Interactive Road Finding for Aerial Images

Jianying Hu Bill Sakoda Theo Pavlidis

Computer Science Department
SUNY at Stony Brook
Stony Brook NY 11794

Abstract

Fully automatic road recognition remains an elusive goal in spite of many years of research. Most practical systems today use tedious manual tracing for the entry of data from satellite and aerial images to geographical data bases. This paper presents a semi-automatic method for the entry of such data. First ribbons of high contrast are found by analyzing gray scale surface principal curvatures. Then, pixels belonging to such ribbons are fitted by conic splines, and then a graph is constructed whose nodes are end points of the arcs fitted by the splines. The key new idea is to assign edges between all nodes and label them with a cost function based on physical constraints on roads. Once a pair of end points is chosen, a shortest path algorithm is used to determine the road between them. Thus a global optimization is performed over all possible candidates.

1. Introduction

Fully automatic road recognition remains an elusive goal in spite of many years of research. The major reason is that in practice, detection of roads is complicated by contrast variations, occlusion, changes in surface texture and nearby objects. As a result, low level road detection modules [1,2,3] produce numerous and highly fragmented segments which do not meet the requirement of applications such as map construction and map updating. It becomes necessary, then, to develop algorithms to deal with noise, select true road segments from the output of low level modules and link them properly to form complete and clean representations of road networks. All the roads in the images should be represented and and all the gaps between road segments properly filled; and all the noisy segments should be eliminated. This problem has been addressed before by other researchers and several automatic or semi-automatic grouping and linking algorithms have been presented which improve the results generated by low level modules [4,5,6,7,8] . Unfortunately, these algorithms work only in restricted cases and do not always produce complete and clean representations of roads. Most practical systems today use tedious manual tracing for the entry of data from satellite and aerial images to geographical data bases: the image is viewed by an operator, possibly in stereo, on a photoplotter, and knot points for splines approximating roads are picked manually. This paper proposes a semi-automatic method for the entry of such data. We develop an interactive system which allows a road to be selected by picking its end points. The system then traces the road between the end points by searching for a shortest path defined by metrics based on well-known properties of road features. This approach provides a practical method for speeding up map construction, while providing the experience necessary for more complete automation of road detection.

The main ideas in this work are: 1. Global analysis is applied to reconstruct the roads. We have cast the problem in terms of a shortest path through a graph to apply global information. 2. Feedback is used in the linking process. The low-level detector is first run to find ribbon segments of high confidence. The grey scale levels from the original image, and grey scale surface curvature information produced by the low level detector, are then used to help decide how to link the segments and fill the gaps.

Figure 1 summarizes our results.

2. Outline of the linking algorithm

Our interactive road linking algorithm takes as input the fragmented segments produced by a low level ribbon center detector. The ribbon center detector attempts to mark pixels which are at the center of elongated, winding shapes, producing thin, connected segments following the shapes of roads where they are well delineated. See the Appendix for details on the ribbon detector. The output of the ribbon center detector are lists of coordinates of pixels representing 8-connected

* Research supported by Grumman Data Systems

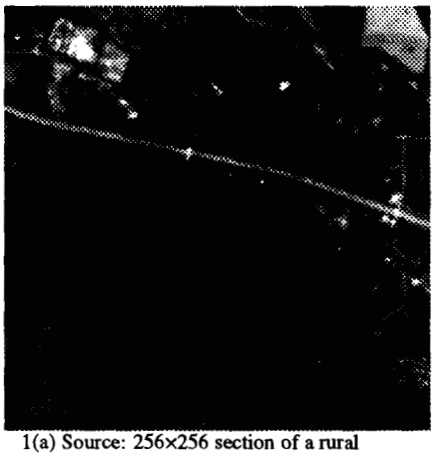
0-8186-2840-5/92 $03.00 © 1992 IEEE

curve segments corresponding to ribbon centers. These curve segments are further fitted by conic splines before being input to the linking program. Fitting pixel aggregates by splines produces a more concise representation and also extracts such feature as direction of a road segment. One of the reasons why we have chosen conic splines is that since conic segments do not have inflection points, filling a gap with a conic segment dose not introduce inflection points that are not inherent in the original data. Gaps with inherent inflection points, which are relatively uncommon along a road, are filled with line segments. Other advantages of using conic splines in edge linking are discussed in [9] . Our conic fitting algorithm is based on an algorithm presented by Pavlidis in 1983 [10] .

