring element, D_{rod}, that we will use later in an edge detection example. The image is operated upon by combining these structuring elements with the image in various ways. One such operation, dilation, is defined on grey level images as

$$d(r,c) = \max_{\text{(i, j) in domain of b}} f(r-i, c-j) + b(i,j),$$

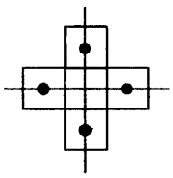


Figure 24: The sample structuring element, D_{rod}. The elements of the set are shown in dark circles. Basically, D_{rod} specifies the four connected neighbors.

where $f(r,c)$ is the image in (row, column) coordinates and b is the structuring element. Another operation, erosion, is defined similarly, but replaces the max operator with min and subtracts the structuring element instead of adding it.

Edge operators can be built by looking at the difference between either dilation or erosion and the original image; this difference is called the dilation or erosion residue. In an example taken from Lee, Haralick, and Shapiro [24], the erosion residue is given by

$$G_e(r,c) = f(r,c) - e(r,c),$$

where $e(r,c)$ is the erosion. If we use D_{rod} as the structuring element and we map it to zero, we can make the following simplification.

$$\begin{aligned} G_e(r,c) &= f(r,c) - min_{(i,j)\in D_{rod}} f(r+i, c+j) \\ &= max_{(i,j)\in N_4(r,c)}(f(r,c) - f(i,j)), \end{aligned}$$

where $N_4(r,c)$ gives the set of four connected neighbors of a point (r,c). This operator simply takes the largest difference between a pixel and his four connected neighbors. Since points near edges will have large difference between neighboring points, edge points are located by thresholding the erosion residue. Lee, Haralick and Shapiro show how this operator behaves and recommend further refinements that combine the erosion and dilation residues.

Noble [32] analyzes how morphological operators can be used for feature detection. In her analysis, she shows how morphological operators take advantage of certain differential geometrical characteristics of the intensity surface. The differential geometry of the intensity surface near step edges and junctions naturally lead themselves to being detected by an operator that computes the difference between the dilation and erosion of the image. She shows some examples of how such an operator performs well at junctions – there appear to be no disconnected edges.

One difficulty with morphological techniques is their sensitivity to noise. When no kind of smoothing or blurring operation is built into the morphological operator, Lee, *et al.* report that the operator is noise sensitive [24]. When blurring is added, the signal to noise ratio improves, but then morphological techniques begin to look more and more like gradient-based edge detection techniques. Both use smoothing and the differences between a center pixel with his neighbors.

5.2 Previous Work on Junction Reconstruction

In the previous analysis of the gradient near junctions, we saw how the pairing of edges and vanishing gradients present problems for gradient-based edge detectors. Also presented were some simple rules for reconstructing broken junctions from the output of the Canny edge detector. How have other researchers dealt with this problem? In this section we look at modifications and extensions to edge detectors that try to restore junctions.

5.2.1 Geometric Heuristics

The Binford-Horn line finder [17] uses geometric heuristics to find vertices (junctions) in an edge map. First, edge points are located by correlating the image with ideal step, roof, and peak kernels. As one would expect from our analysis of the gradient near junctions, edges are broken up at junctions. Binford and Horn use geometric heuristics to clean up the edge image and find vertices. After straight lines are fit to the detected edges, a few rules are applied involving extensions of these linear fits. For clusters of nearby endpoints, a vertex is placed at the point of least squares perpendicular distance to the linear extensions. Next, edges that have an endpoint close to another line are completed, allowing for the detection of T and K vertices. Finally, unattached lines are extended to nearby vertices if their extensions are short and pass nearby the vertex. These edge and vertex finding techniques were successfully applied to creating a "clean" edge and vertex map for blocks world images, suitable for 3D interpretation programs.

5.2.2 Scale Space Approaches

For edge detectors that smooth the image before extracting edges, there is a fundamental trade-off between the ability to localize edges and the ability to filter out noise. As one increases σ used in the smoothing Gaussian, noise suppression improves but edge localization deteriorates, as the smoothing actually changes the location of the edge in the smoothed image. The ability of the edge detector to find junctions also enters this trade-off. In our previous analysis of the gradient near junctions, we saw how the gap size in broken junctions increases as we increase σ. Thus, when trying to preserve junctions in the output of an edge detector, it would be nice to use the smallest σ possible. However, this has the bad effect of increasing the noise response. In general, one would like to have both the localization and junction preserving advantages of using low σ **and** the noise suppression advantages of high σ.

