segments out hands from the rest of the body by using skin color. However, since detecting humans in a cluttered video sequence is itself a very difficult problem, and the human body could easily be partially occluded in the scene, we try to bypass the human detection problem in our work by finding hands directly, without any attempt to find the entire human body first.

Most of the modern hand tracking systems fall into one of two main categories. The first approach uses skin color information to segment hands from the background and then tracks segmented hands between frames using a tracking algorithm. The face and hand tracking system for sign language recognition proposed by Soontranon et al. (2004) first segments the image into skin and non-skin regions using an elliptical model for skin pixels in CbCr space. Then face detection is used to locate the face skin blob ideally leaving only the skin blobs of hands. The system constructs a template for each hand then in subsequent frames, finds the region best matching that template using a minimum mean-squared error cost function. A similar approach is used by Wachs et al. (2005) to detect and track hands for human-robot interaction. The system proposed by Varona et al. (2004) also takes this approach but their system is extended to track hands and faces in 3D for a virtual reality application. The hand tracker of Shamaie and Sutherland (2005) does not use skin color information so it works on monochrome video sequences. Hands are extracted from the background using a blob analysis algorithm then tracked using a dynamic model from control theory. Unfortunately, these approaches relying on tracking are not suitable when the goal is to extract hands from single images.

However, in a second approach, a detection window is scanned over the image and each of the scanned image patches are classified as hand or non-hand. In contrast to

the first approach, this approach can be used to detect hands in static images. Barreto et al. (2004) applied improved version of Viola and Jones (2001) face detector by Lienhart and Maydt (2002), to detect hands for a human-robot interaction application. Their hand detection system works quite well at various scales and with different backgrounds under various illumination conditions. The hand tracking system proposed by Ong and Bowden (2004) uses a similar approach, but they construct a tree-structured classifier, instead of a linear cascade, not only to detect hands but also to classify hand posture. Both of these systems require high-resolution imagery. The hand detector by Caglar and Lobo (2006) also detects hands in high resolution static images, in this case making use of the geometric properties of the hand without the use of skin color or motion information. Their proposed system is robust to the size and the orientation of hands with the limitation that one or more fingers must be visible.

The goal of our proposed system is to detect and track multiple hands in arbitrary postures in relatively low-resolution video sequences. Our approach uses grayscale appearance information to reject most of the non-hand image regions very rapidly and then uses the shape of skin color regions to reject most of the remaining non-hand image patches. We conducted a thorough evaluation on our proposed system and found that its detection rate was 86.8% and that its false positive rate was 1.19 false detections per image on average. The system's speed and accuracy will enable many useful applications that are based on hand detection and tracking.

Materials and Methods

Hand Detector

A block diagram of our hand detection system is shown in Figure 1. A scan window is scanned over the input image at different scales and each of the resulting image patches is fed into a classifier cascade which rapidly determines whether the image patch is a hand. Our classifier cascade eliminates more than 95% of the non-hand regions in a given image. However, due to the large number of candidate regions in one image, to be practical, the false positive rate must be further reduced. To serve this need, we add a postprocessor to the system in order to further reduce false positive detections. The postprocessor takes advantage of a priori knowledge of hand's color and geometry. Skin detection, feature extraction, and Mahalanobis classification are the essential building blocks of our postprocessor.

Scanning Window

When an image is presented to our hand detection system (Figure 1), a detection window is scanned over the image at multiple scales, and each resulting image patch is passed to the boosted classifier cascade. The scanning process is to begin with a 24×24 detection at the image's original scale. After every possible image patch at that scale has run through the classifier cascade, the image is scaled down by 90% and the process is repeated until a minimum image size (maximum detection window size) is reached.

Boosted Classifier Cascade

Viola and Jones (2001, 2004) originally proposed the cascade of boosted classifiers as a real-time general object detector and applied it to face detection. They showed that the system could robustly detect faces in static images independent of the background. The system runs in real-time since the feature detector is limited to a class of Haar-like filters that can be computed in constant time with the help of integral images, regardless of the spatial extent of the filters. The speed of the system is increased even further by arranging the classifiers in a cascaded fashion, so that the early stages reject most of the image patches unlikely to contain the object of interest. The cascade therefore only spends significant compute time on the image patches most likely to contain the object of interest.

Each stage in the cascade is constructed from a set of simple Haar-like filters using Freund and Shapire's (1997) AdaBoost algorithm. AdaBoost builds a strong nonlinear classifier from multiple weak threshold classifiers, in this case each using a Haar-like filter, a threshold, and a weight, all of which are selected by AdaBoost to minimize the weighted error for the whole stage over the training set, while maintaining the desired detection rate. Viola and Jones (2001, 2004) used the four types of Haar-like filters shown in Figure 2 (a). The filters can take on arbitrary positions and sizes within an 24×24 image patch. The output of each filter is simply the difference between the average pixel value within the clear rectangular regions and the shaded rectangular regions.

Recently, Lienhart and Maydt (2002) modified Viola and Jones (2001, 2004) detector. Their system adds additional rotated Haar-like filter types, as shown in Figure 2 (b). On a particular test set, they found that their modified system gave 10% fewer

false positives than the original system for certain detection rates. The empirical analysis of detection cascade of boosted classifier by Lienhart et al. (2002) compared Discrete AdaBoost (the algorithm used by Viola and Jones [2001]), Real AdaBoost, and Gentle AdaBoost and found that classifiers trained with Gentle AdaBoost performed the best.

We apply Lienhart and colleagues' methods, as implemented in the OpenCV, Open Computer Vision Library (2006), to the hand detection problem, using all the filter types in Figure 2 (b), 24×24 image patches, and the Gentle AdaBoost learning algorithm.

Since boosting algorithms are supervised learning algorithms, a large number of labeled positive and negative examples must be input to the training process. Besides the examples, some learning parameters must be specified. The most important parameter is the desired true positive and false positive rate for each stage of the cascade. We train one stage at a time until that stage achieves the specified true positive and false positive rates. Then a new stage is begun, and the process continues until some stopping criterion is reached. Only the positive examples that are correctly classified by the previous stages and the negative examples that are incorrectly classified by the previous stages are used to train each new stage.

Skin Detector

There are many approaches to segmenting regions with similar color and texture from other regions. To extract skin color blobs from images, color information is the obvious choice. The skin detector for our system need not be extremely robust but it should be fast.

The Bayesian maximum likelihood classifier based on color histograms, as presented by Zarit et al. (1999), meets all of these needs. Based on their results and our own follow-up study, we selected the HS (hue and saturation) color model. Histograms used in the skin detector have two dimensions, namely hue and saturation. Each axis of the plane is quantized into 16 bins, so that each histogram will have $16^2 = 256$ bins. We selected 16-bin quantization based on comparison experiments with different bins counts of 8, 16, 32, and 64. We found that 16 bins along each axis gave the best performance. The reasons for excluding the intensity component from the histogram are to eliminate the effect of non-uniform illumination and to save computational cost. We construct histograms for skin and non-skin pixels from a large training set. The histogram counts are used to construct a discrete class-conditional likelihood for a Bayesian maximum likelihood classifier which we then use to determine whether a given pixel is most likely skin or not skin.

Each image patch which is classified as a hand by the cascade is scaled to a standard size 24×24 pixels and then fed to the skin detector, which produces a binary image, in which the value 1 represents a putative skin pixel and the value 0 represents a non-skin pixel.

Features Extractor and Mahalanobis Classifier

The shape and relative size of the skin blob within the detection window give useful information for discriminating image patches containing hand from those not containing hand. We extract four simple features from the binary skin image that are surprisingly useful for accurate classification:

1. The area of the largest connected component of skin pixels.

2. The length of the perimeter of the largest connected component of skin pixels.
3. The eccentricity of the largest connected component of skin pixels.
4. The number of pixels on the boundary of the largest connected component of skin pixels that intersect the detection window boundary.

The area feature is simply the number of pixels in the largest connected skin component; it is normalized by the total number of skin pixels ($24 \times 24 = 576$) in the image patch. It is very obvious that the given image patch is unlikely to contain a hand if the area feature is very large or very small. The perimeter feature is the total number of pixels on the perimeter of the largest connected skin component; it is normalized in the same way as the area feature. The eccentricity feature is the eccentricity of the ellipse having the same second moments as the largest connected skin component, i.e., the ratio of the distance between the foci of the ellipse and its major axis length. The eccentricity is between 0 and 1, with 0 indicating a circle and 1 indicating a line segment. This feature helps to discriminate face skin regions, which tend to be quite round, from true hand skin regions, which tend to be more eccentric. Finally, the boundary feature helps to discriminate between arm skin regions, which tend to intersect the boundary of the detection window in two places, from true hand skin regions, which only intersect the detection window at the wrist. The boundary feature provides information about how wrist-like the boundary is.

No matter how good those four features are, they will not be efficiently utilized for classification without a suitable classifier. We prefer classifiers that are simple with few parameters to tune. We find that a simple classifier based on Mahalanobis distance is a reasonable choice. Each image patch can be represented by a feature vector consisting

of the area, perimeter, eccentricity, and boundary features. To classify a given feature vector x as a true hand or not a hand, we calculate the Mahalanobis distance

$$d(x) = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

between the feature vector x and the mean feature vector μ, then we classify x as a hand if $d(x)$ is less than some threshold θ. Here the mean hand feature vector μ, the covariance matrix Σ, and the distance threshold θ are estimated from the training set.

Once classification for each possible detection windows is done, the positively detected hands are fed to the final module, the grouping, filtering, and averaging module. Further reduction of false positives is done there.

Grouping, Filtering, and Averaging Module

Our Mahalanobis classifier produces a few very sparsely distributed false positives and densely distributed true detections around the actual targets. Since it produces several true detections around each of the actual detections, grouping and averaging is necessary to ensure only one detection for each target. A group which contains less than some number of detections can be disposed of on the assumption it is a false positive. We use the existing implementation of this technique in the OpenCV. The positively detected hands output from this module could then be forwarded to another component in an integrated application, for example a gesture recognition module. But, in this article we simply evaluate the performance and efficiency of the proposed algorithm on a series of video sequences. We now describe our experiments in detail.

Data Acquisition

For the purpose of training, testing and evaluation of the proposed hand detection system, we captured 12 video sequences in a moderately cluttered laboratory environment. Four people volunteered to be models, and we captured three video sequences for each person. In the first sequence, each model walked away from the camera then came back to the starting position, in a direction parallel to the camera angle. In the second sequence, each model walked back and forth across the field of view in a direction perpendicular to the camera angle, at three different distances from the camera. In the last sequence, each model walked diagonally across the field of view, starting from a position to the right or left of the camera then returned to the start position, and repeated the procedure beginning from the other side of the camera.

We captured the video sequences at 15 frames per second with an inexpensive IEEE1394 Web camera at a resolution of 640×480 pixels. Each sequence lasted approximately 30 seconds. After video capture, all visible hands not smaller than the standard size of 24×24 pixels in every image of all 12 sequences were manually located. A total of 2246 hand locations were obtained. Our criteria for locating the selection window on the hand was that the hand should be roughly at the center of the window while taking up about 50% of the pixel area of the selection window. Some examples are shown in Figure 3.

Of the 12 video sequences, 11 were used to train the system and the remaining sequence was reserved for testing and evaluating the complete hand detection system. To train the boosted classifier cascade, we used 2000 hands as positive examples, and negative examples were automatically extracted from a set of background images. As background images, we used four randomly selected images from the video sequences

that did not contain any human. We created an additional set of background images using six randomly selected images containing humans. From each image, we cut out two large regions that did not containing hands but did contain other body parts such as faces and arms. From the test image sequence, we selected 99 images, each of containing at least one hand not smaller than 24×24 pixels. These 99 images contained a total of 106 proper hands. All of our test evaluation calculations are based on those 106 proper hands.

We also prepared a holdout set by randomly selecting 100 images from the 11 training video sequences. This holdout set was used to monitor system performance as well as to tune system parameters.

Boosted Classifier Training

To train the classifier cascade, we used Lienhart and colleagues' approach, implemented in OpenCV. We used the previously-described 2000 manually located hands from the eleven training video sequences as positive examples and the 16 previously-described background images.