A key to our linking algorithm is the use of global information. In the results of the ribbon center detector, roads are typically broken and mixed with curve segments extracted from other features or background noise. When considered individually, there is little difference between a short road segment and a curve segment extracted from other features or noise. However, if all the curve segments in a picture are considered together, we see that road segments are distinguished by forming a smooth path between two points. In our algorithm, first all the curve segments are organized through a global data structure - a graph - then the road segments between two given points are connected through shortest path searching.

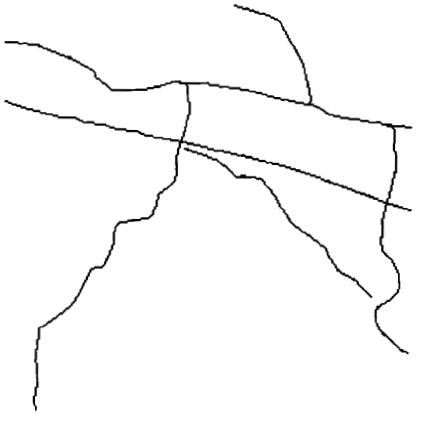
We sort the results of the ribbon center detector according to the degree of confidence. When the ribbon center finding algorithm is applied, a decision needs to be made on the threshold for the surface curvature. Generally speaking, if a smaller threshold is used more features can be picked up, but the result is also of lower confidence. In producing the set of curve segments on which the connected graph is to be built, the thresholds should not be chosen too small in order to keep certain level of reliability of the segments and a manageable number of nodes in the graph. However, the more detailed information contained in segments obtained using smaller thresholds should also be used to improve the accuracy of the linking result. In our algorithm two different sets of curve segments are generated. One set is generated using relatively large spatial curvature threshold and is referred to as the set of major segments. The other is generated using relatively small threshold and is referred to as the set of minor segments. The set of major segments is of higher confidence and is used to build the graph. The set of minor segments is used in one of the criteria in linking.

In the following sections we explain in detail each step of the algorithm.



1(a) Source: 256×256 section of a rural SPOT image

1(b) Ribbon centers



1(c) Result after linking

1(d) Linking result overlayed on 1(a)

Figure 1. Results of the algorithm.

3. The graph

In this section we give a formal definition of the graph used in our algorithm.

Let the graph be denoted by $G = (V, E)$, where $V = \{v_i\}$ is the set of vertices and $E = \{\{v_i, v_j\}, v_i, v_j \in V\}$ is the set of edges in the graph. G is an undirected graph. Each vertex $v_i \in V$ represents an end point of a curve segment. If an end point is shared by $k>1$ segments , which is the case at a joint or junction, then k different vertices are created - one for each segment. An edge $\{v_i, v_j\}$ is a member of E if and only if it satisfies one of the following conditions: (1) v_i and v_j are end points of the same curve segment. (2) v_i and v_j are not end points of the same curve segment but the distance between them is less then a threshold D. An edge satisfying condition (1) represents a curve segment. An edge satisfying condition (2) represents the gap between two curve segments (joints and junctions are considered as gaps with zero length). As we mentioned before, the graph is built on the set of major segments. In other words, each curve segment represented in the graph is a major segment. The result shown in figure 1 is obtained with $D = 40$ *pixels*.

The classic Dijkstra's algorithm [11] can be used to find the shortest path between any two connected vertices. The basic idea of our shortest path approach is that if the costs of the edges are assigned properly, then given two vertices representing the end points of a road, there is a good chance that the shortest path in between follows the road. Thus the road can be recovered by connecting the curve segments traversed by the shortest path.