One can gain the advantages of both good localization and good noise suppression by examining the edge maps at several scales or resolutions, using many different σ values. The idea of analyzing the image at several different scales was first advocated by Rosenfeld and Thurston [35]. Later, Marr and Hildreth [29] observed that strong edges are those that are relatively stable across scales. Witkin [41] generalized this by developing the scale-space representation of a 1D signal, a plot of the zero crossing location on one axis and scale on the other. He suggested using the coarser scales

(high σ) to locate important edges and then to localize them by tracking how they move in the scale space plot as σ is reduced.

Bergholm [2] has implemented an edge detector based on Witkin's idea of tracking edges from coarse to fine resolutions, a technique he calls "edge focusing". He traces the edges using a "continuous" approach, meaning that the change in σ is carefully chosen so that edges are almost never displaced by more than one pixel from one resolution to the next. This makes the edge matching across different resolutions trivial. Since he uses the Canny edge detector to find edges at a particular scale, junctions are broken at coarse scales. However, as σ gets smaller in finer resolutions, the gaps in junctions get smaller, and some junctions are completely restored. Not all smoothing can be eliminated, however. This is because the optics of the camera always introduces some blur, and typically some small amount of smoothing is used even at the finest resolutions to control the noise. Thus, edge focusing cannot restore all junctions in full, but it is a good point for other junction restoration techniques to start from since it helps reduce gap size.

Giraudon and Deriche [14] have used scale space in a different way to build a junction detector especially apt at localizing junctions. Their detector is based on how elliptic points in the image intensity function move across scales near junctions. Elliptic points are found via local maxima in the DET operator, defined as

$$\mathrm{DET} = I_{xx}I_{yy} - I_{xy}^2$$

where I is the image intensity. They have found that a line drawn through the elliptic points found at different scales intersects zero crossings of $\nabla^2 f$ at the junction center. For trihedral junctions, two elliptic points are found, one in each extremal intensity surface (highest and lowest). When there is not enough contrast among the intensity surfaces, then only one elliptic point can be robustly recovered, and Giraudon and Deriche call these junctions "vertices like corners." Overall, the technique shows much promise in localizing junctions accurately. One issue not addressed is that of false positives. Elliptic points near a zero crossing generated by noise might be registered as a junction. The only technique they apparently use to eliminate these responses is a threshold on the gradient magnitude.

5.2.3 Gradient-Based Approaches

Other systems that try to restore junctions from edge maps either change the way the gradient is computed or look for certain properties of the gradient nearby junctions. We now turn our attention to some of these methods.

One simple method for reconstructing broken junctions from Canny edges is to group together all of the edges incident to a junction. This can be done by applying a thresholding operation to the gradient magnitude. This method, developed by Huttenlocher and Cass [8] for a larger 3D recognition system, relies on the fact that the gradient magnitude of the points in the gap between the endpoint(s) of the disconnected edge(s) and the junction is high. All pixels with gradient magnitude above a certain threshold are "turned on", producing connected blobs in the image. All the incident

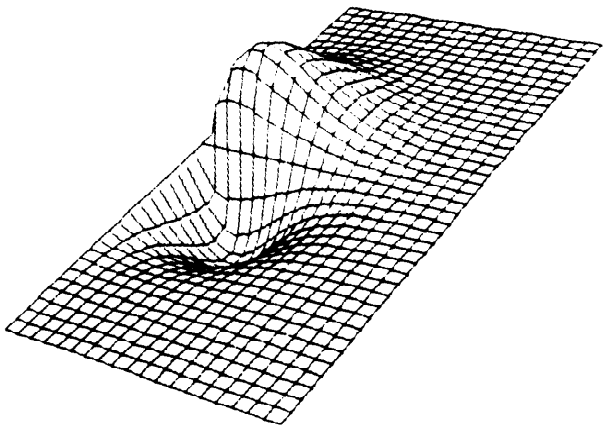


Figure 25: Gennert's half-edge operator.

