Linear feature detection of rural imagery using multiresolution filters

Samuel Peter Kozaitis
Rufus H. Cofer

Lineal feature detection of rural imagery using multiresolution filters

Samuel P. Kozaitis, and R. H. Cofer
Florida Institute of Technology
Department of Electrical and Computer Engineering
150 W. University Blvd.
Melbourne, FL 32901

ABSTRACT

We detected roads in aerial imagery using a method based on lineal feature detection. Our method used the products of wavelet coefficients at several scales were to identify and locate lineal features. Using our approach effectively increased the size of the region we examined when looking for possible road pixels, and decreased the probability of false positive road pixels. Then, we used a shortest path algorithm to link road pixels to form road networks. Our approach restricted possible road network solutions based on the initial detection of road pixels. We found that our approach leads to an effective method for detecting roads in aerial imagery. The method is general and can be applied to other features in imagery.

Keywords: dyanmic programming, feature detection, multiresolution, road detection, wavelet transform

1. INTRODUCTION

Many approaches to lineal feature extraction involve the use of low-level vision methods using local neighborhoods followed by a symbolic approach using a global criterion. For example, to detect roads, several approaches use local neighborhoods to detect edges in an image, then a tracking method is used to find roads.[1-12] Often, methods use conventional or custom filters based on edge or line detection. Because the global structure of a road network in usually not considered initially, errors in determining road pixels are unavoidable. A method that would use more global information in the detection of road pixels or segments could form the basis of a particularly effective approach to road detection.

Multiresolution approaches effectively use larger neighborhoods than those at just one resolution or scale. Generally, these methods are based on knowledge that different characteristics of objects can be best detected in different scales. Features such as roads can be found at different resolutions then combined using a graph or rule-based method.[13-16] Using the results from different resolutions and knowledge of roads, hypotheses for road segments may be generated. Then, segments may be grouped into larger segments or networks. The wavelet transform provides a convenient method for relating information at different image resolutions.[17] For example, it has been shown that the evolution of wavelet local maxima across scales characterizes the local shape of structures.[18] Furthermore, a robust method has been developed to detect edges in noisy images using information across scales.[19-20] Such an approach uses relatively large neighborhoods for feature detection when compared to conventional approaches and could be useful if applied to road detection. However, this approach has not been used much for lineal features.[21]

We use a wavelet-based multiresolution approach to detect lineal features in an image. We examine the product of wavelet coefficients across scales to identify potential features, and combine this information with the original image to perform detection. In addition, we describe a new linking method to form road networks that is well suited to our approach. In our linking algorithm, we use the information from the multiresolution feature detection algorithm to restrict the possible road solutions.

2. THEORY

2.1 Wavelet transform

The wavelet transform maps a signal in the space domain into a scale-translation domain using scaled and translated versions of a mother wavelet. The discrete one-dimensional wavelet transform of the function *f(x)* with respect to a mother wavelet $w_{jk}(x)$ is

$$b_{jk} = \sum_{x} f(x) w_{jk}(x), \tag{1}$$

where b_{jk} are the collection of wavelet coefficients, and *j* and *k* indicate the scale and shift of the wavelet. A dyadic family of wavelets are often used so the wavelet is written as $w_{jk}(x) = 2^{-j/2} w_{jk}(2^{-j}x - 2^{j}k)$, where $j = 0, 1, 2, \ldots$. In this case, the wavelet is dilated by a factor of two at each increasing scale. The amplitude of the wavelet decreases by a factor of $\sqrt{2}$ at each increasing scale, so scaled wavelets have the same energy. In addition, the wavelet is shifted by a factor of two at each increasing scale.

In the wavelet transform, the input function *f(x)* is being compared to a wavelet $w_{jk}(x)$ through a correlation or projection. A wavelet domain coefficient is computed for each particular scale and shift value; the wavelet coefficient is the correlation coefficient between *f(x)* and $w_{jk}(x)$. Determining these coefficients is the wavelet transform. In other words, a function is approximated by a weighted sum of the scaled and shifted mother wavelet. When summed together the original signal is obtained. In addition, the wavelet transform is linear and superposition holds.

The inverse wavelet transform reconstructs the original function by summing weighted, scaled and shifted versions of the mother wavelet. The weights are the wavelet coefficients b_{jk}. The inverse wavelet transform sums over the two-dimensional scale-translation space as

$$f(x) = \sum_{j,k} b_{jk} w_{jk}(x). \tag{2}$$

2.2 Wavelet coefficients across scales

Mallat showed that the evolution across scales of wavelet transform coefficients is bounded by the Lipschitz regularity of the signal[18]. The Lipschitz regularity of a signal feature can be estimated from the exponent of the absolute value of the wavelet transform across scales by,

$$|b_{jk}| \leq A2^{j(\alpha+1)}, \tag{3}$$

where $A > 0$, and α is the Lipschitz regularity. The Lipschitz regularity can be estimated from the slope of $\log_2|b_{jk}|$ as a function of *j*. The estimate of α gives an indication of the decay of the wavelet transform maxima over a given range of scales. For signals with wavelet coefficients that decay with increasing *j*, the Lipschtiz regularity has values $-1 \leq \alpha < 0$.