We now explain how to fill a gap between two curve segments. In our algorithm, each gap with length larger than 0 is filled by a conic, if it can be fitted smoothly by a conic segment, and a line joining the two end points if it can not. We say a gap can be fitted smoothly by a conic segment if there exists a conic segment which joins the two curve segments separated by the gap and satisfies position and tangent continuity at the two end points. Based on properties of conic splines we developed the following criteria:

A gap can be fitted smoothly by a conic segment if and only if

(1) *There is no inherent inflection point along the gap or there is an inherent inflection point but it can be placed at one of the end points while keeping the tangent continuity.*

(2) *The overall direction change (the angle turned from the tangent at one end point to the tangent at the other end point) is less than 180 degrees.*

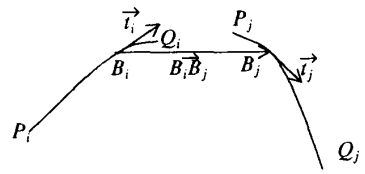


Figure 2. Criteria for smooth conic fitting of a gap

The first criterion is based on the property that a conic segment does not include any inflection points. The second criterion is a constraint commonly used for conic splines which implies that only those conic segments whose tangent direction change from one end to the other is less than 180 degrees are considered. As we shall see later, this constraint also matches the property that a road usually does not turn more than 180 degrees over a short distance.

There is a simple way to check the above criteria which also has the advantage of being tolerant to small shifts in positions caused by noise and error in previous processing. Let P_iQ_i, P_jQ_j be two curve segments and the gap between Q_i and P_j be the gap to be filled (figure 2). First we back up from the end points before analyzing the shape. This is because the gap was caused by some obstruction of the road image or contrast variation, and this causes the shape at an end point to be less accurate than near the center of the segment. Suppose the two points reached after backing up are B_i and B_j, instead of checking if there is a smooth conic fit between Q_i and P_j, we check if there is one between B_i and B_j. First we assign a direction to the curve assuming the gap is filled. Without loss of generality, let it be: $P_i \rightarrow B_i \rightarrow B_j \rightarrow Q_j$. Then we compute vector $\vec{t_i}$ which represents the tangent direction of segment P_iQ_i at point B_i, and vector $\vec{t_j}$ which represents the tangent direction of segment P_jQ_j at point B_j. Let $B_i\vec{B}_j$ be a vector pointing from B_i to B_j. It is easy to verify that criterion (1) is satisfied if and only if the angle turned from $\vec{t_i}$ to $B_i\vec{B}_j$ is of the same direction as the angle

turned from $\vec{B_iB_j}$ to $\vec{t_j}$ (both clockwise or both counter-clockwise); criterion (2) is satisfied if and only if the angle turned from $\vec{t_i}$ to $\vec{t_j}$ is less than Π. Figure 3 and figure 4 show examples of classification by this condition. Notice that the longer the back up length, the more tolerant the condition is to shifts in positions. Take the case in figure 3(c) for example, the gap would be classified as without smooth conic fit if a very small back up length were used. On the other hand, the case shown in figure 4(a) would be classified as having a smooth conic fit if a very large back up length were used. Our experiments show that a back up length of 5–10 pixels gives results matching human judgements. As we can see from the examples, human eyes tend to see smooth connections for gaps in figure 3 but not for gaps in figure 4.

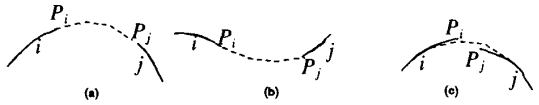


Figure 3. Examples of gaps with smooth conic fit

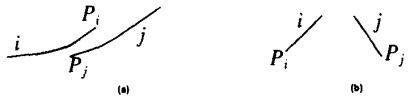


Figure 4. Examples of gaps without smooth conic fit

4. Costs for the shortest path searching

A major concern in the shortest path approach is to assign cost parameters properly so that the shortest path between two end points of a road indeed follows the road. To solve this problem, we first make the following four observations:

1. A gap is not as reliable as a curve segment (major segment), so the cost of a gap should be higher then the cost of a curve segment of the same length.
2. Roads are usually smooth in the sense that they do not have small wiggles, which suggests that a path with many turns should have higher cost then a path with fewer turns.
3. In an aerial image, grey scale values along a road usually do not change very much within a short distance. This suggests that a path with small grey level changes at the gaps should have lower cost then a path with larger grey level changes at the gaps.
4. A common cause of gaps between road segments is the contrast variation between the background and the road. To be more specific, a road segment would be missing from the set of major segments if its spatial curvature falls below the threshold used for major segments due to contrast variation. Since a lower threshold is used in generating the minor segments, such road segments are likely to be present in the set of minor segments. So a gap whose connecting segment coincides more often with the minor segments should have a lower cost.