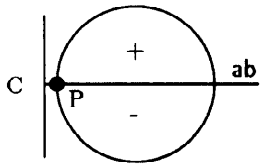


Figure 26: When Gennert's half edge operator is applied at point P oriented along edge ab, it ignores surface C.

edges of a junction should be connected to the same blob. From our analysis of section 2, for junctions affected by vanishing gradients, the gradient threshold should be chosen equal to $|\nabla f|$ at the nearby saddle point.

Gennert [13], noting that edge detectors generally perform poorly at junctions, has explored using directionally selective operators that ignore one side of the image at the point of application, a technique that finds "half edges." Figure 25 shows an example filter for a particular orientation. The operators directionally smooth the image and take the gradient, each time ignoring half the image. Thus, when an operator is applied in the gap area at a point P between two given surfaces (see figure 26), and it is oriented so that the third surface of the junction is being ignored, the gradient is not affected by the third surface. This reduces the deleterious effects of inter-edge gradient interference. Edge points are found by applying the directional operator to the image at many orientations. At each orientation, the point can be labeled an edge if the following conditions are met: (a) the edge detector output is above the estimated noise, (b) the output of the detector at (x, y, θ) is greater than in directions perpendicular to θ, a step similar to non-maximum suppression in Canny, and (c) the output at (x, y, θ) is greater than the values at nearby values of θ. Unfortunately, the idea is expensive to implement because convolutional operators must be applied in many directions at each point. Also, the results that Gennert shows are disappointing in that the junctions are not restored and the edges are more noise sensitive than Canny edges.

Lacroix [22] has investigated a generalization of the non-maximum suppression step to avoid disconnecting junctions. In her modified nms algorithm, each point in the image maintains two counters, a v counter that records how many times the point was visited during nms, and an m counter that remembers how many times that point was a maximum. For each point (x, y), we visit three points in the image: the original point (x, y), the point in the direction of the gradient, and the point in the opposite direction. The v counter in all three points is incremented and the point that has the highest gradient magnitude increments its m counter. After all points have been visited, a "Likelihood of Being an Edge" (LBE) measure, defined as m/v, is used for determining whether a point is an edge point. Since strong edge points always have the maximum gradient magnitude when visited, m should be equal to v, or LBE = 1 for these points. Thus, all points for which LBE = 1 are immediately accepted as edge points. If a point is never a maximum ($m = 0$), it certainly cannot be an edge point, so points with LBE = 0 are immediately rejected as edge points. Points with LBE measures between 0 and 1 have their fate decided by a contour follower; they will be incorporated as edge points if they are the natural extensions of contours of LBE = 1. Lacroix shows a simple example of her modified nms algorithm correctly detecting a T junction, with no disconnected edges. One of the disadvantages of her algorithm as I see it, however, is that updating rule for the m counter is not intuitive, as a point does not have to be a local max for its m field to be incremented – the two lesser points can be on one "side" of the maximum.

Finally, Korn [21] has advocated generalizing the non-maximum suppression stage by looking for a maximum *or minimum* in the gradient in four search directions, along the x and y axes and along the diagonals. In his analysis, he noticed that the gradient direction was skewed near junctions and not representative of the edge perpendicular. This led him to suggest changing nms to look for a maximum in the gradient in any of four search directions. Although he also mentions looking for minima in the gradient, he does not motivate the search for minima. Minima need to be used in order to process a particular example correctly. Indeed, the example is one that is afflicted by vanishing gradients, where, according to our model, using max bow ties will not work. The disconnected edge that the minima rule reconnects to the junction forms a very sharp angle with another incident edge. Furthermore, the surface between the two edges is the darkest surface in the junction. Thus, the junction fits our model for vanishing gradients.

6 Summary

This paper began with the observation that gradient-based edge detectors commonly fragment edges at junctions. One goal of this paper has been to explore two causes of edge fragmentation by looking at the image gradient and zero crossings of the second differential operators ∇^2 and $\frac{\partial^2}{\partial n^2}$. First, zero crossings of $\nabla^2 f$ and $\frac{\partial^2 f}{\partial n^2}$ naturally pair junction edges in twos, except for structurally unstable hyperbolic points. As the amount of image smoothing is increased, the edge pairs "repel" each other more, enlarging the gaps in edge maps. Assuming that an authentication measure is used to suppress phantom edges, the real edges paired with them will terminate near the junction in a disconnected endpoint. A second cause of fragmentation at junctions is the destructive interference of opposing edge gradients caused by smoothing. This effect, called vanishing gradients, contributes to edge fragmentation by decreasing the gradient magnitude at junctions. This increases the $\frac{\partial^2}{\partial n^2}$ operator's sensitivity to noise and quantization, possibly causing two real edges to break apart. The destructive interference of edge gradients forms a saddle point in the gradient magnitude near the junction. We hope this analysis will help researchers understand better the properties of the gradient and zero crossings near junctions.