The important parameters of the training process are the minimum hit rate (true positive rate) and maximum false alarm rate (false positive rate). Every stage in the cascade must satisfy these criteria on the training set. We used 100% for the hit rate and 60% for the false alarm rate. This means when adding a new stage to the classifier, the training system keeps adding additional weak classifiers to that stage until it correctly classifies all of the positive training examples with at most a 60% false alarm rate. Lienhart and colleagues' training system extracts the desired number of negative examples, 4000 for our experiment, by scanning a window with different scales over the

background images. After training one stage of the classifier, the negative examples which are correctly classified are disposed of and the system extracts a sufficient number of new negative examples. We use the Gentle AdaBoost variant of AdaBoost and the full Haar-like feature set of Lienhart and Maydt (2002).

The performance of the cascade is tested on the holdout set every time a new stage is constructed and added to the cascade. The results of the training process will be discussed in more detail in the Results and Discussion section.

Skin Detector Training

To train our skin detector, we selected 10 images containing one or more humans from a set of independent video sequences captured under various lighting conditions at several different locations. Skin pixels on those images were manually marked and the resulting 70,475 skin and 1,203,094 non-skin pixels were fed to the skin detector training process. The training process computes the hue (H) and saturation (S) for each pixel and quantizes each value into one of 16 bins. From the quantized values of skin pixels, one 2D histogram is constructed, and another is constructed from the quantized values of the non-skin pixels. Both histograms are constructed by simply counting the number of pixels which belong to same bin, and they are normalized by the total number of pixels used to construct the histogram.

Mahalanobis Classifier Training

The purpose of the Mahalanobis classifier is to eliminate the false detections made by the boosted classifier cascade while still maintaining a high detection rate. As the detection window is scanned over every image in the training set, the boosted

classifier outputs both true positive and false positive image patches. We found 78,658 true positives on our training set then randomly selected 6,000 true positives for computing the mean feature vector μ and covariance matrix Σ for the Mahalanobis classifier.

After we obtain μ and Σ for the Mahalanobis classifier, we need to find the optimum threshold. To do so, we scanned a detection window over every image in the holdout set and separated the detected image patches into false positives and true positives using the known hand locations for the holdout set. We extracted the Mahalanobis classifier's four features from each detected image patches and calculated the Mahalanobis distance between the feature vector of each image patch and the mean feature vector μ. As the class for each image patch is known, we plotted the ROC curve as shown in Figure 4. At this point, a detection rate of less than 100% is acceptable because the classifier cascade typically produces multiple true detections around each hand. Examining the ROC curve, we found that a Mahalanobis distance of 2.9 is a reasonable threshold since this threshold gives a very low false positive rate (6%) while giving an acceptable true positive rate (60%) on the image patches output by the classifier cascade.

Parameter Tuning for the System

Once all the required building blocks for hand detection are in place, we need to specify one last parameter, i.e., the minimum number of nearby positive image patches required for the Group, Filter, and Average block. In practice, this parameter must be tuned to achieve a good detection rate. To tune this parameter, we assembled all of the building blocks into a complete system then tested it on the holdout set with various

values for neither parameter. We found that a minimum of 4 neighboring patches produced the optimal result: 81.8% of the hands in the holdout set were detected and the false positive rate was also relatively low, an average of 1.55 false positives per image.

Testing the Complete System

We tested the complete hand detection system on the test set that was never used in any part of the training process. As previously described we used 99 images containing 106 hands in known locations. The detailed results of the test are discussed in the next section.

Results and Discussion

During the training process, we monitored the performance of the cascade and found that 12 stages of strong classifiers gave the optimum performance. The 12-stage classifier had a 97.5% detection rate on the holdout set, while having a reasonably low false positive rate of about 0.3% on the holdout set. A false positive rate of 0.3% may seem quite low but in fact this means we had an average of 1,000 false positive detections per image because one image contains more than 300,000 possible image patches. Clearly, these results indicate that a post processor is necessary to further eliminate false positives if the system is to be useable in practical applications.

When we tested our system on the test set, we found that the boosted classifier cascade frequently detected incorrect body parts such as arms, as shown in the left half of Figure 5 (b). However, the skin detection images shown in the right halves of Figures 5 (a) and 5 (b) show that the Mahalanobis classifier's boundary feature can distinguish between these cases. We found that most of the remaining false positives contained

either too few skin pixels or sparsely distributed skin pixels. These cases are easily eliminated by the Mahalanobis classifier's area feature since it operates on the largest connected skin component.

Our final hand detector detects 92 hands (86.8%) of the 106 hands in the test set, with an acceptable false positive rate of 1.19 false detections per image on average. A detection rate of 86.8% will enable many applications based on hand detection, such as human action recognition systems for security. Images (a) and (b) in Figure 6 illustrate example detections by our complete system, and all detected hands in the test set are shown in Figure 7. In the example, all hands were detected in both images, and only one false detection occurred in each image. The false detection of the desktop computer in the middle of the image is present in almost every image because the computer's color and texture are in fact similar to that of a hand. This kind of false positive detection on a stationary object will be eliminated if we add motion information between two consecutive frames in the video sequence.

Conclusion

From the literature, we know that hand detectors incorporating AdaBoost and Haar-like features perform quite well in applications like sign-language recognition, in which images are relatively high resolution with less cluttered background and constrained hand gesture. These approaches suffer from high false positive rates and low detection rates when applied to detect less constrained hands in low resolution and cluttered images. However we find that these limitations can be overcome with the help of a simple but efficient post processing system – in our experiments, the prototype hand detection system achieved excellent performance on its test set. One important

limitation of this work is that both the training and testing image sequences were captured in the same environment. This means that the performance of our current system is likely background dependent; if so, the reported performance is optimistic. However, the current results are encouraging, and in future work we plan to explore integrating our system with gesture recognition and human action recognition systems.

Acknowledgement

We thank the member of the Image and Vision Computing Laboratory at the Sirindhorn International Institute of Technology for the participation in our data collection efforts. This research was partially supported by Thailand Research Fund grant MRG4780209 to Matthew N. Dailey.

References

Barreto, J., Menezes, P. and Dias, J. 2004. Human-robot interaction based on haar-like features and eigenfaces. Proceedings of the 2004 IEEE Conference on Robotics and Automation, 2004,1888-1893.

Caglar, M.B. and Lobo, N. 2006. Open hand detection in a cluttered single image using finger primitives. Proceeding of the 2006 Computer Vision and Pattern Recognition Workshop, June 17-22, 2006.

Freund Y. and Shapire, R.E. 1997. A decision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Sciences 5(1):119-139.

Intel Corporation. Open Computer Vision Library (software). 2006. Open source software available at http://sourceforge.net/projects/opencv/.

Lienhart, R. and Maydt, J. 2002. An extended set of Haarlike features for rapid object detection. Proceedings of the IEEE International Conference on Image Processing, 2002, 900-903.

Lienhart, R., Kuranov, A. and Pisarevsky, V. 2002. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs.

Ong, E. and Bowden, R. 2004. A boosted classifier tree for hand shape detection. Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, 889-894.

Shamaie, A. and Sutherland, A. 2005. Hand tracking in bimanual movements. Image and Vision Computing 23(13):1131-1149.

Soontranon, N., Aramvith, S. and Chalidabhongse, T.H. 2004. Face and hands localization and tracking for sign language recognition. International Symposium on Communications and Information Technologies, 2004, 1246-1251.

Varona, J., Buades, J.M. and Perales, F.J. 2005. Hands and face tracking for VR applications, Computers & Graphics 29(2):179-187.

Viola P.A. and Jones, M.J. 2004. Robust real-time face detection. International Journal of Computer Vision 57(2):137-154.

Viola P.A. and Jones, M.J. 2001. Rapid object detection using a boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001, 511-518.

Wachs, J., Stern, H., Edan, Y., et al. 2005. A real-time hand gesture system based on evolutionary search. Genetic and Evolutionary Computation Conference, 2005.

Wren, C.R., Azarbayejani, A., Darrell, T. and Pentland, A. 1997. Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7): 780-785.

Zarit, B.D., Super, B.J. and Quek, F.K.H. 1999. Comparison of five color models in skin pixel classification. International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems, 1999, 58–63.

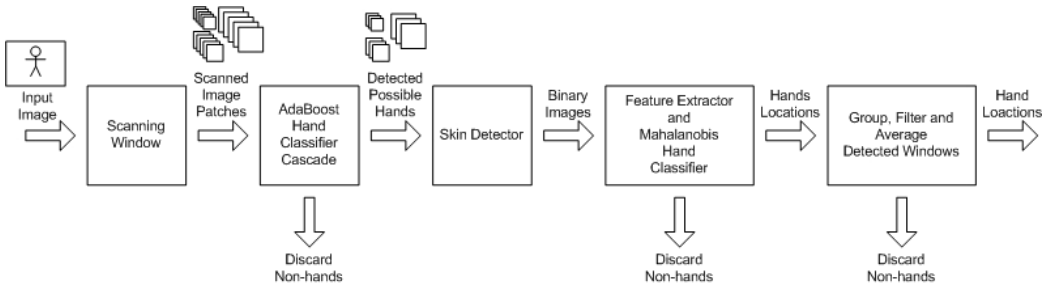
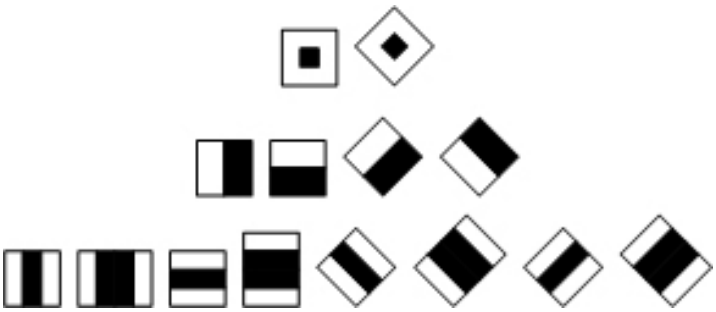


Figure 1, Hand detection system architecture.

(a)



(b)

Figure 2, Haar-like features used to construct weak classifies in the boosted classifier cascade. (a) Features used by Viola and Jones. (b) Features used by Lienhart and colleagues.

Figure 3, Example training images Scaled to 24x24.

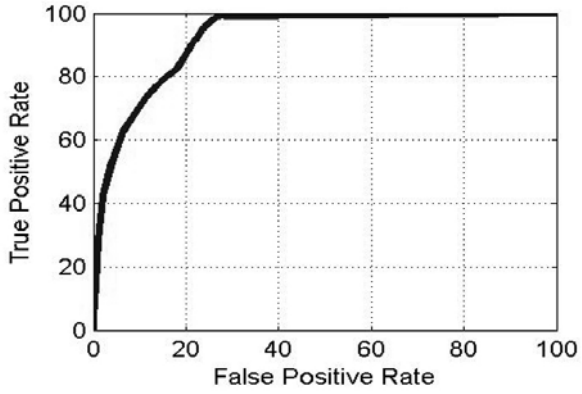
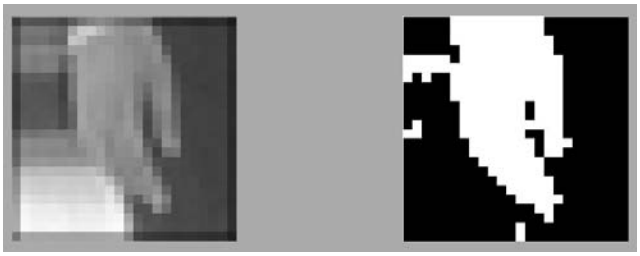
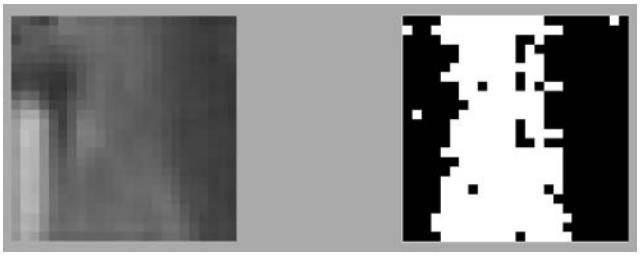


Figure 4, ROC cure between true positive and false positive for different threshold on Mahalanobis distance. True positive and false positive rate are calculated based on number of true detection and false detection input to the Mahalanobis Classifier.



(a) Properly detected hand.



(b) Non-hand body part detected as hand.

Figure 5, Original (left) image patches detected as hand by boosted hand classifier and binary images patches (right) after skin detection.