The Lipschitz exponent has been shown to be useful for discriminating edges and singularities.[19] However, for narrow linear features, the Lipschitz regularity has a relatively large magnitude which makes it difficult to discriminate from clutter or noise. In addition, when the wavelet coefficients decay rapidly across scales, the number of scales that can be used to estimate the Lipschitz exponent is reduced. This further makes the discrimination of narrow features difficult. In addition, the Lipschitz regularity does not depend on the initial response of a feature. Therefore, features with a large response are weighted equally to those with a small response because only their slopes are considered.

If feature detection is based on slope alone, then noise or clutter may significantly affect the output. A more advantageous method is one that considers the responses of both the slope and magnitude of the wavelet coefficients at the scale with the maximum response. Therefore, we considered multiscale products of wavelet coefficients. We considered features for further processing if they had a significant response at a particular scale, even though their regularity seemed large. We determined that the product of two scales is related to the Lipschitz regularity by,

$$p = b_{jk}{}^2 \, 2^{j\alpha}. \tag{4}$$

Figure 1 shows the relationship between the products and the Lipschitz exponent for values of $-1 < \alpha < 1$. Values of $\alpha < 0$, and $\alpha > 0$ represent wavelet coefficients that decrease and increase with scale respectively.

The results show that for large values of a product, it varies approximately linearly with the Lipschitz exponent as $p = 1 + 1/2\alpha$. As the product decreases the relationship is still approximately linear, but a smaller multiple of α. For small values of the product, the product is almost independent of α. The data shows that the larger the product, a wider range of slopes is tolerated.

2.3 Multiresolution feature detection

We identified potential features by forming the products of wavelet coefficints across different scales. For example, to detect features using scales $j = 1$ and 2, a product $C_{12}(k)$ is formed between a signal's nondownsampled wavelet transform coefficients of the two scales, b_{1k} and b_{2k}. A feature is identified at a position where $|b_{1k} \cdot b_{2k}| > T$, where T is a threshold and is used to create a binary. After combining coefficients from different levels, potential feature locations are saved in a spatial binary mask.

Because feature locations may shift according to scale, this method could produce errors as more scales are considered. Therefore, we formed the product between b_{1k} and shifted versions of b_{2k}, and used the maximum values at each location of b_{1k} as the product between scales.[19] For a feature to shift a maximum of s locations, and considering the product of n scales, this method uses the product,

$$C_{1n}(k) = \max\{b_{1,k} b_{p,k+t}\} \tag{5}$$

where $p = 2, 3, \ldots n$, $t = 0, \pm 1, \pm 2, \ldots \pm s$, and $(2s+1)^{n-1}$ signal multiplications are needed to determine the feature locations.

To reduce the number of computations, direct multiplications are approximated by forming the products between two scales at a time. For example, using wavelet coefficients from four scales b_{1k}, b_{2k}, b_{3k}, and b_{4k}, we found features from the two largest scales b_{3k}, and b_{4k}. Then, the result was combined with b_{2k}. The result of that step was then combined with b_{1k} to determine feature locations. For n scales, the number of signal multiplications is $(2s+1)(n-1)$.

The individual wavelet transforms in the horizontal and vertical directions can be used to extend the 1-D feature extraction algorithm to images. The block diagram of the algorithm for images is shown in Fig. 2 using four scales of the wavelet transform. First, the wavelet transform is applied to the input image in both the horizontal and vertical directions as shown in Fig. 2(a). Then, the vertical and horizontal results are produced separately and labeled W_{cv} and W_{ch} respectively in Fig. 2(b). The binary feature location images W_h and W_v in the horizontal and vertical directions are determined from W_{cv} and W_{ch} by thresholding. Features are found by forming the product of the binary masks and the wavelet coefficients from the smallest scale as indicated in Fig. 2(d). Finally, the modulus is determined from the images in the horizontal and vertical directions as,

$$M(x,y) = \sqrt{(\mathrm{W_v} \cdot \mathrm{b_{1v}})^2 + (\mathrm{W_h} \cdot \mathrm{b_{1h}})^2} \, . \tag{6}$$

3. EXPERIMENT

To help characterize our approach for all line angles, we examined the detection of pixels of circles embedded in noise. We used images of circles of 147 pixels in diameter of 1 and 7 pixel widths. The mexican-hat wavelet because it has been found useful for detecting lines in an image. We found the mean-square error (MSE) between the noisy circle images processed by our algorithm and the ideal result had there been no noise. Furthermore, we compared the results to those using only one scale. The results for a single scale, and multiple scales are shown in Fig. 3 for a 1 pixel wide circle. The results shown in Fig. 3(a) indicate the total MSE/pixel was significantly reduced when using multiple scales. Fig. 3(b) shows the results when the MSE is calculated only over the circle region. The results show that the MSE/pixel are similar for all combination of scales, but that using more scales tends to increase the MSE. Fig. 3(c) shows the results when the MSE is calculated only over the region not occupied by the circle. The results show that the MSE/pixel are significantly less when using multiple scales. It seems that in this case, increasing the number of scales increases the error in detecting line pixels by a small amount, but significantly decreases the error in potential false alarms, especially at low SNRs.