We used the gradient analysis to develop two techniques for filling the gaps at junctions by extending the endpoints of disconnected edges. One technique is to follow the disconnected edge's gradient ridge until it reaches other junction edges. Secondly, for junctions affected by vanishing gradients, the minimum direction of the nearby saddle point bridges the gap between the disconnected edge and other junction edges. To implement these rules, we developed a representation for the gradient called the bow tie map, which exposes gradient ridges, valleys, and saddle points. We demonstrated a junction detector built using these endpoint extension techniques. It performs well with real edges, but suffers from attaching "noise" edges onto real edges. Overall, the main contributions of the junction detector are the bow tie map and the gradient-based rules for extending disconnected edges.

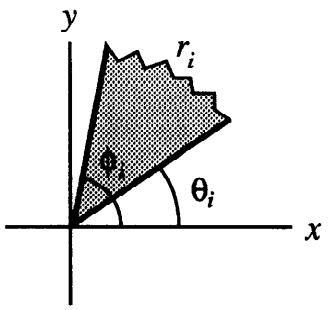


Figure 27: The region specified by our intensity surface model.

A An Analytical Model for an n-ary Junction

This appendix derives analytical expressions for the partial derivatives in x and y of $f(x,y)$, the smoothed intensity function of the n-ary junction introduced in section 2 (see figure 2). From these partial derivatives we shall compute differential operators of interest: the gradient magnitude $|\nabla f|$, the laplacian $\nabla^2 f$, the second directional derivative $\frac{\partial^2 f}{\partial n^2}$, and Clark's authentication measure χ [10].

Recall from section 2 that our model of an n-ary junction is an intersection of n regions of constant intensity r_i, $i = 0,\ldots,n-1$. Each region r_i can be constructed by multiplying r_i by appropriate unit step functions that select the value r_i within its region. The piecewise constant intensity function $I(x,y)$ is simply the sum of n region terms,

$$I(x,y) = \sum_{i=0}^{n-1} \text{region}_i.$$

The region term for region r_i, bounded by edges at angles θ_i and ϕ_i (see figure 27), is

$$\text{region}_i = r_i\, u(y - m_{\theta_i} x) u(m_{\phi_i} x - y), \tag{1}$$

where $m_{\theta_i} = \tan\theta_i$ and $m_{\phi_i} = \tan\phi_i$. The first unit step selects the half plane above $y = m_{\theta_i} x$, and the second unit step selects the half plane below $y = m_{\phi_i} x$. In this model for region$_i$, θ_i and ϕ_i are restricted to the range $-90° < \theta_i, \phi_i < 90°$, but very similar models (measuring angles relative to the y axis, for instance) can help relieve this restriction. Another restriction is $\phi_i - \theta_i \leq 180°$; we will use another model to handle this case.

Having modeled each intensity region r_i, we want to compute partial derivatives of the smoothed intensity function $f(x,y)$, where

$$f(x,y) = I(x,y) * G(x,y,\sigma),$$

and $G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. These partials will enable us to compute the following differential operators:

$$|\nabla f| = \sqrt{f_x^2 + f_y^2} \tag{2}$$

$$\begin{aligned}
\nabla^2 f &= f_{xx}^2 + f_{yy}^2 \\
\frac{\partial^2 f}{\partial n^2} &= \frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_{yy} f_y^2}{f_x^2 + f_y^2} \\
\chi &= \frac{\partial f}{\partial n} \frac{\partial^3 f}{\partial n^3}
\end{aligned}$$

Since convolution and differentiation are linear operators, we can find the partials of $f(x,y)$ by computing the partials of each region convolved with $G(x,y,\sigma)$. Thus, we can carry out the analysis for just one region r_i and sum the results to get the desired partial. We have

$$\begin{aligned}
f(x,y) &= I(x,y) * G(x,y,\sigma) \\
&= \sum_{i=0}^{n-1} \text{region}_i * G(x,y,\sigma) \quad \text{since convolution is linear.}
\end{aligned}$$