(a)

(b)

Figure 6, Example detection results of our proposed hand detection system.

Figure 7, Hands detected by our complete hand detector system. All detections are scaled down to standard size 24x24 pixels for easy visualization.

A Family of Quadratic Snakes for Road Extraction

Ramesh Marikhu[1] Matthew N. Dailey[2] Stanislav Makhanov[3] Kiyoshi Honda[4]

[1] Information and Communication Technologies, Asian Institute of Technology
[2] Computer Science and Information Management, Asian Institute of Technology
[3] Sirindhorn International Institute of Technology, Thammasat University
[4] Remote Sensing and GIS, Asian Institute of Technology

Abstract. The geographic information system industry would benefit from flexible automated systems capable of extracting linear structures from satellite imagery. Quadratic snakes allow global interactions between points along a contour, and are well suited to segmentation of linear structures such as roads. However, a single quadratic snake is unable to extract disconnected road networks and enclosed regions. We propose to use a family of cooperating snakes, which are able to split, merge, and disappear as necessary. We also propose a preprocessing method based on oriented filtering, thresholding, Canny edge detection, and Gradient Vector Flow (GVF) energy. We evaluate the performance of the method in terms of precision and recall in comparison to ground truth data. The family of cooperating snakes consistently outperforms a single snake in a variety of road extraction tasks, and our method for obtaining the GVF is more suitable for road extraction tasks than standard methods.

1 Introduction

The geographic information system industry would benefit from flexible automated systems capable of extracting linear structures and regions of interest from satellite imagery. In particular, *automated road extraction* would boost the productivity of technicians enormously. This is because road networks are among the most important landmarks for mapping, and manual marking and extraction of road networks is an extremely slow and laborious process. Despite years of research and significant progress in the computer vision and image processing communities (see, for example, [1, 2] and Fortier et al.'s survey [3]), the methods available thus far have still not attained the speed and accuracy necessary for practical application in GIS tools.

Among the most promising techniques for extraction of complex objects like roads are *active contours* or *snakes*, originally introduced by Kass et al. [4]. Since the seminal work of Kass and colleagues, techniques based on active contours have been applied to many object extraction tasks [5] including road extraction [6].

Rochery et al. have recently proposed *higher-order active contours*, in particular *quadratic snakes*, which hold a great deal of promise for extraction of linear

structures like roads [7]. The idea is to use a quadratic formulation of the contour's geometric energy to encourage anti-parallel tangents on opposite sides of a road and parallel tangents along the same side of a road. These priors increase the final contour's robustness to partial occlusions and decrease the likelihood of false detections in regions not shaped like roads.

In this paper, we propose two heuristic modifications to Rochery et al.'s quadratic snakes, to address limitations of a single quadratic snake and to accelerate convergence to a solution. First, we introduce the use of a *family* of quadratic snakes that are able to split, merge, and disappear as necessary. Second, we introduce an improved formulation of the image energy combining Rochery et al.'s oriented filtering technique [7] with thresholding, Canny edge detection, and Xu and Prince's Gradient Vector Flow (GVF) [8]. The modified GVF field created using the proposed method is very effective at encouraging the quadratic snake to snap to the boundaries of linear structures. We demonstrate the effectiveness of the family of snakes and the modified GVF field in a series of experiments with real satellite images, and we provide precision and recall measurements in comparison with ground truth data. The results are an encouraging step towards the ultimate goal of robust, fully automated road extraction from satellite imagery.

As a last contribution, we have developed a complete GUI environment for satellite image manipulation and quadratic snake evolution, based on the Matlab platform. The system is freely available as open source software [9].

2 Experimental Methods

2.1 Quadratic snake model

Here we provide a brief overview of the quadratic snake proposed by Rochery et al. [7]. An *active contour* or *snake* is parametrically defined as

$$\boldsymbol{\gamma}(p) = \left[x(p)\ y(p)\right]^T, \tag{1}$$

where p is the curvilinear abscissa of the contour and the vector $\left[x(p)\ y(p)\right]^T$ defines the Cartesian coordinates of the point $\boldsymbol{\gamma}(p)$. We assume the image domain Ω to be a bounded subset of $\mathbb{R}^2$.

The energy functional for Rochery et al.'s *quadratic snake* is given by

$$E_s(\boldsymbol{\gamma}) = E_g(\boldsymbol{\gamma}) + \lambda E_i(\boldsymbol{\gamma}), \tag{2}$$

where $E_g(\boldsymbol{\gamma})$ is the *geometric energy* and $E_i(\boldsymbol{\gamma})$ is the *image energy* of the contour $\boldsymbol{\gamma}$. λ is a free parameter determining the relative importance of the two terms.

The geometric energy functional is defined as

$$E_g(\boldsymbol{\gamma}) = L(\boldsymbol{\gamma}) + \alpha A(\boldsymbol{\gamma}) - \frac{\beta}{2} \iint \boldsymbol{t}_{\boldsymbol{\gamma}}(p) \cdot \boldsymbol{t}_{\boldsymbol{\gamma}}(p')\ \Psi\left(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|\right)\ dp\ dp', \tag{3}$$

where $L(\boldsymbol{\gamma})$ is the length of $\boldsymbol{\gamma}$ in the Euclidean metric over Ω, $A(\boldsymbol{\gamma})$ is the area enclosed by $\boldsymbol{\gamma}$, $\boldsymbol{t}_{\boldsymbol{\gamma}}(p)$ is the unit-length tangent to $\boldsymbol{\gamma}$ at point p, and $\Psi(z)$, given the

distance z between two points on the contour, is used to weight the interaction between those two points (see below). α and β are constants weighting the relative importance of each term. Clearly, for positive β, $E_g(\boldsymbol{\gamma})$ is minimized by contours with short length and parallel tangents. If α is positive, contours with small enclosed area are favored; if it is negative, contours with large enclosed area are favored.

The interation function $\Psi(z)$ is a smooth function expressing the radius of the region in which parallel tangents should be encouraged and anti-parallel tangents should be discouraged. $\Psi(z)$ incorporates two constants: d, the expected road width, and ϵ, the expected variability in road width. During snake evolution, weighting by $\Psi(z)$ in Equation 3 discourages two points with anti-parallel tangents (the opposite sides of a putative road) from coming closer than distance d from each other.

The image energy functional $E_i(\boldsymbol{\gamma})$ is defined as

$$\begin{aligned} E_i(\boldsymbol{\gamma}) = & \int \boldsymbol{n}_{\boldsymbol{\gamma}}(p) \cdot \nabla I(\boldsymbol{\gamma}(p))\ dp \\ & - \iint \boldsymbol{t}_{\boldsymbol{\gamma}}(p) \cdot \boldsymbol{t}_{\boldsymbol{\gamma}}(p')\ \nabla I(\boldsymbol{\gamma}(p)) \cdot \nabla I(\boldsymbol{\gamma}(p'))\ \Psi(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|)\ dp\ dp', \end{aligned} \tag{4}$$

where $I : \Omega \rightarrow [0, 255]$ is the image and $\nabla I(\gamma(p))$ denotes the 2D gradient of I evaluated at $\gamma(p)$. The first linear term favors anti-parallel normal and gradient vectors, encouraging counterclockwise snakes to shrink around or clockwise snakes to expand to enclose dark regions surrounded by light roads.[5] The quadratic term favors nearby point pairs with two different configurations, one with parallel tangents and parallel gradients and the other with anti-parallel tangents and anti-parallel gradients.

After solving the Euler-Lagrange equations for minimizing the energy functional $E_s(\gamma)$ (Equation 2), Rochery et al. obtain the update equation

$$\begin{aligned} \boldsymbol{n}_{\boldsymbol{\gamma}}(p) \cdot \frac{\partial E_s}{\partial \boldsymbol{\gamma}}(p) = & -\kappa_{\boldsymbol{\gamma}}(p) - \alpha - \lambda \|\nabla I(\boldsymbol{\gamma}(p))\|^2 + G(\boldsymbol{\gamma}(p)) \\ & + \beta \int \boldsymbol{r}\left(\boldsymbol{\gamma}(p), \boldsymbol{\gamma}(p')\right) \cdot \boldsymbol{n}_{\boldsymbol{\gamma}}(p')\ \ \Psi'\left(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|\right) dp' \\ + 2\lambda \int \boldsymbol{r}\left(\boldsymbol{\gamma}(p), \boldsymbol{\gamma}(p')\right) \cdot \boldsymbol{n}_{\boldsymbol{\gamma}}(p')\ \ & \left(\nabla I(\boldsymbol{\gamma}(p)) \cdot \nabla I(\boldsymbol{\gamma}(p'))\right)\ \ \Psi'\left(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|\right) dp' \\ & + 2\lambda \int \nabla I(\boldsymbol{\gamma}(p')) \cdot \left(\nabla \nabla I(\boldsymbol{\gamma}(p)) \times \boldsymbol{n}_{\boldsymbol{\gamma}}(p')\right)\ \Psi(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|)\ dp', \end{aligned} \tag{5}$$

where $\kappa_{\boldsymbol{\gamma}}(p)$ is the curvature of $\boldsymbol{\gamma}$ at $\boldsymbol{\gamma}(p)$ and $G(\boldsymbol{\gamma}(p))$ is the "specific energy," evaluated at point $\boldsymbol{\gamma}(p)$ (Section 2.2). $\boldsymbol{r}\left(\gamma(p), \gamma(p')\right) = \frac{\gamma(p) - \gamma(p')}{\|\gamma(p) - \gamma(p')\|}$ is the unit

[5] For dark roads in light background, we negate all the terms involving image, including $G(\boldsymbol{\gamma}(p))$ in Equation 5. In the rest of the paper, we assume light roads on a dark background.

vector pointing from $\gamma(p)$ towards $\gamma(p')$. $\nabla\nabla I(\gamma(p))$ is the Hessian of I evaluated at $\gamma(p)$.

α, β, and λ are free parameters that need to be determined experimentally. d and ϵ are specified a priori according to the desired road width. Following Rochery et al., we normally initialize our quadratic snakes with a rounded rectangle covering the entire image.

2.2 Oriented filtering

We use Rochery's oriented filtering method [10] to enhance linear edges in our satellite imagery. The input image is first convolved with oriented derivative-of-Gaussian filters at various orientations. Then the minimum (most negative) filter response over the orientations is run through a ramp function equal to 1 for low filter values and -1 for high filter values. The thresholds are user-specified. An example is shown in Fig. 1(b).

2.3 GVF energy

Rather than using the oriented filtering specific image energy $G(\boldsymbol{x})$ from Section 2.2 for snake evolution directly, we propose to combine the oriented filtering approach with Xu and Prince's Gradient Vector Flow (GVF) method [8]. The GVF is a vector field $V^{\mathrm{GVF}}(\boldsymbol{x}) = \left[u(\boldsymbol{x})\ v(\boldsymbol{x})\right]^T$ minimizing the energy functional

$$E(V^{\mathrm{GVF}}) = \int_{\Omega} \mu(u_x^2(\boldsymbol{x}) + u_y^2(\boldsymbol{x}) + v_x^2(\boldsymbol{x}) + v_y^2(\boldsymbol{x})) + \|\nabla\tilde{I}(\boldsymbol{x})\|^2\ \|V(\boldsymbol{x}) - \nabla\tilde{I}(\boldsymbol{x})\|^2\ d\boldsymbol{x}, \tag{6}$$

where $u_x = \frac{\partial u}{\partial x}$, $u_y = \frac{\partial u}{\partial y}$, $v_x = \frac{\partial v}{\partial x}$, $v_y = \frac{\partial v}{\partial y}$, and $\tilde{I}$ is a preprocessed version of image I, typically an edge image of some kind. The first term inside the integral encourages a smooth vector field whereas the second term encourages fidelity to $\nabla\tilde{I}$. μ is a free parameter controlling the relative importance of the two terms.

Xu and Prince [8] experimented with several different methods for obtaining $\nabla\tilde{I}$. We propose to perform Canny edge detection on G (the result of oriented filtering and thresholding, introduced in Section 2.2) to obtain a binary image $\tilde{I}$ for GVF, then to use the resulting GVF V^{GVF} as an additional image energy for quadratic snake evolution. The binary Canny image is ideal because it only includes information about road-like edges that have survived sharpening by oriented filters. The GVF field is ideal because during quadratic snake evolution, it points toward road-like edges, pushing the snake in the right direction from a long distance away. This speeds evolution and makes it easier to find suitable parameters to obtain fast convergence. Fig. 1 compares our method to alternative GVF formulations based on oriented filtering or Canny edge detection alone.