The results for a seven-pixel wide circle are shown in Fig. 4. The best single scale results where shown for j = 3, so we compared our algorithm's results to that scale. The results here are similar to that of the single-pixel wide case. It can be seen somewhat more clearly that at low SNR, the MSE in the single-scale case increases rapidly, while the multiscale case does not show the same behavior.

Two satelllite images of 256 x 256 pixels are shown in Fig. 5. They were both processed with $j = 1$ and the results shown in Fig. 6. They were both processed with $j = 1$, 2, 3, and 4, and the results shown in Fig. 7. Visually, the differences between the single-scale and multiscale approach can be. It appears that the road pixels are; however, the image processed with the single scale has a significant amount of clutter when compared to the multiscale result. Note that the results here only select candidate road pixels.

4. LINKING

4.1 Approach

We used a computer-assisted approach to link pixels to find road networks. The result of the previous lineal detection process assigns the probability to a pixel that it is part of a road. To create a road using our approach, a user selects a start and end pixel with a mouse, and the road is found between the two pixels. The path drawn between the start and end pixels is the shortest path, where shortest path is defined as the sum of pixel values. In our approach, we use the information from the multiresolution pixel detection scheme described earlier to restrict possible paths.

Dynamic programming methods for finding shortest-paths between two pixels have been very successful in detecting long, line-like features in images. Dynamic programming algorithms are simple to program and very efficient computationally, and are guaranteed to find a globally optimal path. Dijkstra's algorithm solves the problem of finding the shortest path from a point in a graph to a destination. It turns out that one can find the shortest paths from a given source to all points in a graph at the same time, hence this problem is sometimes called the single-source shortest path problem. This problem is related to the spanning tree problem. The graph representing all the paths from one vertex to all the others must be a spanning tree – it must include all vertices. There are no cycles, as a cycle would define more than one path from the selected vertex to at least one other vertex.

After line pixel detection, we divide the image into road and nonroad regions. Then, we use Dijkstra's algorithm to find roads. By restricting Dijkstra's algorithm to operate on a limited portion of the image, improved accuracy and a decrease in execution time can be obtained.

4.2 Road and nonroad regions

We divide our preprocessed image into possible road and nonroad regions. The significance is that identification of possible road regions restricts the possible solutions for a road network. Therefore, our method operates only on limited regions of the image, which can significantly reduce the execution time. We basically blur the preprocessed image until a possible solution exists, then we use Dijkstra's algorithm to find the final solution. The process is shown schematically in Fig. 8. We initially filter the image containing the preprocessed road pixels $M(x,y)$ using a low-pass filter $f^i(x,y)$according to,

$$M'(x,y) = \begin{cases} M(x,y) & if\ M(x,y) < M(x,y) * f^{i}(x,y) \\ M(x,y) * f^{i}(x,y) & if\ M(x,y) > M(x,y) * f^{i}(x,y) \end{cases}, \tag{7}$$

where the *i* indicates an iteration described in the next section. In short, if a pixel value is larger than the filtered pixel value, it will be replaced, otherwise it won't. Then we threshold the filtered image according to

$$MT(x,y) = \begin{cases} \infty & if \quad M'(x,y) < \sum_{x}\sum_{y} M'(x,y) \\ M'(x,y) & if \quad M'(x,y) > \sum_{x}\sum_{y} M'(x,y) \end{cases}, \tag{8}$$

where the threshold value is the average value of *M´*(*x*,*y*). If a pixel value of the filtered image is lower than the average, its value is replaced by a large value indicating a nonroad region, otherwise, it retains its value. In this way the image is separated into road and nonroad regions. An example of *MT*(*x*,*y*) from on of the images from Fig. 5 is shown in Fig. 9. The nonroad regions are indicated in white and do not allow possible road networks based on this iteration.

The next step in the linking process is checking for connectivity. This step consists of searching for at least one path from the start to end pixel that does not contain any nonroad pixel. If a path can be drawn between both the start and end pixels given by an operator that does not contain a nonroad pixel, then connectivety has been established. Then the algorithm proceeds to the next step. If connectivity has not been established, then the image *M*(*x*,*y*) is filtered again with $f^i(x,y)$. However, for increasing *i*, the filter $f^i(x,y)$ contains fewer higher spatial frequencies and blurs the image more. This process is repeated until connectivety exists. Once connectivity has been established, then the product of *M*(*x*,*y*) and the image containg road and nonroad regions is formed for use by Dijkstra's algorithm.