In this appendix we will compute the first partial in x, $f_x(x,y)$, and y, $f_y(x,y)$; second and third partials follow by further differentiation. Let $f_{x,i}$ be the partial of region r_i in x,

$$f_{x,i} = \frac{\partial}{\partial x} \text{region}_i * G(x,y,\sigma).$$

Again, we find f_x by summing over regions:

$$f_x = \sum_{i=0}^{n-1} f_{x,i}.$$

Now, to compute $f_{x,i}$ for region r_i from equation 1:

$$\begin{aligned}
f_{x,i} &= \frac{\partial}{\partial x} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} r_i u(y - y' - m_{\theta_i}(x - x')) u(m_{\phi_i}(x - x') - y + y') G(x', y', \sigma) dx' dy' \\
&= -r_i m_{\theta_i} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta((y - y' - m_{\theta_i}(x - x')) u(m_{\phi_i}(x - x') - y + y') G(x', y', \sigma) dx' dy' + \\
&\qquad r_i m_{\phi_i} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(y - y' - m_{\theta_i}(x - x')) \delta(m_{\phi_i}(x - x') - y + y') G(x', y', \sigma) dx' dy' \\
&\qquad \text{Letting } y' = y - m_{\theta_i}(x - x') \text{ in the first term and } y' = y - m_{\phi_i}(x - x') \text{ in the second:} \\
&= -r_i m_{\theta_i} \int_{-\infty}^{\infty} u(m_{\phi_i}(x - x') - m_{\theta_i}(x - x')) G(x', y - m_{\theta_i}(x - x'), \sigma) dx' + \\
&\qquad r_i m_{\phi_i} \int_{-\infty}^{\infty} u(m_{\phi_i}(x - x') - m_{\theta_i}(x - x')) G(x', y - m_{\phi_i}(x - x'), \sigma) dx' \\
&\qquad \text{Since } m_{\phi_i} > m_{\theta_i} \text{ in our model, the unit step selects } x' < x: \\
&= -r_i m_{\theta_i} \int_{-\infty}^{x} G(x', y - m_{\theta_i}(x - x'), \sigma) + r_i m_{\phi_i} \int_{-\infty}^{x} G(x', y - m_{\phi_i}(x - x'), \sigma) \qquad (3)
\end{aligned}$$

Thus, we have decomposed the partial $f_{x,i}$ into two terms that separate the parameters θ_i and ϕ_i. The θ_i term gives $f_{x,i}$ along the edge $y = m_{\theta_i} x$ and the ϕ_i term along the edge $y = m_{\phi_i} x$. Since the adjacent regions r_i and r_{i-1} share the same bounding edge e_i (θ_i for r_i equals ϕ_{i-1} for region r_{i-1}, see figure 28), we can combine terms from r_i and r_{i-1} for edge e_i. Define $d_i = r_{i-1} - r_i$. Then

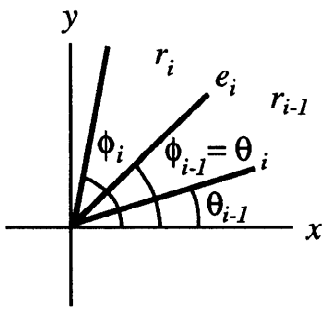
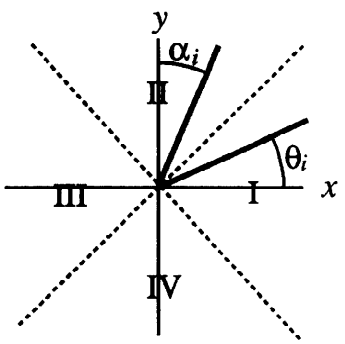


Figure 28: Because regions r_{i-1} and r_i share edge e_i in common, terms from $r_i's$ and $r_{i-1}'s$ contribution to f_x may be combined.