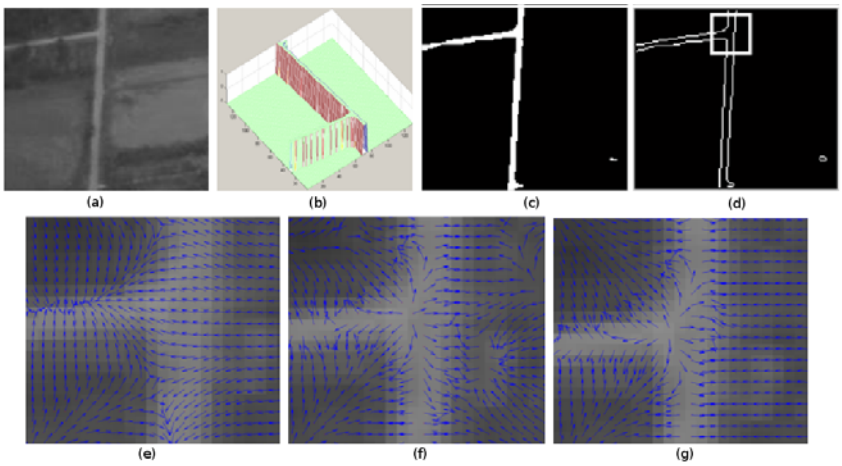


Fig. 1. Comparison of GVF methods. (a) Input image. (b) $G(\boldsymbol{x})$ obtained from oriented filtering on $I(\boldsymbol{x})$. (c) Image obtained from $G(\boldsymbol{x})$ using threshold 0. (d) Canny edge detection on (c), used as $\tilde{I}$ for GVF. (e-f) Zoomed views of GVFs in region delineated in (d). (e) Result of using the magnitude of the gradient $\nabla(G_\sigma * I)$ to obtain $\tilde{I}$. (f) Result of using Canny edge detection alone to obtain $\tilde{I}$. (g) GVF energy obtained using our proposed edge image. This field pushes most consistently toward the true road boundaries.

2.4 Family of quadratic snakes

A single quadratic snake is unable to extract enclosed regions and multiple disconnected networks in an image. We address this limitation by introducing a *family* of cooperating snakes that are able to split, merge, and disappear as necessary.

In our formulation, due to the curvature term $\kappa_\gamma(p)$ and the area constant α in Equation 5, specifying the points on γ in a counterclockwise direction creates a *shrinking snake* and specifying the points on γ in a clockwise direction creates a *growing snake*.

An enclosed region (loop or a grid cell) can be extracted effectively by initializing two snakes, one shrinking snake covering the whole road network and another growing snake inside the enclosed region.

On the one hand, our method is heuristic and dependent on somewhat intelligent user initialization, but it is much simpler than level set methods for the same problem [7], and, assuming a constant number of splits and merges per iteration, it does not increase the asymptotic complexity of the quadradic snake's evolution.

Splitting a snake We split a snake into two snakes whenever two of its arms are squeezed too close together, i.e. when the distance between two snake points is less than d^{split} and those two points are at least k snake points from each other

in both directions of traversal around the contour. d^{split} should be less than 2η, where η is the maximum step size.

Merging two snakes Two snakes are merged when they have high curvature points within a distance d^{merge} of each other, the two snakes' order of traversal (clockwise or counterclockwise) is the same, and the tangents at the two high curvature points are nearly antiparallel. High curvature points are those with $\kappa_\gamma(p) > 0.6\kappa_\gamma^{\max}$ where $\kappa_\gamma^{\max}$ is the maximum curvature for any point on γ. High curvature points are taken to ensure merging only occurs if two snakes have the semi-circular tip of their arms facing each other. Filtering out the low curvature points necessitates computing angle between the tangents at two points only for the high curvature points.

When these conditions are fulfilled, the two snakes are merged by deleting the high curvature points and joining the snakes into a single snake while preserving the direction of traversal for the combined snake.

Deleting a snake A snake γ is deleted if it has low compactness ($\frac{4\pi A(\boldsymbol{\gamma})}{L(\boldsymbol{\gamma})^2}$) and a perimeter less than L^{delete}.

2.5 Experimental design

We analyze extraction results on different types of road networks using the single quadratic snake proposed by Rochery et al. [7] and the proposed family of cooperating snakes. The default convergence criterion is when the minimum $E_s(\boldsymbol{\gamma})$ has not improved for some number of iterations.

Experiments have been performed to analyze the extraction of tree-structured road networks and those with loops, grids and disconnected networks.

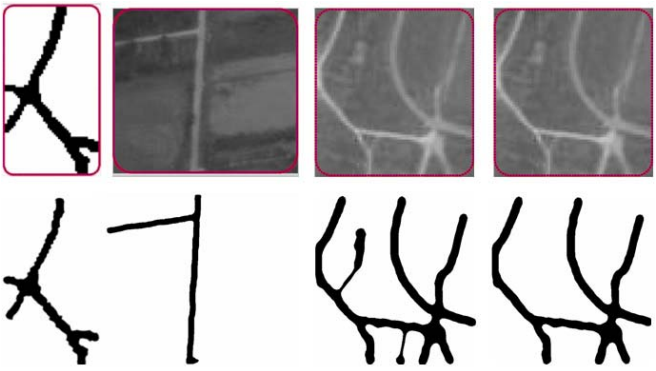
We then analyze the effectiveness of GVF energy obtained from the proposed edge image in Experiment 4. For all the experiments, we digitize the images manually to obtain the ground truth data necessary to compute precision and recall.

3 Results

We have obtained several parameters emperically. For splitting a snake, d^{split} should be less than `d`. k to be chosen depending on how far the two splitting points should be to ensure that the snakes formed after splitting have at least k points.

In order to ensure that merging of snakes takes place only among the arms with the semi-circular tips facing each other, the tangents at the high curvature points are checked for antiparallel threshold of $130\pi/180$..

The compactness should be greater than 0.2 to ensure that linear structured contours are not deleted.



Column	Snakes	Iter	Precision	Recall	α	β	λ	d
1	single	480	0.937	0.8605	0.75	0.35	4	3
2	single	308	0.9483	0.8503	0.15	0.2	3.5	4
3	single	473	0.7091	0.9074	0.1	0.2	4	3
4	family	450	0.8011	0.9115	0.1	0.2	4	3

Fig. 2. Evolution of quadratic snake on roads with tree structure.Each column displays an image with initial contour in red and the extracted road network below it.

3.1 Experiment 1: Simple (tree-structured) road networks

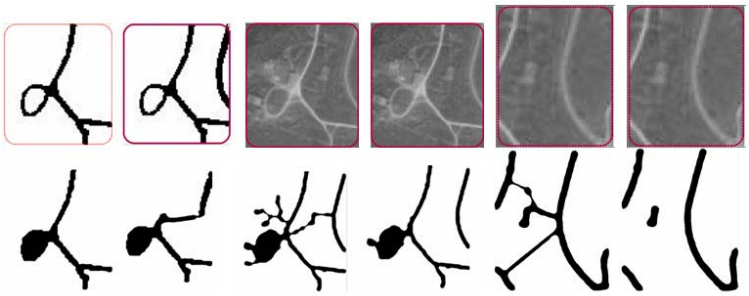
A single quadratic snake is well suited for tree-structured road networks as the snake will not need to change its topology during evolution (Figure 2). A family of snakes enable faster and better road extraction as non-road regions are eliminated using splitting and deletion of snakes.

3.2 Experiment 2: Road networks with single loop and multiple disconnected networks

The family of quadratic snakes are able to extract disconnected networks with high accuracy (Figure 3) but are not able to extract enclosed regions automatically as the snakes are not able to develop holes inside it in the form of growing snakes.

3.3 Experiment 3: Complex road networks

A road network is considered complex if it has multiple disconnected networks and enclosed regions and large number of branches. With the appropriate user initialization (Figure 4), the snakes are able to extract the road networks with high accuracy and in less time.



Column	Snakes	Iter	Precision	Recall	α	β	λ	d
1	single	1080	0.6364	0.9833	0.75	0.35	4	3
2	single	1300	0.5638	0.7322	0.75	0.35	4	3
3	single	182	0.5473	0.9114	0.1	0.2	4	2
4	family	182	0.6344	0.9833	0.1	0.2	4	2
5	single	219	0.6823	0.8673	0.1	0.2	3	2.5
6	family	163	0.8582	0.8731	0.1	0.2	3	2.5

Fig. 3. Evolution of quadratic snake on roads with loops and disconnected networks. Each column displays an image with initial contour in red and the extracted road network below it.

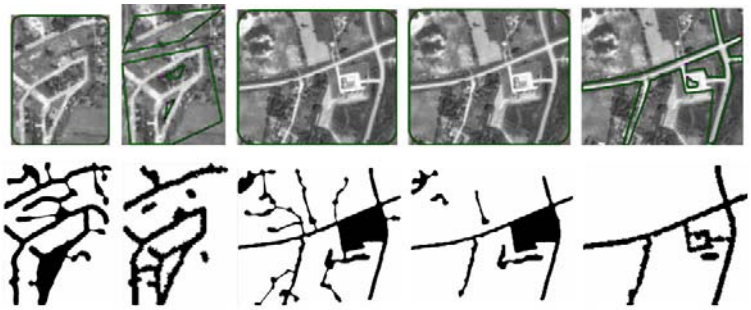
3.4 Experiment 4: GVF energy to enable faster evolution

The Gradient Vector Flow Field [8] boosts the evolution process as we can see from the number of iterations required for each evolution in Experiment 4 with and without the use of GVF energy. From the evolution in the fifth column, we see that the snake was able to extract the network with greater detail. Also, from the evolution in the last column, we see that it is necessary for the quadratic image energy to enable robust extraction and thus the GVF weight and λ need to be balanced appropriately.

4 Discussion and Conclusion

In Experiment 1, we found that the our modified quadratic snake is able to move into concavities to extract entire tree-structured road networks with very high accuracy. Experiment 2 showed that the family of quadratic snakes is effective at handling changes in topology during evolution, enabling better extraction of road networks. Currently, loops cannot be extracted automatically.

We demonstrated the difficulty in extracting complex road networks with multiple loops and grids in Experiment 3. However, user initialization of a family of contours enable extraction of multiple closed regions and help the snake to avoid road-like regions. The level set framework could be used to handle change in topology enabling effective extraction of enclosed regions. Rochery et al. [10] evolved the contour using the level set methods introduced by Osher and Sethian.



Column	Snakes	Iter	Precision	Recall	α	β	λ	d
1	single	519	0.5301	0.9067	0.1	0.25	3	3.5
2	user defined	166	0.7675	0.9117	0.1	0.25	3	3.5
3	single	450	0.3851	0.8171	0.05	0.05	2	4
4	family	350	0.5058	0.7987	0.05	0.05	2	4
5	user defined	104	0.799	0.8951	0.1	0.25	4	4

Fig. 4. Evolution of quadratic snake on roads with enclosed regions. Each column displays an image with initial contour in green and the extracted road network below it.

However, our method is faster, conceptually simpler, and a direct extension of Kass et al.'s computational approach.

In Experiment 4, we found that faster and robust extraction is achieved using oriented filtering and GVF energy along with image energy of the quadratic snakes. Our proposed edge image obtained from oriented filtering is effective for computing GVF energy to enhance the process of extraction. We also found that our method for obtaining the GVF outperforms standard methods.

Finally, we have developed a complete GUI environment for satellite image manipulation and quadratic snake evolution, based on the Matlab platform. The system is freely available as open source software [9].

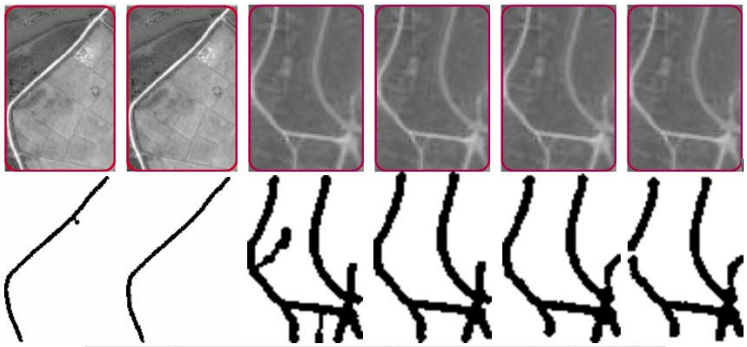
Future work will focus on possibilities to automate the extraction of enclosed regions. Digital elevation models could be integrated with image energy for increased accuracy.