4.3 Dijkstra's Algorithm for line detection

To apply Dijkstra's algorithm, we have to change an image to a weighted directed graph. The graph is a collection of vertices, directed edges, and weights. The vertices relate to pixels, directed edges relate to adjacent pixels, and weights relate to pixel values. To get vertices, the row and column position of each pixel converts to vertices V using, $V = x \times column + y$, where *x*, and *y* are the row and column coordinates of the pixel, and *column* is the number of columns in the image. Each pixel has eight neighbor pixels except the pixels on the outer edge of an image. To get a directed edge, we convert connected eight neighbor pixels to eight directed edges. Dijkstra's algorithm searches for the minimum cost value between the starting and ending vertices. Therefore, the output of the feature detection portion is converted to a negative image. Then, the next pixel value is the weight of the graph except diagonally connected pixel values. To assign these weights, the diagonal pixel values are multiplied by $\sqrt{2}$. Using this approach, the shortest path between two selected points can be found.

We compared our method to Dijkstra's algorithm in terms of execution time and accuracy for one of the images shown in Fig. 5. We identified the two main roads in the image and showed the results using our algorithm, and Dijikstra's in Fig. 10(a) and (b) respectively. The conventional Dijkstra's algorithm took a factor of 1.9 longer than our method in terms of execution time. In addition, the results show that a portion of the road was not detected properly. Using the conventional approach resulted in 78.9% of the road pixels being identified correctly, while our approach detected 93.5% of the road pixels correctly.

5. CONCLUSION

We detected roads in aerial imagery the products of wavelet coefficients at several scales were to identify and locate lineal features. Using our approach effectively increased the size of the region we examined when looking for possible road pixels, and decreased the probability of false positive road pixels. The low number of false positives was particularly useful for the shortest path algorithm used to link road pixels to form road networks. By restricting the region of possible road network solutions, we improved the accuracy and decreased the execution time of the algorithm.

REFERENCES

1. R. Bajcsy, and M. Tavakoli, "Computer recognition of roads from satellite pictures," *IEEE Trans. Systems Man Cybernet*., vol. **6**, 623-637 (1976)
2. R. Nevatia, and K. R. Babu, "Linear feature extraction and description," *Computer Graphics and Image Proc*., vol. **13**, 257-269 (1980)
3. 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," *Computer Graphics and Image Proc*., vol. **15**, 201-223 (1981)
4. J. M. Canning, J. J. Kim, N. S. Netanyahu, and A. Rosenfeld, "Symbolic pixel labeling for curvilinear feature detection," Pattern Recognition Lett., vol. **8**, 299-310 (1988)
5. D. M. McKeown Jr., and J. L. Denlinger, "Cooperative methods for road tracking in aerial imagery, *in Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 662-672 (1988)
6. N. Merlet, and J. Zerubia, "New prospects in line detection by dynamic programming," *IEEE Trans. Pattern Anal. Machine Intel.*, vol. **18**, 426-431 (1996)
7. D. Ziou, "Line detection using an optimal IIR filter," *Pattern Recognition*, vol. **24**(6), pp. 465-478 (1991)
8. M. Petrou, "Optimal convolution filters and an algorithm for the detection of wide linear features," *IEE Proceedings-I*, vol. **140**(5), (1993)
9. F. Tupin, H. Maitre, J. Mangin, J. Nicolas, and E. Pechersky, "Detection of linear features in SAR images: application to road network extraction," *IEEE Trans. Geosci. Remote Sensing*, vol. **36**(2), 434-453 (1998)
10. C. Zhang, S. Murai, E. P. Baltsavias, "Road network detection by mathematical morphology," Bulletin SFPT, 94, (1999)
11. D. Geman, and B. Jedynak, "An active testing model for tracking roads in satellite images," *IEEE Trans. Pattern Anal. Machine Intel.*, vol. **18**, 1-14 (1996)
12. J. C. Trinder, and Y. Wang, "Automatic road extraction from aerial images," *Digital Signal Processing*, vol. **8**, 215-224 (1998)
13. H. Mayer, "Automatic knowledge based extraction of objects of the real world from scanned maps," *Int. Archives of Photogrammetry and Remote Sensing*, vol. **30**(3/2), 547-554 (1994)
14. A. Baumgartner, C. Steger, H. Mayer, W. Eckstein, and H. Ebner, "Automatic road extraction based on multi-scale, grouping, and context," *Photogrammetric Engineering & Remote Sensing*, vol. **65**(7), 777-785 (1999)
15. C. Heipke, C. Stegar, and R. Multahmmer, "A hierarchical approach to automatic road extraction from aerial imagery," *Proc.* SPIE 2486, 222-231 (1995)
16. I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Stegar, and A. Baumgartner, "Automatic extraction of roads from aerial images based on scale space and snakes," *Machine Vision and Applications*, vol. **12**, 23-31 (2000)
17. S. G. Mallat, "Multifrequency channel decompositions of images and wavelet models," *IEEE Trans. Acoustics Speech, and Sig. Proc*., vol. **37**(12), 2091-2110 (1989)
18. S. G. Mallat, and S. Zhong "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Anal. Machine Intel.*, vol. **14**(7), 710-732 (1992)
19. Y. Lee, and S. P. Kozaitis, "Multiresolution gradient-based edge detection in noisy images using wavelet domain filters," *Optical Engineering*, vol. **39**(9), 2405-2412 (2000)
20. Y. Xu, J. B. Weaver, D. M Healy Jr., and J. Lu, "Wavelet transform domain filters: a spatially selective noise filtration technique," *IEEE Trans. Image Process.*, vol. **3**(6), 747-758 (1994)
21. S. P. Kozaitis, S. Udomhunsakul, R. H. Cofer, A. Agarawal, and S-W. Song, "Linear feature detection using multiresolution wavelet filters," *in SPIE Proc*. 4741B, *Geo-Spatial Image and Data Exploitation Developments and Applications III*, paper 39 (2002)