Variations in edge contributions to partials

region	integration limits	variable of integration
I	$-\infty$ to x	x
II	$-\infty$ to y	y
III	x to ∞	x
IV	y to ∞	y

Figure 29: A different type of intensity model for region$_i$ is used in each of the four regions I, II, III, and IV.

the contribution of edge e_i to f_x is

$$m_{\theta_i} d_i \int_{-\infty}^{x} G(x', y - m_{\theta_i}(x - x'), \sigma) dx' \tag{4}$$

Thus, to compute f_x, we can either sum equation 3 over regions or equation 4 over edges. Summing 4 over edges is better because of the combined terms. Since our model for region r_i is limited to $-90° < \theta_i, \phi_i < 90°$, we need to use a set of models that will cover all possible angles. We divide the xy plane into four regions (see figure 29) and use a different model in each. The contribution of edge e_i to f_x for each type of region model is presented in table 3. The slight differences in region models give rise to the differences in integration limits and the variable of integration. For regions II and IV, the angle α_i, measured from the positive y axis (see figure 29), is used to avoid problems with the $\theta_i = \pm 90°$ case in region I ($m_{\theta_i}|_{\theta_i=\pm 90°} = \tan \pm 90° = \infty$).

The analysis for f_y is similar and yields the contributions to edge e_i listed in table 4. Second and third partial derivatives can be found through further differentiation. For instance, the contribution

region	contribution of e_i to f_x
I	$m_{\theta_i} d_i \int_{-\infty}^{x} G(x', y - m_{\theta_i}(x - x'), \sigma) dx'$
II	$d_i \int_{-\infty}^{y} G(x - m_{\alpha_i}(y - y'), y', \sigma) dy'$
III	$-m_{\theta_i} d_i \int_{x}^{\infty} G(x', y - m_{\theta_i}(x - x'), \sigma) dx'$
IV	$-d_i \int_{y}^{\infty} G(x - m_{\alpha_i}(y - y'), y', \sigma) dy'$

Table 3: Edge contributions to f_x by region.

region	contribution of e_i to f_y
I	$-d_i \int_{-\infty}^{x} G(x', y - m_{\theta_i}(x - x'), \sigma) dx'$
II	$-m_{\alpha_i} d_i \int_{-\infty}^{y} G(x - m_{\alpha_i}(y - y'), y', \sigma) dy'$
III	$d_i \int_{x}^{\infty} G(x', y - m_{\theta_i}(x - x'), \sigma) dx'$
IV	$m_{\alpha_i} d_i \int_{y}^{\infty} G(x - m_{\alpha_i}(y - y'), y', \sigma) dy'$

Table 4: Edge contributions to f_y by region.

of edge e_i to f_{xx} in region I is

$$m_{\theta_i} d_i \left[G(x, y, \sigma) + \int_{-\infty}^{x} \frac{m_{\theta_i}(y - m_{\theta_i}(x - x'))}{\sigma^2} G(x', y - m_{\theta_i}(x - x'), \sigma) dx' \right]$$

$$\left(\text{helpful fact:} \quad \frac{d}{dx} \int_{-\infty}^{x} f(x, x') dx' = f(x, x) + \int_{-\infty}^{x} \frac{d}{dx} f(x, x') dx' \right)$$

Using these equations for the partials of $f(x, y)$ for each edge e_i, we simply sum over the edges to get the final partials f_x, f_y, f_{xx}, and so on. From here, we can use equations 2 to compute the desired differential operators over $f(x, y)$.

On a final note, we mentioned that the modeled intensity for region r_i in equation 1 restricts θ_i and ϕ_i to $\phi_i - \theta_i \leq 180°$. For $\phi_i - \theta_i > 180°$, we must use the new model

$$\text{region}_i = r_i \left[u(y - m_{\theta_i} x) + u(y - m_{\phi_i} x) u(m_{\theta_i} x - y) \right].$$

Running this new model through the same analysis as before (including combining two edge terms from adjacent regions) yields exactly the same results for the edge contributions to the partials.

B Fitting Cubics to Edges

In this appendix, we develop a method for fitting cubic polynomials to edges. In our junction detection algorithm, we use the smoothness of fitted cubics to evaluate how well two edge fragments "continue" with one another during the gap filling stage. We measure a cubic's smoothness using its second derivative, which we will examine after showing how to fit a cubic to a list of edge points.