In the following we explain in detail how the cost of an edge is computed following the above observations. Given an edge $\{v_i,v_j\}$, let the w's be weights which are real numbers greater than or equal to 0. Table 1 specifies the cost C_{ij} of the edge.

Table 1. Computation of cost for an edge

Input: Graph G, edge $\{v_i,v_j\}$, the source image, the grey scale image of minor segments.

Output:

The cost C_{ij} of the given edge.

Parameters:

w_{smooth} : weight for the breaking penalty of a gap with smooth conic fit.

$w_{non-smooth}$: weight for the breaking penalty of a gap without smooth conic fit.

w_{angle} : weight for the angle penalty.

$w_{grey-scale}$: weight for the grey scale penalty.

$w_{curvature}$: weight for the curvature penalty.

$$P_b = \begin{cases} L \cdot w_{smooth} & \text{if } \textit{there is a smooth conic fit} \\ L \cdot w_{non-smooth} & \text{if } \textit{there is no smooth conic fit} \end{cases}$$

$$P_a = \theta \cdot w_{angle}$$

$$P_g = d \cdot w_{grey-scale}$$

$$P_c = n \cdot w_{curvature}$$

$$P = P_b + P_a + P_g + P_c$$

Let L be the length of the curve/connecting segment.

$$C_{ij} = \begin{cases} L & \text{if } \{v_i,v_j\} \textit{ represents a curve segment} \\ L+P & \text{if } \{v_i,v_j\} \textit{ represents a gap} \end{cases}$$

End of Table

The cost of a real segment is simply the length of the segment. The cost of a gap is the length of the connecting segment of the gap plus the penalty P. There are four kinds of penalties: breaking penalty P_b, angle penalty P_a, gray scale difference penalty P_g and spatial curvature penalty P_c.

The breaking penalty P_b is $L \cdot w_{smooth}$ for a gap with a smooth conic fit and is $L \cdot w_{non-smooth}$ for a gap without a smooth conic fit, where L is the length of the connecting segment. Theoretically, w_{smooth} should be chosen much

smaller than $w_{non-smooth}$ because two pieces of curve segments are more likely to be broken parts of one road if they can be joined together smoothly by a conic segment. However, experimental results suggest that w_{smooth} and $w_{non-smooth}$ should not differ very much. The reason is that the judgement made beforehand on whether or not a gap can be fitted smoothly by a conic segment is not completely reliable due to noise.

The angle penalty $P_a = \theta \cdot w_{angle}$ is introduced based on the second observation. θ is the amount of total direction change in radius at a gap. For a gap with length greater than 0, θ can be computed through angles between the tangent direction right before the gap, the direction of the line joining the two end points and the tangent direction right after the gap. For a gap with 0 length which is actually a joint, θ is simply the angel between the tangents right before and right after the joint. Angle penalty proves to be very effective in reducing small wiggles introduced by noise in the shortest path.

The grey scale penalty for a gap is designed based on observation 3. For each gap, the average grey level of the two real segments separated by the gap is computed and taken as the expected grey level for the gap. Then, the average deviation d from the expected grey level is computed over the whole connecting segment of the gap and the grey scale penalty is assigned as $d \cdot w_{grey-scale}$.

According to the fourth observation, a gap whose connecting segment contains more pixels that lie on a minor segment should be assigned a lower cost than a gap that does not. For each pixel along the connecting segment of a gap, a small neighborhood around the pixel (eg., 3×3) is checked to see if it is passed through by a minor segment. The pixel is classified as a *hit* if it is, and a *non-hit* if not. For each gap, n is set to the number of pixels along the connecting segment that are *non-hit*, and the curvature penalty P_c is set to $n \cdot w_{curvature}$. The name "curvature penalty" is used because here we look at the spatial curvature information contained in the minor segments. A gap filled with more minor segments and thus having higher spatial curvature is assigned a lower cost.