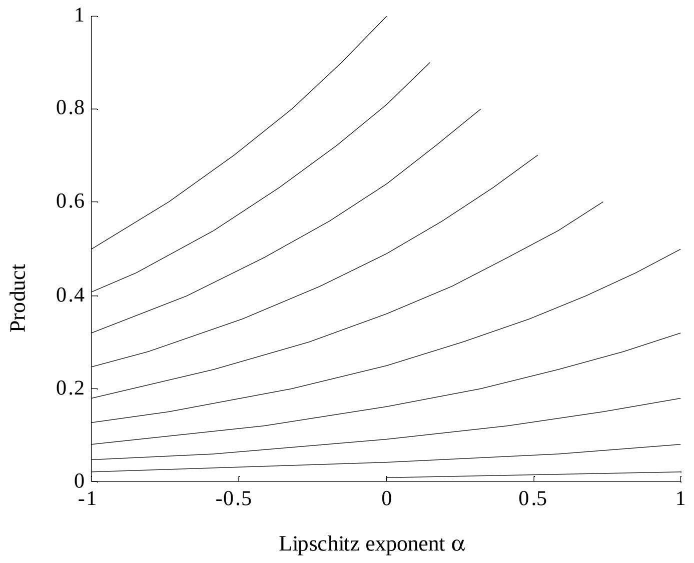


Figure 1 Relationship between products and Lipschitz exponent

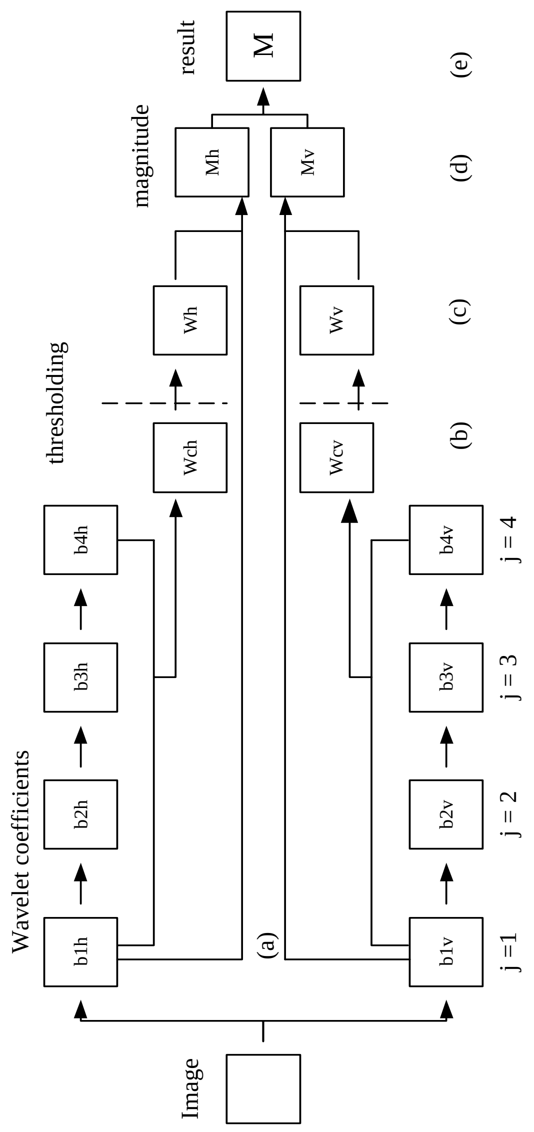
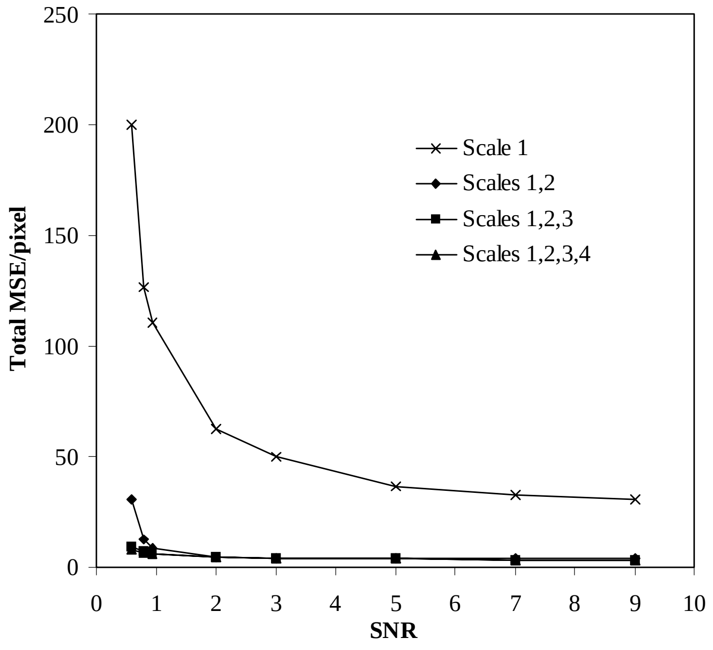
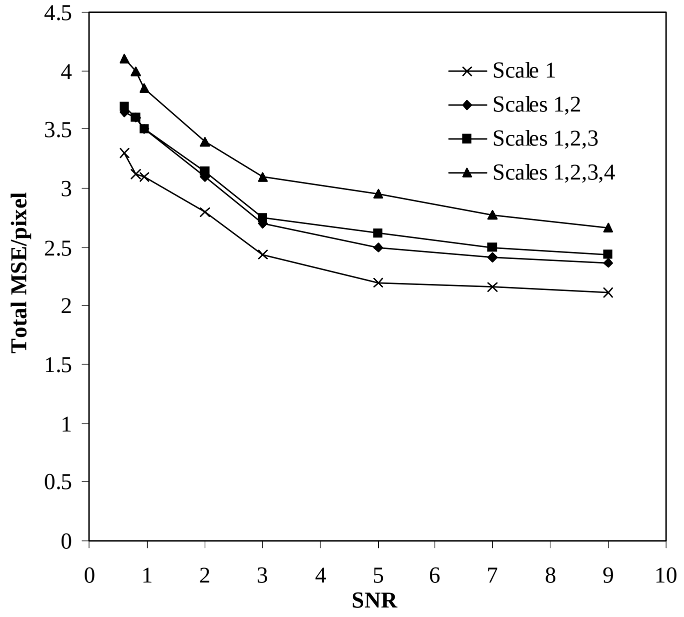


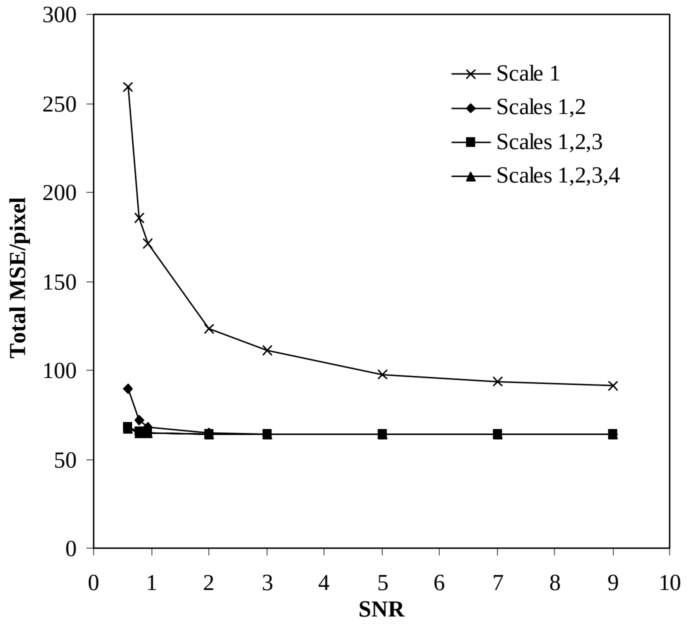
Figure 2 Block diagram of image feature detection algorithm (a) wavelet transform coefficients in horizontal and vertical directions for j = 1, 2, 3, 4 (b) results from combining different scale information (c) binary feature location (d) product of binary mask and wavelet coefficients (e) result.