Acknowledgments

This research was supported by Thailand Research Fund grant MRG4780209 to MND. RM was supported by a graduate fellowship from the Nepal High Level Commission for Information Technology.

References

1. Fischler, M., Tenenbaum, J., Wolf, H.: Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. Computer Graphics and Image Processing **15** (1981) 201–223



Column	Snakes	Iter	Precision	Recall	GVF Wt	α	β	λ	d
1	single	670	0.8789	0.9171	0	0.2	0.15	4	2.5
2	single	390	0.9066	0.9129	1	0.2	0.15	4	2.5
3	single	950	0.7321	0.9467	0	0.05	0.2	4	3
4	family	650	0.8279	0.9382	0	0.05	0.2	4	3
5	family	430	0.8017	0.9094	1	0.05	0.2	2	3
6	family	400	0.7925	0.8998	1	0.05	0.2	0	3

Fig. 5. Evolution of quadratic snake on roads with enclosed regions. Each column displays an image with initial contour in green and the extracted road network below it.

2. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. IEEE Transactions on Pattern Analysis and Machine Intelligence **18**(1) (1996) 1–14
3. Fortier, A., Ziou, D., Armenakis, C., Wang, S.: Survey of work on road extraction in aerial and satellite images. Technical Report 241, Université de Sherbrooke, Quebec, Canada (1999)
4. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1**(4) (1987) 321–331
5. Cohen, L.D., Cohen, I.: Finite-element methods for active contour models and balloons for 2-D and 3-D images. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(11) (1993) 131–147
6. Laptev, I., Mayer, H., Lindeberg, T., Eckstein, W., Steger, C., Baumgartner, A.: Automatic extraction of roads from aerial images based on scale space and snakes. Machine Vision and Applications **12**(1) (2000) 23–31
7. Rochery, M., Jermyn, I.H., Zerubia, J.: Higher order active contours. International Journal of Computer Vision **69**(1) (2006) 27–42
8. Xu, C., Prince, J.L.: Gradient Vector Flow: A new external force for snakes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (1997) 66–71
9. Marikhu, R.: A GUI environment for road extraction with quadratic snakes (2007) Matlab software, available at http://www.cs.ait.ac.th/∼mdailey/snakes.
10. Rochery, M.: Contours actifs d'order supérieur et leur application à la détection de linéiques dans des images de télédétection. PhD thesis, Université de Nice, Sophia Antipolis — UFR Sciences (2005)

Multiple Quadratic Snakes for Road Extraction ⋆

Matthew N. Dailey [a,*], Stanislav Makhanov [b], and Ramesh Marikhu [a]

[a] *Computer Science and Information Management*
Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120 Thailand

[b] *Sirindhorn International Institute of Technology*
Thammasat University
131 M. 5, Tiwanont Road, Bangkadi, A. Muang, Pathumthani 12000 Thailand

Abstract

We propose a family of quadratic cooperating snakes, which are able to split, merge, and disappear as necessary, for segmentation of roads in satellite imagery. We combine the multiple snake framework with a preprocessing method based on oriented filtering, Canny edge detection, and Gradient Vector Flow (GVF). We evaluate the performance of the method in terms of precision and recall in comparison to ground truth data. The family of cooperating snakes consistently outperforms a single snake in a variety of road extraction tasks.

Key words: Active contours, segmentation, road extraction

1 Introduction

The geographic information system industry would benefit from flexible automated systems capable of extracting linear structures and regions of interest from remote sensing imagery. In particular, *automated road extraction* would boost the productivity of technicians enormously. This is because road networks are among the most important landmarks for mapping, and manual marking and extraction of road networks is an extremely slow and laborious process.

1.1 Related work

Towards the ultimate goal of fully automated road extraction, there has been a great deal of progress in the computer vision and image processing communities on partially automating the process. For example, Geman and Jedynak [1] proposed statistical modeling of the responses of simple nonlinear oriented ridge filters to track a given road from a seed point and direction. The method is extremely accurate, even on extremely difficult imagery.

In fully automatic road extraction, we are required to detect all of the roads of a particular range of widths in a given region of an input image. The typical approach to solving this problem combines a local neighborhood analysis step that generates feature points or calculates local likelihoods, followed by im-

⋆ This is a revised and extended version of a paper presented in ACCV 2007.

* Corresponding author. Tel: +66 2 524 5712 Fax: +66 2 524 5721.
Email addresses: mdailey@ait.ac.th (Matthew N. Dailey),
smakhanov@siit.tu.ac.th (Stanislav Makhanov), marikhu@yahoo.com (Ramesh Marikhu).

position of global constraints to link possible road points and eliminate false positives by minimizing a global cost function. Typical global cost minimization techniques include dynamic programming [2,3], active contours or snakes [4], and Markov random fields [5,6].

After 30 years of research on road extraction in the computer vision and image processing communities (see [7] for a review), there is still no system attaining the speed, robustness, and level of automation necessary for practical application on arbitrary imagery. There are very good methods for tracking single roads (e.g. [1]), but it is very difficult to reliably extract complete road networks in the presence of variability in shape, radiometry, connectivity, and geometry.

Among the most promising techniques for extraction of complex objects like roads are *active contours* or *snakes*, originally introduced by Kass et al. [8]. Since the seminal work of Kass and colleagues, techniques based on active contours have been applied to many object extraction tasks [9].

Despite their popularity, the classical parametric snake model and its variations have several major drawbacks. Chief among them is the lack of topological flexibility. When there are several objects in the image to capture, the model requires multiple snakes which have to be manually initialized to be close to the contour of each object. The initialization can be done, at best, semi-automatically, which is often time-consuming and prone to misplacement. The number of snakes is usually fixed; they cannot merge, split, or disappear. Besides this topological inflexibility, individual snakes can intersect themselves and separate snakes can collide with each other. This is due to the inability of traditional snakes to repel parts of other snakes and repair self intersections

and loops. Remedying these problems requires geometric constraints to ensure that multiple snakes do not intersect, and these geometric constraints are difficult to implement. The problem is further exacerbated when nested snakes are initialized inside one another.

To deal with these topological issues, Wong et al. [10] introduced an adjustable "blow force" to detect convex and concave shapes and to avoid self intersections. Ivins and Porrill [11] introduced a "repulsion force" discouraging contours from intersecting themselves. However, neither of these techniques deal with multiple snakes or topology changes. Samadani [12] was perhaps the first to break an active contour into several pieces, using heuristic techniques based on "energy of deformation." Durkovich et al. [13] presented a heuristic rule to split a snake into several contours whenever two parts of a snake approach each other.

Ngoi and Jia [14] applied a positive/negative contour scheme to prevent self looping and to allow a splitting into multiple contours. A "positive" active contour is initialized as a point inside the object then expelled towards the object's boundary by negative charges enclosed by the contour. This outward deformation is constrained by positive charges outside the boundary. After each iteration, the contour points are subjected to a check for self-intersections. A self-intersection is detected if the minimum distance between two non-adjacent control points is less than three pixels. When a self-intersection is detected, a positive contour can either split into two positive contours (in the case of detection of another adjacent object) or a positive and a negative contour (in the case of detection of an internal region of the object).

In order to determine which part of a contour should be split to form two

contours, Choi et al. [15] classify the segments of a snake into "contour" and "non-contour" segments, calculating the surrounding image forces along a segment. If the surrounding image forces of a point are smaller than a threshold, it is deemed a non-contour point; otherwise, it is deemed a contour point. A sequence of contour points or non-contour points forms a contour or non-contour segment. Critical points are defined as the end points of a contour segment adjacent to non-contour segments. The method decides when to split or merge contours by evaluating the distance between the critical points.

Delingnette and Montagnat [16] proposed a new topology operator for automatically creating or merging active contours.

Evaluating these existing approaches for merging and splitting snakes, Ji and Yan [17] write that

> these approaches are high in computational cost since they require checking the potential self looping/connectivity change at every iteration ... Also, these approaches suggest that the moving speed of all snake points should be equivalent, therefore being unable to deal with more complex objects (e.g. long tube-like shapes) due to the nature of their test criteria (e.g. one based on the minimum distance between two non adjacent control points (p. 149)).

The authors overcome these limitations with a very complicated but apparently robust merging algorithm which employs polygon analysis as well as geometrical analysis of intersection points of colliding snakes. The algorithm involves analysis of many cases and requires verification of many geometric conditions associated with relative positions of points.

It should be noted that the heuristic techniques previously described attempt to *prevent* self-looping. However, it is possible to make use of self-loops by splitting useful loops into separate contours and eliminating useless loops. The "T-snakes" technique proposed by McInerney and Terzopoulos [18] and later improvements like the "dual T-snakes" technique [19] are based on iterative re-parameterization of the original contour. They are able to make the use of self loops, but the approach allows only "rigid" deformations limited by a superimposed "simplicial grid."

Rochery et al. have recently proposed a parametric model for *higher-order active contours*, in particular *quadratic snakes*, for extraction of linear structures like roads [20]. The idea is to use a quadratic formulation of the contour's geometric energy to encourage anti-parallel tangents on opposite sides of a road and parallel tangents along the same side of a road. These priors increase the final contour's robustness to partial occlusions, decrease the likelihood of false detections in regions not shaped like roads, and help to prevent self-looping, since different segments of a contour with anti-parallel tangents repel each other in the absence of image forces.

Finally, to address the topological flexibility problem with traditional active contours, Caselles et al. [21] and Malladi et al. [22] independently introduced "geometric active contour models." These models are based on curve evolution theory and level set methods [23]. Rochery et al. [20] have also proposed a level set method for their quadratic snakes. Level set methods for snakes introduce a higher dimensional hypersurface in which the snake is embedded as the zero level set of the hyper surface. The geometric approach has two advantages over traditional parametric representations. First, the curve can automatically break or merge as the hypersurface evolves. Second, since the hypersurface is

represented as a mathematical function, it admits straightforward and efficient numerical adaptation schemes.

However, geometric snake models have several inherent drawbacks compared to parametric models. First, the level set representation makes it difficult, if not impossible, to impose arbitrary geometric or topological constraints on the evolving contour via the higher dimensional hypersurface [18]. Second, they do not readily admit specification of a user-defined external force. Finally, the geometric active contour models may generate shapes having inconsistent topology with respect to the actual object, when applied to noisy images characterized by large boundary gaps [24]. Rochery et al.'s system [20] requires extensive optimization to achieve reasonable run times.

Li et al. [25], in reference to the problem of topological adaptation, write "in light of the ... inherent weaknesses of geometric active contour models, it is worthwhile to seek solutions within the parametric model realm."

1.2 Our approach

The *quadratic multiple snake model* developed in this paper presents a compromise between geometric snakes' ability to split and merge easily and parametric snakes' flexibility to incorporate arbitrary constraints. We use quadratic constraints [20] both to avoid self-intersections and loops and as a means to encourage capture of thin elongated objects such as roads, rivers, canal systems, pipes, and vascular systems. We develop efficient split and merge algorithms employing straightforward conditions on the closeness of non-adjacent contour points. In the model, separate snakes can repel each other but are still capable

of approaching an object from opposite sides. The split and merge algorithms make it possible to extract highly complex networks of roads and other linear structures. The model thus provides the topological adaptability of geometric models without sacrificing the simplicity, efficiency, or flexibility of traditional parametric models.

The ability of our cooperating snakes to split, merge and disappear combined with their self-repelling feature makes it possible to easily segment several objects without knowing the number of objects in advance and without initializing the snake close to the desired contour. As an introductory example, Figure 1 shows how our snakes can segment a "broken bar" consisting of two pieces. Figure 1(a) shows an initial configuration in which the snake is relatively far from the object of interest and defined by only a few points. Figure 1(b–c) shows two successful iterations of the snake evolution algorithm. Note that the "peninsula" (highlighted by the window in Figure 1(b)) splits from the snake then disappears in the next iteration. In fact, the snakes' ability to split and disappear also accelerates convergence to a minimum. Figure 1(d) shows a split at a point with high curvature, and Figure 1(e) shows the final configuration at convergence.