Notice that the original grey scale image and the curvature information obtained during road center finding are used at this step in computing the grey scale penalty and curvature penalty. This is another difference between our algorithm and previous algorithms. In most of the previous algorithms, data flow along one direction, i.e., from low level descriptions to high level representations. Low level information is usually discarded completely once a higher level is reached. In our algorithm, the low level information of grey scale values and the curvature information generated by a low-level detector are used to help linking road segments in a high level graph operation. Experiments show that this strategy of "back tracking" helps to improve the accuracy of high level representations.

5. Experimental results

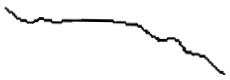
Figure 5 shows the effect of angle penalty and grey-scale penalty. Minor segments are not used in this example. The graph is built with parameters $D=60$ and back up length 5. Figure 5(c) shows the result of linking the top, horizontal road with $w_{grey-scale}$ and w_{angle} both set to 0. Figure 5(d) is the result of linking the same road with w_{angle} set to 80, which shows the smoothing effect of angle penalty. Figure 5(e) shows the result with $w_{grey-scale}$ set to 80. As we can see, the accuracy is improved by the introduction of grey scale penalty.

Experiments also show that for images of very poor quality, the curvature penalty based on minor segments usually works better than the grey scale penalty based on raw grey scale values. This is not surprising if we make a closer analysis of these two kinds of information. The raw grey scale values are isolated, absolute values which contain no explicit information about the neighborhood, while the minor segments represent sequences of points of local maximum of grey scale surface curvature, which is contrast independent. Given a minor segment we know that it is a ribbon like region and thus it is likely to be part of a road if supported by major segments around it. However, given pixels with similar grey scale value as surrounding major segments we do not have such confidence because other features such as buildings may also have similar grey scale values. Also, if a road is thinner than the size of a pixel, the grey scale value of a pixel passed through by the road is dependent on the percentage of the area of the pixel covered by the road. The observation that pixels along a road have nearly uniform grey scale value dose not hold in this case. On the other hand, since a Gaussian smoothing filter is applied before grey scale surface curvature is computed, such a thin road may still be recognized as a ribbon like feature using the curvature information.

Figure 6 shows the effect of using spatial curvature information in linking weak roads. Here the major segments are obtained with the size of Gaussian filter $\sigma=1.0$ and the threshold for spatial curvature $t_{curve}=5.0$, and the minor segments are generated with $\sigma=1.0$ and $t_{curve}=3.0$. The graph is built with $D=60$ and back up length 5. The road on the left part of the source image is particularly weak and can not be recovered using the information from the set of major segments alone even when grey scale penalty is applied, as shown in figure 6(d). After introducing the curvature penalty by setting $w_{curvature}$ to 40 instead, the more detailed curvature information shown in figure 6(c) is considered in linking and the road is properly recovered, as shown in figure 6(e).

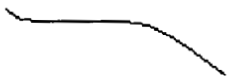
5(a) Source (128×128)

5(b) Ribbon centers



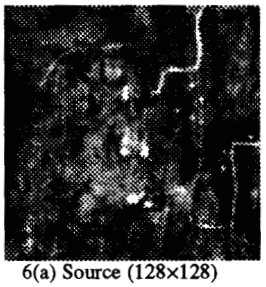
5(c) Result of linking one road with $w_{smooth}=9$, $w_{non-smooth}=10$, $w_{angle}=0$, $w_{grey-scale}=0$, $w_{curvature}=0$.

5(d) Result of linking the same road with $w_{smooth}=9$, $w_{non-smooth}=10$, $w_{angle}=80$, $w_{grey-scale}=0$, $w_{curvature}=0$.



5(e) Result of linking the same road with $w_{smooth}=9$, $w_{non-smooth}=10$, $w_{angle}=0$, $w_{grey-scale}=80$, $w_{curvature}=0$.

Figure 5. An example showing the effect of grey scale penalty and angle penalty.



6(a) Source (128×128)

6(b) Major segments

6(c) Minor segments

6(d) Result of linking one road without curvature penalty. Parameters: $w_{smooth}=9$, $w_{non-smooth}=10$, $w_{angle}=100$, $w_{grey-scale}=40$, $w_{curvature}=0$.

6(e) Result of linking the same road with curvature checking. Parameters: $w_{smooth}=9$, $w_{non-smooth}=10$, $w_{angle}=100$, $w_{grey-scale}=0$, $w_{curvature}=40$.