Suppose that we have a list of n edge points (x_i, y_i), $1 \leq i \leq n$. The cubic polynomial that we fit to these points is of the form $(p_x(t), p_y(t)) = (a_x t^3 + b_x t^2 + c_x t + d_x, a_y t^3 + b_y t^2 + c_y t + d_y)$. Thus, there are actually two cubics, one for x and one for y. We shall deal with the x and y coordinates separately, fitting $p_x(t)$ to the x_i's and $p_y(t)$ to the y_i's. We demonstrate the fitting procedure for the x coordinates only; the y_i's fitting is similar.

We fit the x_i's to the polynomial $p_x(t)$ by minimizing the summed squared error between the points and n points on the cubic. The error is given by

$$E = \sum_{i=1}^{n} (p_x(t_i) - x_i)^2, \tag{5}$$

where t_i is the point on the cubic closest to x_i; i.e. $p_x(t_i) \approx x_i$. We have freedom in choosing the t_i's – minimizing the expression for E will *make* them close to the x_i's.. We basically want to keep the parametric distance $t_i - t_{i-1}$ roughly proportional to the Euclidean distance $|x_i - x_{i-1}|$. We can accomplish this by setting $t_1 = 1$, $t_n = n$, and scaling the intermediate t_i's by their distance along the piecewise linear fit to the list of points

$$t_i = 1 + (n-1)\frac{\sum_{j=2}^{i} |x_j - x_{j-1}|}{\sum_{j=2}^{j=n} |x_j - x_{j-1}|}.$$

To minimize the error E, we differentiate equation 5 with respect to the parameters for $p_x(t)$, a, b, c, and d and set the partial derivatives to zero. This leads to the following set of equations

$$\begin{aligned}
2 \sum_{i=1}^{n} (p_x(t_i) - x_i) t_i^3 &= 0 \\
2 \sum_{i=1}^{n} (p_x(t_i) - x_i) t_i^2 &= 0 \\
2 \sum_{i=1}^{n} (p_x(t_i) - x_i) t_i &= 0 \\
2 \sum_{i=1}^{n} (p_x(t_i) - x_i) &= 0
\end{aligned}$$

These can be rewritten as

$$\sum_{i=1}^{n} (a_x t_i^3 + b_x t_i^2 + c_x t_i + d_x) t_i^3 = \sum_{i=1}^{n} x_i t_i^3$$

$$\begin{aligned}
\sum_{i=1}^{n}(a_x t_i^3 + b_x t_i^2 + c_x t_i + d_x)t_i^2 &= \sum_{i=1}^{n} x_i t_i^2 \\
\sum_{i=1}^{n}(a_x t_i^3 + b_x t_i^2 + c_x t_i + d_x)t_i &= \sum_{i=1}^{n} x_i t_i \\
\sum_{i=1}^{n}(a_x t_i^3 + b_x t_i^2 + c_x t_i + d_x) &= \sum_{i=1}^{n} x_i
\end{aligned}$$

which, in turn, can be written in matrix form to isolate the cubic parameters

$$\begin{bmatrix}
\sum_{i=1}^{n} t_i^6 & \sum_{i=1}^{n} t_i^5 & \sum_{i=1}^{n} t_i^4 & \sum_{i=1}^{n} t_i^3 \\
\sum_{i=1}^{n} t_i^5 & \sum_{i=1}^{n} t_i^4 & \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^2 \\
\sum_{i=1}^{n} t_i^4 & \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i \\
\sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} 1
\end{bmatrix}
\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}
=
\begin{bmatrix}
\sum_{i=1}^{n} x_i t_i^3 \\
\sum_{i=1}^{n} x_i t_i^2 \\
\sum_{i=1}^{n} x_i t_i \\
\sum_{i=1}^{n} x_i
\end{bmatrix}$$

Now, we simply solve this linear system for a, b, c, and d, the parameters of the cubic $p_x(t)$.

We used a "second derivative" measure to judge the smoothness of a cubic fit during the gap filling phase of our junction detector. Again, the second derivative is closely related to curvature (equal to curvature when the curve is parameterized by arclength), so sampling the second derivative should give a good estimate of the cubic's smoothness (smooth curves will have low measures). The measure is simply the sum of the magnitude of the second derivative at the t_i's:

$$\text{second derivative measure} = \sum_{i=1}^{n} \sqrt{(p_x''(t_i))^2 + (p_y''(t_i))^2},$$

where $p_x''(t) = 6a_x t + 2b_x$ and $p_y''(t) = 6a_y t + 2b_y$.