Scale 1
Scales 1,2
Scales 1,2,3
Scales 1,2,3,4
Total MSE/pixel
SNR
0
50
100
150
200
250
0
1
2
3
4
5
6
7
8
9
10

(a)


Scale 1
Scales 1,2
Scales 1,2,3
Scales 1,2,3,4
Total MSE/pixel
SNR
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
1
2
3
4
5
6
7
8
9
10

(b)



(c)

Figure 3 MSE/pixel for various combinations of scales as a function of SNR for an image of a circle of 1 pixel in width (a) entire image (b) circle area only (c) noncircle area only.

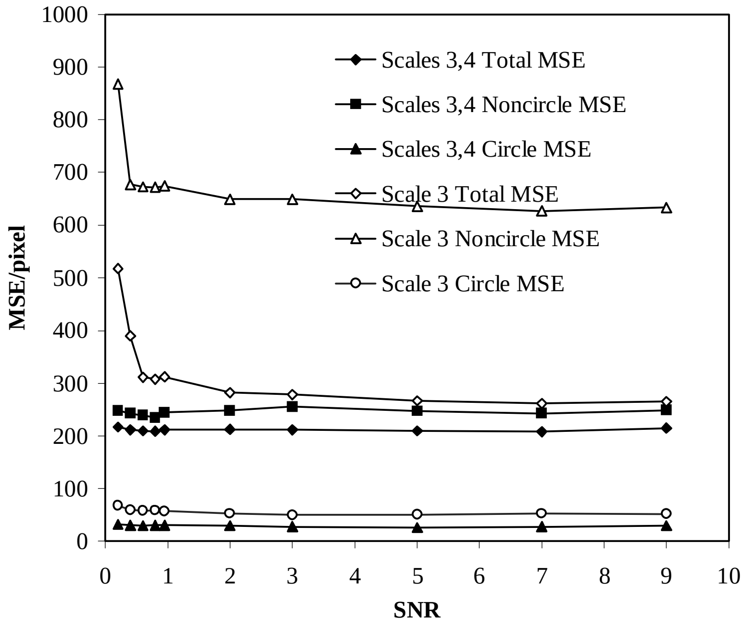


Figure 4 MSE/pixel for combination of scales 3 and 4 as a function of SNR for an image of a circle of 7 pixels in width.

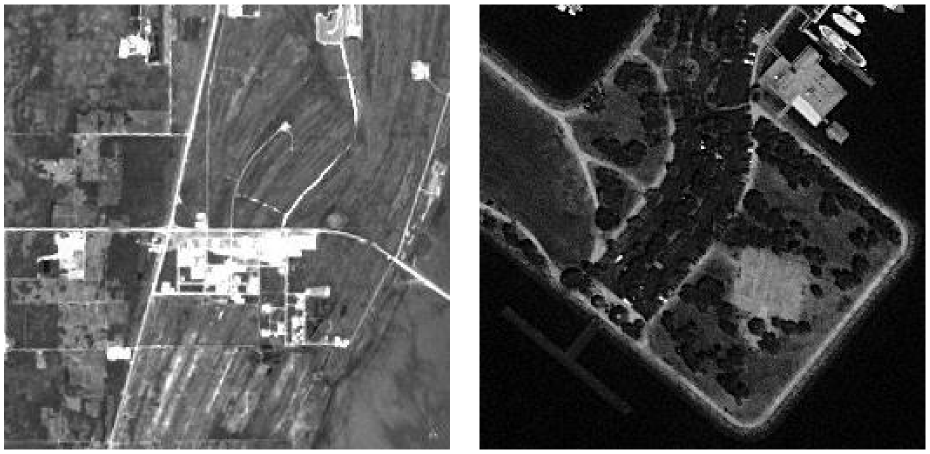


Figure 5 Images used in experiments.

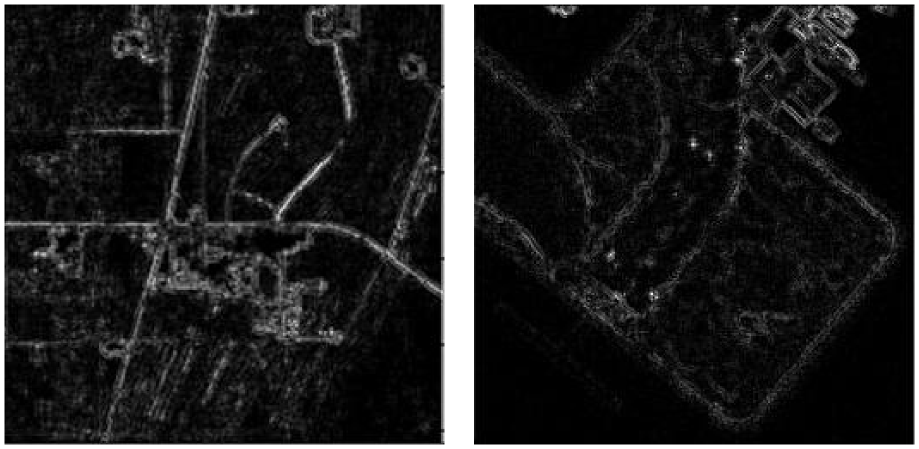


Figure 6 Images in Fig. 5 processed with *j* =1.