In addition to the multiple snake model, to accelerate convergence to a solution, we introduce an improved external force combining oriented filtering with Canny edge detection and Xu and Prince's Gradient Vector Flow (GVF) [26]. The modified GVF field created using the proposed method is very effective at encouraging the quadratic snake to snap to the boundaries of linear structures.

A second introductory example, illustrated in Figure 2, shows a horseshoe

Fig. 1. A quadratic snake splitting and converging around two island-like objects. (a) Initial configuration. (b–c) An arm (highlighted by the square) develops, splits off, and is deleted. (d) Configuration just before a split. (e) Final configuration.

shape used to verify the ability of the snake to converge into to deep concavities and to ignore noise. The image is distorted by a grid of curves that would distract the snake from the object were it not for the GVF force encouraging the snake to snap to the boundaries of road-like objects. The combination of oriented filtering, Canny edge detection, and the GVF external force makes it possible for the snake to ignore the obstructing grid entirely and attach itself to the object of interest despite the large gradients located far from the desired boundary.

These simple examples demonstrate that the variational formulation governed by the quadratic energy functional not only allows the snake to split and merge without creating loops within one snake or intersections between different

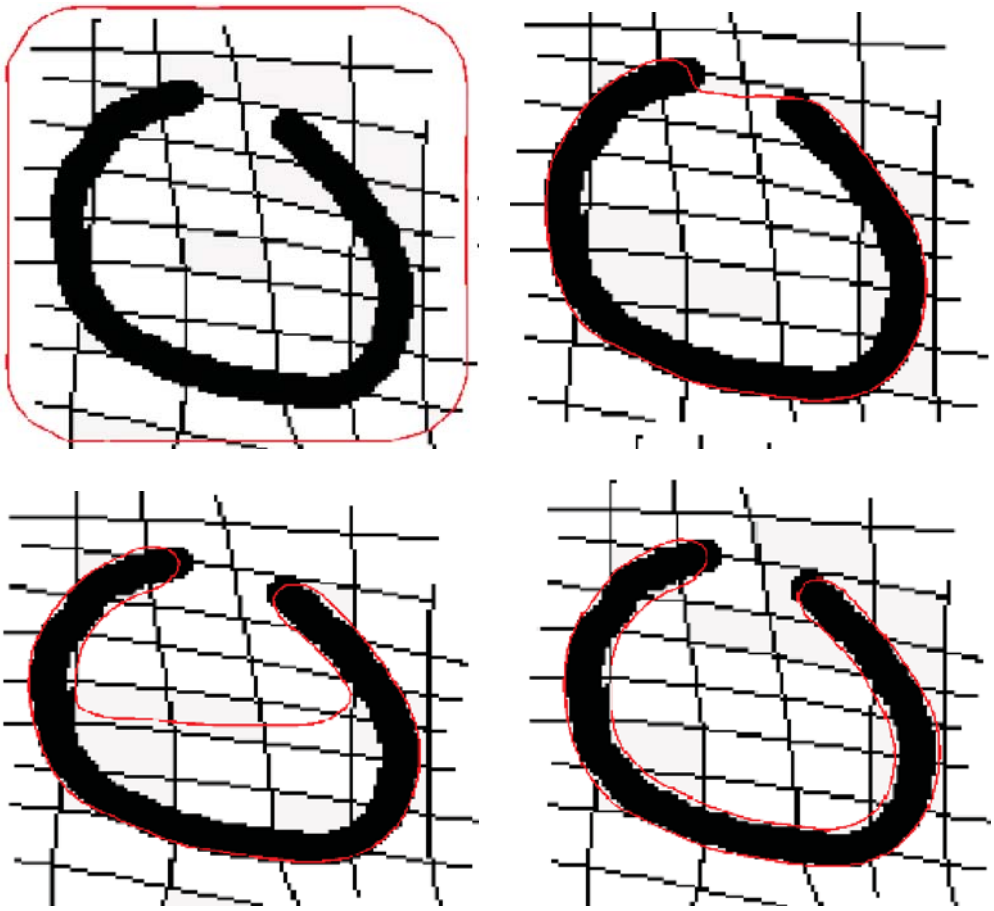


Fig. 2. A quadratic snake converging into a deep concavity despite the presence of noise.

snakes, but also makes it simple to incorporate the GVF, allowing the snake to capture the shape of complex objects with deep, narrow concavities.

In this paper, we demonstrate the effectiveness of the family of snakes and the modified GVF field in a series of experiments with real satellite images, and we provide precision and recall measurements in comparison with ground truth data. The results are an encouraging step towards the ultimate goal of fully automated road extraction from satellite imagery.

As a last contribution, we have developed a complete GUI environment for satellite image manipulation and quadratic snake evolution, based on the Mat-

lab platform. The system is freely available as open source software from `http://www.cs.ait.ac.th/~mdailey/snakes`.

2 Method

2.1 Quadratic snake model

This section provides a brief overview of the quadratic snake proposed by Rochery et al. [20]. An *active contour* or *snake* is parametrically defined as

$$\gamma(p) = \begin{bmatrix} x(p) & y(p) \end{bmatrix}^T, \tag{1}$$

where p is the curvilinear abscissa of the contour and the vector $\begin{bmatrix} x(p) & y(p) \end{bmatrix}^T$ defines the Cartesian coordinates of the point $\gamma(p)$.

The energy functional is given by

$$E_s(\gamma) = E_g(\gamma) + \lambda E_i(\gamma), \tag{2}$$

where $E_g(\gamma)$ is the *geometric energy* and $E_i(\gamma)$ is the *image energy* of the contour γ. λ is a free parameter determining the relative importance of the two terms.

To apply the method to road extraction, we define the geometric energy functional to be

$$E_g(\gamma) = L(\gamma) + \alpha A(\gamma) - \frac{\beta}{2} \iint \mathbf{t}(p) \cdot \mathbf{t}(p') \; \Psi\left(\|\gamma(p) - \gamma(p')\|\right) \; dp \; dp', \tag{3}$$

where $L(\gamma)$ is the Euclidean length of γ, $A(\gamma)$ is the area enclosed by γ, $\mathbf{t}(p)$ is the unit-length tangent to γ at point p, and $\Psi(z)$, given the distance z between

two points on the contour, is used to weight the interaction between those two points (see below). α and β are constants weighting the relative importance of the terms. Clearly, for positive β, $E_g(\gamma)$ is minimized by contours with short length and parallel tangents. If α is positive, contours with small enclosed area are favored; if it is negative, contours with large enclosed area are favored.

The interaction function $\Psi(\cdot)$ is a smooth function expressing the radius of the region in which parallel tangents should be encouraged and anti-parallel tangents should be discouraged:

$$\Psi(z) = \begin{cases} 1 & \text{if } z < d - \epsilon, \\ 0 & \text{if } z > d + \epsilon, \\ \frac{1}{2}\left(1 - \frac{z-d}{\epsilon} - \frac{1}{\pi}\sin\pi\frac{z-d}{\epsilon}\right) & \text{otherwise.} \end{cases} \tag{4}$$

In application to road extraction, d is the expected road width and ϵ expresses the expected variability in road width. During snake evolution, weighting by $\Psi(z)$ in Equation 3 discourages two points with anti-parallel tangents (the opposite sides of a putative road) from coming closer than distance d from each other.

The image energy functional $E_i(\gamma)$ is defined as

$$\begin{aligned} E_i(\gamma) = & \int \mathbf{n}(p) \cdot \nabla I(\gamma(p))\ dp \\ & - \iint \mathbf{t}(p) \cdot \mathbf{t}(p')\ \nabla I(\gamma(p)) \cdot \nabla I(\gamma(p'))\ \Psi(\|\gamma(p) - \gamma(p')\|)\ dp\ dp', \end{aligned} \tag{5}$$

where $I : \Omega \to [0, 255]$ is an image and $\nabla I(\gamma(p))$ is the gradient of I evaluated at $\gamma(p)$.

The first (linear) term favors anti-parallel normal and gradient vectors, encouraging counterclockwise snakes to shrink around or clockwise snakes to expand

to enclose dark regions surrounded by light roads.[1] The second (quadratic) term favors nearby point pairs with two different configurations, one with parallel tangents and parallel gradients and the other with anti-parallel tangents and anti-parallel gradients.

After solving the Euler equations for minimizing the energy functional $E_s(\gamma)$ (Equation 2), ignoring flow in the direction tangent to γ, we can obtain the update equation

$$\begin{aligned}\mathbf{n}(p) \cdot \frac{\delta E_s}{\delta \gamma}(p) &= -\kappa(p) - \alpha - \lambda \|\nabla I(\gamma(p))\|^2 \\ &\quad + \beta \int \mathbf{r}\left(\gamma(p), \gamma(p')\right) \cdot \mathbf{n}(p') \;\; \Psi'\left(\|\gamma(p) - \gamma(p')\|\right) dp' \\ &+ 2\lambda \int \mathbf{r}\left(\gamma(p), \gamma(p')\right) \cdot \mathbf{n}(p') \;\; \left(\nabla I(\gamma(p)) \cdot \nabla I(\gamma(p'))\right) \;\; \Psi'\left(\|\gamma(p) - \gamma(p')\|\right) dp' \\ &\quad + 2\lambda \int \nabla I(\gamma(p')) \cdot (\nabla \nabla I(\gamma(p)) \mathbf{n}(p')) \;\; \Psi(\|\gamma(p) - \gamma(p')\|) \; dp'. \quad (6)\end{aligned}$$

In the equation, $\kappa(p)$ is the curvature of γ at $\gamma(p)$.

$$\mathbf{r}\left(\gamma(p), \gamma(p')\right) = \frac{\gamma(p) - \gamma(p')}{\|\gamma(p) - \gamma(p')\|}$$

is the unit vector pointing from point $\gamma(p)$ towards $\gamma(p')$. $\nabla \nabla I(\gamma(p))$ is the 2×2 Hessian of I evaluated at $\gamma(p)$. α, β, and λ are free parameters that need to be determined experimentally. d and ϵ are specified a priori according to the desired road width.

2.2 GVF external force

The term $\alpha A(\gamma)$ in Equation 3 leads to the constant term $-\alpha$ in Equation 6. This "balloon force" [9] increases the capture region around objects, but

[1] For dark roads on a light background, we simply negate the terms involving the image. In the rest of the paper, we assume light roads on dark background.

its effect is uniform throughout the image. This makes it difficult to specify a value for α that is appropriate in all regions of the image.

Xu and Prince [26,27] have proposed to use, rather than a global balloon force, a smooth, diffuse gradient field as a local external force with the traditional linear snake. They find that this technique, Gradient Vector Flow (GVF), improves the traditional snake's convergence to a minimum energy configuration.

We propose the use of GVF with quadratic road extraction snakes.

2.2.1 GVF

The GVF is a vector field

$$V^{\mathrm{GVF}}(\mathbf{x}) = \begin{bmatrix} u(\mathbf{x}) \; v(\mathbf{x}) \end{bmatrix}^T$$

minimizing the energy functional

$$\begin{aligned} E(V^{\mathrm{GVF}}) = \int_\Omega & \mu(u_x^2(\mathbf{x}) + u_y^2(\mathbf{x}) + v_x^2(\mathbf{x}) + v_y^2(\mathbf{x})) \\ & + \|\nabla \tilde{I}(\mathbf{x})\|^2 \, \|V(\mathbf{x}) - \nabla \tilde{I}(\mathbf{x})\|^2 \, d\mathbf{x}, \end{aligned} \tag{7}$$

where

$$u_x = \frac{\partial u}{\partial x}, \quad u_y = \frac{\partial u}{\partial y}, \quad v_x = \frac{\partial v}{\partial x}, \quad v_y = \frac{\partial v}{\partial y},$$

and $\tilde{I}$ is a preprocessed version of image I, typically an edge image of some kind. The first term inside the integral encourages a smooth vector field whereas the second term encourages fidelity to $\nabla \tilde{I}$. μ is a free parameter controlling the relative importance of the two terms.