Figure 6. An example showing the effect of using curvature information. Comparing 6(e) to 6(d) it can be seen that the introduction of curvature penalty pulls the shortest path towards the road.

6. Parameter analysis

In this section we give an analysis on how various parameters affect the performance of our algorithm. The analysis is based on tests using SPOT images.

Two parameters are involved in the ribbon center finder: the size of the Gausian filter σ and the threshold for the spatial curvature t_{curve} (Appendix). The choice for σ is dependent on the width of the road - it should be about half the width of the road (in pixels). For an image containing roads with various widths (e.g. figure 1) it should be set for the thinnest road to be recovered. The value of σ is 0.5 for figure 1(b), 0.8 for figure 5(b), 1.0 for figure 6(b) and 6(c). The threshold t_{curve} depends on the contrast of the road and σ. It should be increased with increasing contrast or decreasing σ. The value of t_{curve} is 10.0 for figure 1(b), 6.0 for figure 5(b) and and 5.0 for figure 6(b). Minor segments, when needed, should usually be obtained with t_{curve} set to about half the value used for major segments.

The only parameter involved in conic fitting is the error bound. Since the accuracy is the major concern here, we suggest using a small error bound. An error bound of one pixel is used in all our experiments, including those shown in this paper.

Two parameters are involved in graph construction: D and back up length. D should be chosen so that it is larger than the length of the longest gap between neighboring road segments. Our experiments show that in most cases once this condition is satisfied, the result of the linking program stays stable for different values of D over a large range (around 60 pixels). So D should be chosen in a conservative way: it should be set to the estimated maximum length of gaps between neighboring road segments plus about 30 pixels to compensate for possible error in the estimation. Up to this stage, the estimation is done manually. As to the back up length, as mentioned before, a value of 5–10 pixels usually gives results that matches human judgement on whether two curve segments can be connected smoothly.

Five cost parameters are involved in the shortest path linking. Our experiments show that w_{smooth} and $w_{non-smooth}$ can be usually fixed to around 10. w_{angle} usually ranges from 30 to 100, larger for relatively flat roads and smaller for roads with more turns. In many cases satisfactory results can be obtained without using grey scale penalty and curvature penalty. For example, the cost parameters used to generate the result in figure 1(c) are: $w_{smooth}=9$, $w_{non-smooth}=10$, $w_{angle}=60$, $w_{grey-scale}=0$ and $w_{curvature}=0$. When the grey scale penalty and curvature penalty are necessary, mostly for roads with poor visibility, $w_{grey-scale}$ and $w_{curvature}$ usually range from 20 to 100. These two parameters are relatively difficult to predict and more interaction may be needed to find the proper values.

The choice of end points normally do not affect the result so long as the points are truly end points of a road. For a road with very sharp turns it is sometimes necessary to use nodes at the turns as intermediate end points and link the road as more than one pieces that join at these points. For example, the rightmost road of figure 5 is linked using an intermediate end point at the lower left conner. All the other roads shown in the examples in this paper are linked without using intermediate end points.

7. Implementation

The algorithm has been implemented in an interactive map generating system on a SPARC station in C. An X window interface has been built which allows interactive linking of roads with different settings of cost parameters. The end points of roads are pointed to by user on a window where conic approximations to the curve segments (major segments) are displayed and the resulting shortest paths are shown on another window. The original image and minor segments can also be displayed at the same time on separate windows. Cost parameters can be adjusted at run time, without rebuilding the graph. Furthermore, they can be adjusted separately for different roads within the same image, which means the user needs to consider only one road at a time, avoiding the sometimes impossible task of finding the set of parameters that fits all the roads in one image. Many real images have been processed by this system and it proves to be a practical system for speeding up map construction. For the source image shown in Figure 1(a), ribbon center detection takes about 15 seconds; conic fitting takes about 3 seconds; graph construction takes about 15 seconds; and shortest path searching for one road takes less than 1 second.

8. Acknowledgements

We thank Li Wang for useful discussions on the gray scale shapes, and Lynne Shigley for providing the program which was the starting point for the ribbon finder. We also thank Herb Tesser of Grumman Data Systems for his overall support of our work.