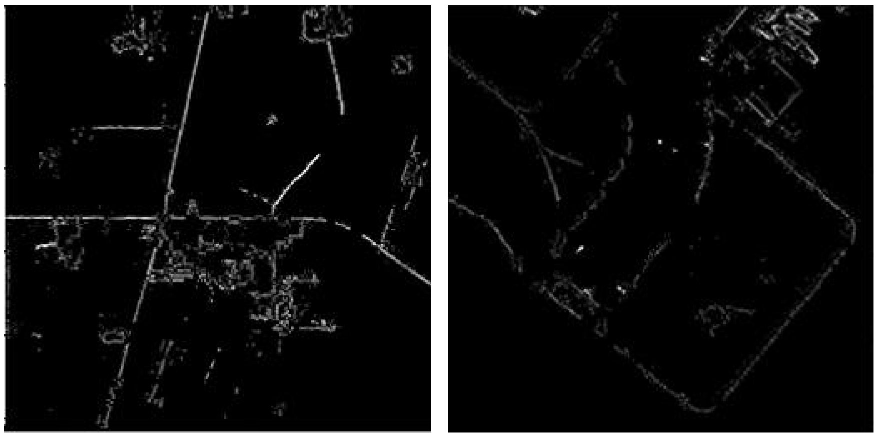


Figure 7 Images in Fig. 5 processed with j = 1, 2, 3, 4.

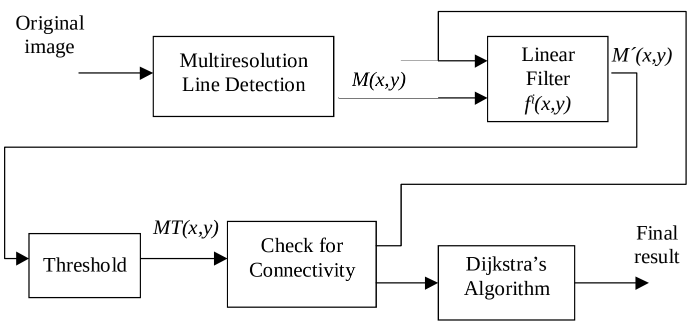


Figure 8 Block diagram of linking algorithm.

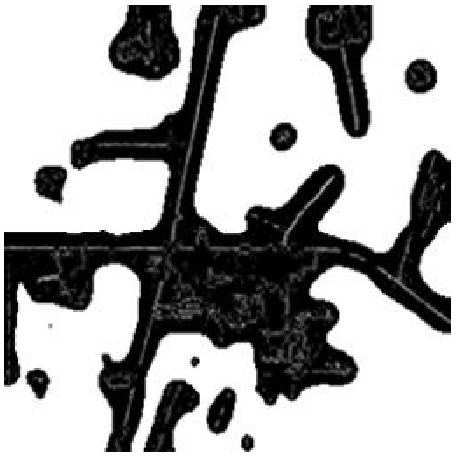


Figure 9 Intermediate linking algorithm; result of thresholding.

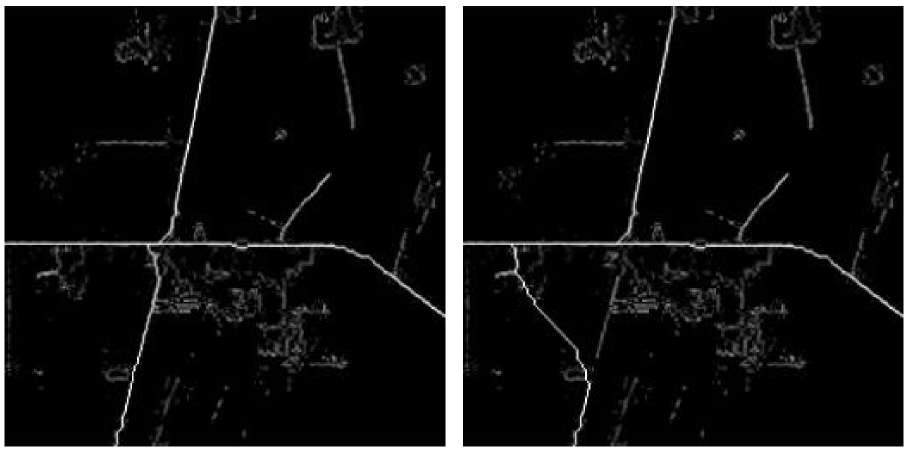


Figure 10 Result of road detection for two main roads using (a) our linking algorithm (b) conventional Dijkstra's algorithm.