We obtain $\tilde{I}$ using oriented filtering and Canny edge detection (see Figure 3). We use elongated Laplacian of Gaussian filters that emphasize road-like structures, deemphasize non-road-like structures, and, to a certain extent, fill

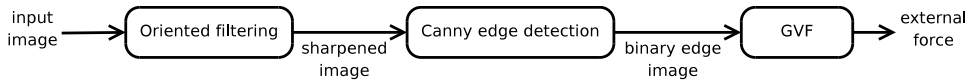


Fig. 3. Schematic of procedure to obtain the GVF external force.

in short gaps where a road has low contrast with the background. The resulting binary Canny image is ideal because it only includes information about road-like edges that have survived sharpening by the oriented filters. The GVF field on top of the sharpened edge image is ideal because it points toward the road-like edges from a long distance, and, during snake evolution, it pushes the snake in an appropriate direction. This speeds evolution and makes it easier to find suitable parameters to obtain fast convergence.

2.2.2 Oriented filtering

Using oriented filters for contour detection, contour completion, and restoration of edges corrupted by noise is a recurring idea in image processing and computer vision (see, e.g., [28–32]). The oriented filters most frequently used are 2D Gabor filters [33] and directional 2nd-derivative-of-Gaussian filters. Gabor filters are thought to be good models of the response of simple cells in primary visual cortex [34]. When paired symmetric (even) and antisymmetric (odd) oriented filter responses are combined by summing their squares, they are thought to be good models of the response of complex cells in primary visual cortex [35]. Perona and Malik [29] advocate these paired "energy filters" for their ability to detect not only step edges but also ridge edges at specific scales.

The ability of Gabor filters and 2nd-derivative-of-Gaussian filters to detect ridge edges makes them ideal for identifying roads in satellite imagery. Our oriented filtering method is the same as that of Rochery et al. [20]. We use

the linear response of even 2nd-derivative-of-Gaussian filters tuned to detect roads at particular scales. We obtain a sharpened image Q defined by

$$Q(\mathbf{x}) = \min_{\theta \in \Theta} \left\{ (\mathcal{F}_\theta * I)(\mathbf{x}) \right\}, \tag{8}$$

where $*$ denotes convolution and the kernel $\mathcal{F}_\theta$ is given by

$$\mathcal{F}_\theta = R_\theta \nabla^2 N_{\sigma_x, \sigma_y}. \tag{9}$$

N_{σ_x,σ_y} is a 2D Gaussian with variance σ_x^2 in the x direction and σ_y^2 in the y direction, R_θ is a matrix rotating N_{σ_x,σ_y} by angle θ, and ∇^2 is the 2D Laplacian. σ_x is chosen according to the desired length of the filter along the road contour whereas σ_y is chosen according to the desired width of the road to detect. We use angles

$$\Theta = \{0, \frac{\pi}{8}, ..., \frac{7\pi}{8}\}.$$

If the roads in I are darker than their surroundings, the maximum rather than the minimum convolution result is used to compute $Q(\mathbf{x})$.

An example of the convolution and minimum response selection procedure is shown in Figure 4(a–j). The filters respond well to long straight edges in the image. This has the effect of emphasizing road-like gradients, deemphasizing non-road-like gradients, and, to a certain extent, filling in short gaps where a road has low contrast with the background.

2.2.3 Obtaining the GVF field

After oriented filtering, we obtain the Canny edge image $\tilde{I}$ from the sharpened image Q. An example is shown in Figure 4(k). This is the input to the GVF relaxation procedure [26]. An example of the resulting GVF field V^{GVF} is shown in Figure 4(l). We precalculate V^{GVF} before snake evolution begins, then dur-

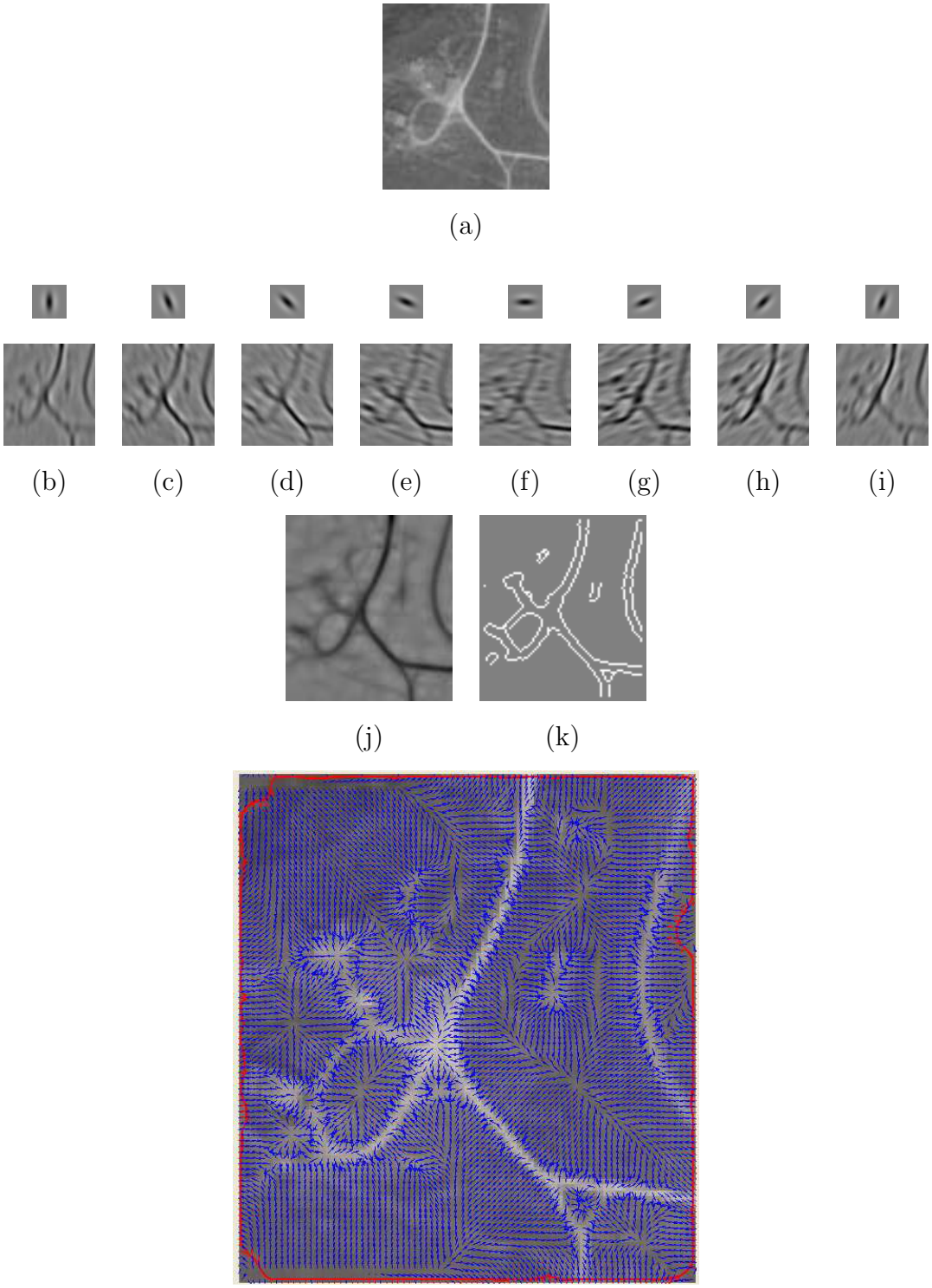


Fig. 4. Image processing for obtaining the GVF. (a) Original image. (b–i) Convolution results. (j) Sharpened image Q. (k) Canny edge image $\tilde{I}$ derived from Q. (l) GVF V^{GVF} based on $\tilde{I}$.

ing evolution, for each point $\gamma(p)$ in Equation 6, we project $V^{\text{GVF}}(\gamma(p))$ onto $\mathbf{n}(\gamma(p))$ and add the resulting force directly to the update equation. Clearly, this encourages the snake to snap to the road edge contours, where ideally $\|V^{\text{GVF}}(\gamma(p))\| = 0$.

2.3 Family of quadratic snakes

A single quadratic snake is unable to extract enclosed regions and multiple disconnected networks in an image. We address this limitation by introducing a *family* of cooperating snakes that are able to split, merge, and disappear as necessary.

In our formulation, due to the curvature term $\kappa(p)$ and the area constant α in Equation 6, specifying the points on γ in a counterclockwise direction creates a *shrinking snake* and specifying the points on γ in a clockwise direction creates a *growing snake*.

An enclosed region (loop or a grid cell) can be extracted effectively by initializing two snakes, one shrinking snake covering the whole road network and another growing snake inside the enclosed region.

2.3.1 Splitting a snake

We split a snake into two snakes whenever two of its arms are squeezed too close together, i.e. when the distance between two snake points is less than d^{split} and those two points are at least k snake points from each other in both directions of traversal around the contour. d^{split} should be less than 2η, where η is the maximum step size.

2.3.2 Merging two snakes

The merging algorithm selects points having high curvature and merges two snakes when 1) two selected points are closer than a prescribed minimal merging distance d^{merge}, 2) the traversal direction (clockwise or counterclockwise) of the two snakes is the same, and 3) the tangents at the two high curvature points are nearly antiparallel. High curvature points are those with $\kappa_\gamma(p) > 0.6\kappa_\gamma^{\max}$, where $\kappa_\gamma^{\max}$ is the maximum curvature for any point on γ. When these conditions are satisfied, the two snakes are combined into a single snake by deleting the high curvature points and merging at the holes.

Limiting the merge decision to high curvature points ensures that merging only occurs if two snakes have semi-circular tips of their arms facing each other. It might seem that merging at low curvature points should also be permitted. However, as already explained, snakes normally repel each other due to the quadratic term in the internal energy (Equation 3). Consequently, low curvature segments can approach each other when high-gradient features allow the external energy to overcome the geometric energy. When this occurs for low curvature segments, the two snakes are most likely positioned on different sides of a road and merging should not be allowed. There are several other (rare) cases when snakes face each other at low curvature parts. However they should not be merged in those cases either.

Considering only the high curvature points also saves computational costs. In particular, the merging procedure requires computation of the angle between tangents only for the selected points. The number of those points usually does not exceed 10% of the total number of points.

The conditions that the traversal direction of two snakes should be the same

and that the tangents at the two high curvature points should be antiparallel reflect the fact that in our system, nested snakes form a tree structure. We initialize all the snakes at the first level with the same direction of traversal. The second level has the opposite direction of traversal and so on. When two snakes from the same level merge, we assign the resulting snake the same direction. Snakes from two consecutive levels do not merge. Growing and shrinking behavior is controlled by the area constant (α) and the weight on the geometric energy (β).

2.3.3 Deleting a snake

A snake γ is deleted if it has perimeter less than L^{delete}.

2.4 Experimental design

We present four experiments aimed at evaluating the effectiveness of the proposed cooperating snake model for road extraction. In Experiment 1, we explore the ability of the model to extract simple tree-structured road networks that do not require multiple snakes. Experiment 2 moves to more complex tree-structured networks with distracting structures. These networks require contours able to split, merge, and disappear in order to ignore noise. Experiment 3 tests the model's ability to capture disconnected networks. In Experiment 4, we evaluate the model's ability to extract networks with cycles with the help of user initialization. Finally, in Experiment 5, we determine the effect of the GVF external force on the model's evolution.

In each experimental condition, we hand-tune the free parameters to achieve

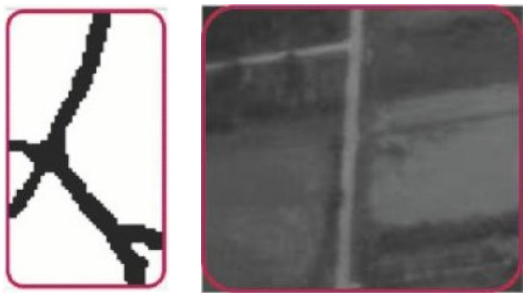
good results. We terminate contour evolution whenever the energy $E_s(\gamma)$ fails to decrease for some number of iterations.

As a baseline for comparison, we use the parametric representation of the single quadratic snake model proposed by Rochery et al. [20]. To evaluate the results, we hand-digitized ground truth images and used them to calculate precision (the proportion of detected pixels that are road pixels according to the ground truth), recall (the proportion of road pixels that are detected), and F_1 (the harmonic mean of precision and recall) for each solution.

3 Results

3.1 Experiment 1: Simple tree-structured networks

We ran a single quadratic snake on the synthetic image shown in Figure 5(a) and the real image shown in Figure 5(b). Our best extraction results are shown in Figure 5(c) and Figure 5(d). The precision and recall results are shown in Table 1. Experiment 1 demonstrates that a single quadratic snake is well suited to simple tree-structured road networks, when the contour does not need to change topology during evolution.