Appendix: Ribbon center detection

The ribbon center detector accepts grayscale images, and marks pixels at the center of elongated, winding ribbon shapes. Design goals include appropriate connectivity of the detected ribbons, resistance to noise, and consistent performance under varying contrast.

Our approach is based on detection of ridgelines in grayscale space. Intuitively, the method can be visualized by aligning the image parallel to the floor, with darker pixel squares plotted lower, and lighter squares

plotted higher. In geographic terminology, this yields a terrain where ridgelines correspond closely to roads in the image.

The technical version of this idea is *principal curvature*. At point p on the grayscale surface, let $\vec{n}$ be the surface normal through point p. Any plane A containing $\vec{n}$ intersects the grayscale surface in a plane curve C passing through p. Call the curvature of C at p induced by plane A κ_A. As the direction of A is varied, κ_A has one maximum and one minimum value, occurring in orthogonal directions. We call these directions the *principal axes* at p, and the extreme values the *principal curvatures*.

For algorithmic implementation, we apply the method of eigenvectors/eigenvalues of the Hessian following Haralick Watson and Laffey [12,13] . Here, directional second derivatives are an analogue to the curvature κ_A. The directional second derivative through point p is extremized in two orthogonal directions. The directions are the eigenvectors of the Hessian matrix at p, and the corresponding eigenvalues are the extreme values. When the gradient of the surface is zero at p, the extreme values and directions of the second derivative are the principal curvatures and principal axes.

There are a number of crucial engineering details needed in passing from this strategy which is based on continuous mathematics to a discrete implementation which will work robustly on real data. An initial Gaussian smoothing improves performance on most data by filtering high frequency noise and providing smoother ridges peaks. Interpolation is used in finding ridge peaks, to obtain thin ridgelines. See [14] for technical details of ribbon detection.

References

1. R. Nevatia and R. Babu, "Linear Feature Extraction and Description," *CVGIP*, vol. 13, pp. 257-269, 1980.
2. A. Huertas, W. Cole, and R. Nevatia, "Detecting Runways in Aerial Images," *Proceedings: AAAI-87*, pp. 712-717, July 1987.
3. A. Aviad and P. D. Carnine Jr, "Road Finding for Road-Network Extraction," *Proceedings: IEEE CVPR*, pp. 814-819, June 1988.
4. S. Vasudevan, R. L. Cannon, and J. C. Bezdek, "Heuristics for Intermediate Level Road Finding Algorithms," *Computer Vision, Graphics and Images Processing*, vol. 44, pp. 175-190, 1988.
5. D. M. McKeown and J. F. Pane, "Alignment and Connection of Fragmented Linear Features in aerial imagery," *Preceedings, IEEE CVPR*, 1985.
6. M. L. Zhu and P. S. Yeh, "Automatic Road Network Detection on Aerial Photographs," *Proceedings, IEEE CVPR*, 1986.
7. M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," *CGIP*, vol. 15, pp. 201-223, 1981.
8. D. M. McKeown and J. L. Denlinger, "Cooperative Methods for Road Tracking in Aerial Imagery," *Proceedings: IEEE CVPR*, pp. 662-672, June 1988.
9. V. S. Nalwa and E. Pauchon, "Edgel Aggregation and Edge Description," *CVGIP*, vol. 40, pp. 79-94, 1987.
10. Theo Pavlidis, "Curve Fitting with Conic Splines," *ACM Transactions on Graphics*, vol. 2, no. 1, pp. 1-31, January 1983.
11. E. W. Dijkstra, "A note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.
12. Robert M. Haralick, Layne T. Watson, and Thomas J. Laffey, "The Topographic Primal Sketch," *International Journal of Robotics Research*, vol. 2, no. 1, pp. 50-72, Spring 1983.
13. Layne T. Watson, Thomas J. Laffey, and Robert M. Haralick, "Topographic Classification of Digital Image Intensity Surfaces Using Generalized Splines and the Discrete Cosine Transform," *CVGIP*, vol. 29, pp. 143-167, 1985.
14. B. Sakoda, J. Zhou, T. Pavlidis, and E. Joseph, "An Address Recognition System Based on Feature Extraction from Gray Scale," *Image Analysis Laboratory Tech Report 06.08.92*, Computer Science Department, SUNY at Stony Brook, June 1992.