References

[1] Dana H. Ballard and Chrisopher M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] Fredrik Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–740, 1987.

[3] Valdis Berzins. Accuracy of laplacian edge detectors. *Computer Vision, Graphics, and Image Processing*, 27:195–210, 1984.

[4] David J. Beymer. Junctions: Their detection and use for grouping in images. Master's thesis, Massachusetts Institute of Technology, 1989.

[5] Andrew Blake and Andrew Zisserman. *Visual Reconstruction.* The MIT Press, Cambridge, MA, 1987.

[6] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: The local–feature–focus method. *International Journal of Robotics Research*, 1(3):57–82, 1982.

[7] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[8] Todd A. Cass and Daniel P. Huttenlocher. A massively parallel implementation of a three-dimensional object recognition system. unpublished manuscript, 1988.

[9] Indranil Chakravarty. A generalized line and junction labeling scheme with applications to scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):202–205, 1979.

[10] James J. Clark. Authenticating edges produced by zero-crossing algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:43–57, 1989.

[11] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[12] Stuart Geman and Don Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[13] Michael A. Gennert. Detecting half-edges and vertices in images. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 552–557, 1986.

[14] Gerard Giraudon and Rachid Deriche. On corner and vertex detection. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 650–655, Lahaina, Maui, Hawaii, 1991.

[15] A. Guzman. Computer recognition of three dimensional objects in a visual scene. Technical Report MAC-TR-59, MIT, 1968.

[16] Robert M. Haralick. Digital step edges from zero crossings of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:58–68, 1984.

[17] Berthold K. P. Horn. The Binford-Horn LINE-FINDER. A.I. Memo No. 285, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1971.

[18] D.A. Huffman. Impossible objects as nonsense sentences. In E. B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6. Edinburgh Univ. Press, Edinburgh, U.K., 1971.

[19] Daniel P. Huttenlocher. Three-dimensional recognition of solid objects from a two-dimensional image. Technical Report AI–TR 1045, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1988.

[20] T. Kanade. Recovery of the three dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–461, 1981.

[21] Axel F. Korn. Toward a symbolic representation of intensity changes in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):610–625, 1988.

[22] Vinciane Lacroix. A three-module strategy for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):803–810, 1988.

[23] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. Technical Report 368, New York University, 1988.

[24] James S. J. Lee, Robert M. Haralick, and Linda G. Shapiro. Morphologic edge detection. *IEEE Transactions on Robotics and Automation*, 3(2):142–156, 1987.

[25] Shih Jong Lee, Robert M. Haralick, and Ming Chua Zhang. Understanding objects with curved surfaces from a single perspective view of boundaries. *Artificial Intelligence*, 26:145–169, 1985.

[26] David G. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Boston, 1985.

[27] A.K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4(2):121–137, 1973.

[28] Jitendra Malik. *Interpreting Line Drawings of Curved Objects.* PhD thesis, Stanford University, 1986.

[29] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London*, B(207):187–217, 1980.

[30] J. Marroquin, S. Mitter, and Tomaso Poggio. Probabilistic solution of ill-posed problems in computational vision. In *Proceedings Image Understanding Workshop*, pages 293–309, Miami Beach, FL, December 1985.

[31] E. De Micheli, B. Caprile, P. Ottonello, and V. Torre. Localization and noise in edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1106–1117, 1989.

[32] J. Alison Noble. Morphological feature detection. In *Proceedings of the International Conference on Computer Vision*, pages 112–116, Dec 1988.

[33] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[34] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*, volume 2. Academic Press, New York, 1982.

[35] Azriel Rosenfeld and Mark Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, C-20(5):562–569, 1971.

[36] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proceedings of the International Conference on Computer Vision*, pages 321–327, dec 1988.

[37] Vincent Torre and Tomaso Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2), 1986.

[38] Lewis W. Tucker, Carl R. Feynman, and Donna M. Fritzsche. Object recognition using the connection machine. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 871–878, Ann Arbor, Michigan, 1988.

[39] Shimon Ullman. Maximizing rigitity: The incremental recovery of 3-d structure from rigid and rubbery motion. A.I. Memo No. 721, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1983.

[40] David L. Waltz. Understanding line drawings of scenes with shadows. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[41] Andrew P. Witkin. Scale-space filtering. In *Proceedings IJCAI*, pages 1019–1022, 1983.