(a) (b)

(c) (d)

Fig. 5. Experiment 1 extraction results. (a) Original synthetic image. (b) Original real image. (c) Road network extracted from the image of (a). (d) Road network extracted from the image of (b).

Table 1

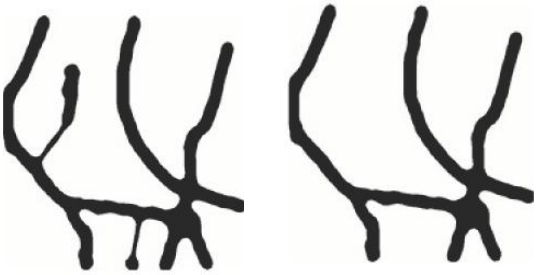
Experiment 1 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Synthetic simple tree	5(c)	480	0.937	0.860	0.897
Real simple tree	5(d)	308	0.948	0.850	0.896

3.2 Experiment 2: Complex tree-structured networks

Beginning with the image shown in Figure 6(a) containing a relatively complex tree-structured road network with distracting road-like structures, we ran a single quadratic snake and the cooperating multiple snake model. The best extraction results for the two models are shown in Figure 6(b) and Figure 6(c), respectively. Table 2 shows the detailed precision and recall results. The cooperating multiple snake model is better able to handle the distracting structures mainly because the evolving snake splits to enclose each road-like structure then the small isolated contours are deleted.

(a)



(b) (c)

Fig. 6. Experiment 2 extraction results. (a) Original image. (b) Road network extracted from the image of (a) with a single snake. (c) Road network extracted from the image of (a) with cooperating snakes that can split, merge, and disappear.

Table 2

Experiment 2 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Complex tree, single snake	6(b)	473	0.709	0.907	0.796
Complex tree, cooperating snakes	6(c)	450	0.801	0.912	0.853

3.3 Experiment 3: Disconnected networks

Beginning with the image shown in Figure 7(a) containing multiple disconnected road networks, we ran a single quadratic snake and the cooperating multiple snake model. The best extraction results for the two models are shown in Figure 7(b) and Figure 7(c), respectively. Table 3 shows the extraction performance details. The cooperating multiple snake model is able to extract the multiple separate road networks, whereas the single snake does its best to model the network with a single contour.

(a)

(b) (c)

Fig. 7. Experiment 3 extraction results. (a) Original image. (b) Road network extracted from the image of (a) with a single snake. (c) Road network extracted from the image of (a) with cooperating snakes that can split, merge, and disappear.

Table 3

Experiment 3 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Disconnected network, single snake	7(b)	219	0.682	0.867	0.764
Disconnected network, cooperating snakes	7(c)	163	0.858	0.873	0.865

3.4 Experiment 4: Networks with cycles

The multiple cooperating snake model cannot extract road networks with loops in a fully automatic fashion, but in our implementation it is simple for the user to manually initialize a separate contour inside each loop. We compared the ability of the single snake, the multiple cooperating snake model with a single initial contour, and the multiple cooperating snake model with user-defined initialization to extract road networks from the images containing loops in Figure 8(a–b). Our best results are shown in Figure 8(c–f). The multiple cooperating snake model obtains excellent results with user-specified initial conditions.

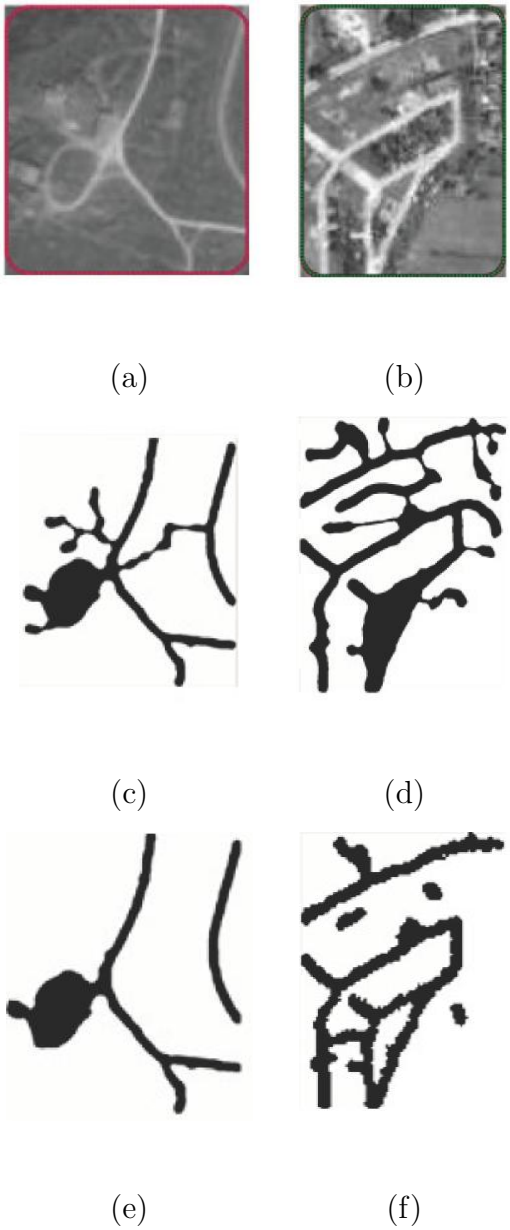


Fig. 8. Experiment 4 extraction results. (a–b) Original images. (c–d) Road networks extracted from images (a–b) with a single snake. (e) Road network extracted from image (a) with cooperating multiple snakes. (f) Road networks extracted from image (b) with cooperating multiple user-defined snakes.

Table 4

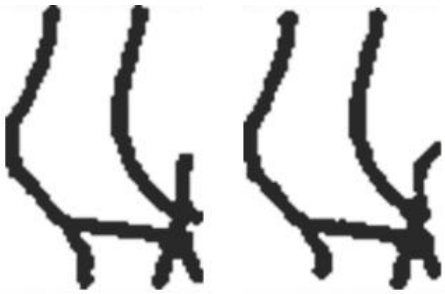
Experiment 4 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Simple loop, single snake	8(c)	182	0.547	0.911	0.684
Complex loop, single snake	8(d)	519	0.530	0.907	0.669
Simple loop, cooperating snakes	8(e)	182	0.6344	0.9833	0.771
Complex loop, user-initialized snakes	8(f)	166	0.768	0.912	0.834

3.5 Experiment 5: GVF external force

Our GVF external force is based on an edge map acquired through oriented filtering, and Canny edge detection. We precomputed the GVF vector field then ran the multiple quadratic snake model on the image shown in Figure 9(a), with and without the GVF force. Our best results for the two experimental conditions are shown in Figure 9(b) and Figure 9(c). The precision and recall results are shown in Table 5. The snakes converge faster in the GVF condition with a slight decrease in precision and recall, although visually the two extracted road networks are of comparable quality.

(a)



(b) (c)

Fig. 9. Experiment 5 extraction results. (a) Original image. (b) Road network extracted from the image of (a) with the multiple snake model and no GVF external force. (c) Road network extracted from the image of (a) with the multiple snake model and the GVF external force.

Table 5

Experiment 5 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Cooperating snakes without GVF	9(b)	650	0.828	0.938	0.880
Cooperating snakes with GVF	9(c)	430	0.802	0.909	0.852

4 Discussion

Experiments 1–5 demonstrate the effectiveness of the proposed multiple cooperating snake model on a variety of road networks. In Experiment 1, we found that a single quadratic snake was sufficient to extract simple tree-structured road networks in synthetic and real imagery. Experiment 2 demonstrated that cooperating snakes converge faster and more accurately than a single snake when the image contains a more complex tree-structured road network with distracting noise. In Experiment 3, we found that the cooperating snake model is effective for extracting disconnected road networks, and in Experiment 4, we found that it is also appropriate for complex networks with cycles, if user-aided initialization is used. Finally, Experiment 5 demonstrated that incorporating an external image force derived from oriented filtering, Canny edge detection, and the GVF provides faster convergence to a minimum-energy configuration.

Our empirical study shows that quadratic snakes avoid self-intersections by incorporating into the energy functional the constraint that contour segments with anti-parallel tangents should repel each other. They are nevertheless still able to enter long and narrow concavities and to approach each other close enough to extract road networks in satellite images.

Our parametric specification of the family of cooperating snakes is simple and efficient compared to level set methods. If a family of snakes is represented by N discrete points, a naive implementation of the evolution requires $O(N^2)$ time per iteration, since Equation 6 must be applied to each of the N points and it involves an integral over all N points. Likewise, naive implementation of the split and merge algorithms developed in Section 2.3 consider at most each

pair of points, so assuming at most a constant number of splits and merges per iteration, the complexity remains $O(N^2)$. However, it is possible to reduce the runtime to $O(N)$. The contour update and split/merge algorithms only perform comparisons with other contour points within a fixed local region, so it is possible to preindex the contour points in the image domain such that at most a constant number of other points are considered for each point on the contour.

One limitation of our approach is that we require manual initialization of the contours to obtain good results when the road network contains loops. The level set method's main strength is its ability to handle such loops without any special treatment. However, our method could be extended to handle this case, if we added autodetection of "holes" in a converged family of contours or if we initialized with many small growing snakes inside a shrinking outer snake, either randomly or in a grid pattern.

A more serious limitation of our approach is the need to determine free parameters such as α, β, and λ empirically. In constrained applications such as road extraction, it should be possible to develop a database of useful parameter settings for particular image resolutions and road network types. But sensitivity to parameter settings is the Achilles' heel of all active contour models; unless this problem is solved, the technique's applicability to real-world GIS problems will be limited.

5 Conclusion

The proposed method is an implementation of multiple active contours in a variational framework based on a quadratic energy functional. Our model performs better than conventional snakes and single parametric quadratic snakes on road extraction tasks. The combination of oriented LoG filters, Canny edge detection, and the GVF provides effective preprocessing of noisy and distorted images and an appropriate external force for the quadratic snake's energy functional. The multiple snake configuration that minimizes the total energy functional accurately snaps to the boundaries of objects. The energy functional's quadratic terms, which encourage parallel tangents and discourage anti-parallel tangents, and its sigmoid interaction function, are designed to extract curvilinear ribbon-like structures such as roads, canals, and pipelines from digital images. We have shown that the scheme is effective at extracting road networks from a series of satellite images, but some calibration of the free parameters is required to achieve good results.

In future research we plan to focus on automatic initialization of contours to handle networks with cycles and reducing the method's run time to the point that it is practical for application in GIS applications. It may also be possible to develop higher order active contour models for other important segmentation problems involving extraction of other shapes such as ellipses and polygons.

6 Acknowledgments

This work was supported by Thailand Research Fund grant MRG4780209 to MND. RM was supported by a graduate fellowship from the Nepal High Level Commission for Information Technology. Ran Zask assisted with software development. We are grateful to Kiyoshi Honda for helpful comments on this research.

References

[1] D. Geman, B. Jedynak, An active testing model for tracking roads in satellite images, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (1) (1996) 1–14.

[2] M. Fischler, J. Tenenbaum, H. Wolf, Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique, Computer Graphics and Image Processing 15 (1981) 201–223.

[3] M. Barzohar, D. Cooper, Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (7) (1996) 707–721.

[4] P. Fua, Y. Leclerc, Model driven edge detection, Machine Vision and Applications 3 (1990) 45–56.

[5] C. Regazzoni, G. Foresti, S. Serpico, An adaptive probabilistic model for straight edge-extraction within a multilevel MRF framework, in: International Geoscience and Remote Sensing Symposium, 1995, pp. 458–460.

[6] F. Tupin, H. Maître, J.-F. Mangin, J.-M. Nicolas, E. Pecersky, Detection of linear features in SAR images: Application to road network extraction, IEEE

Transactions on Geoscience and Remote Sensing 36 (2) (1998) 434–453.

[7] M.-F. Auclair-Fortier, D. Ziou, C. Armenakis, S. Wang, Survey of work on road extraction in aerial and satellite images, Tech. Rep. 247, Departement de mathematiques et d'informatique, Universitede Sherbrooke (2000).

[8] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, International Journal of Computer Vision 1 (4) (1987) 321–331.

[9] L. D. Cohen, I. Cohen, Finite-element methods for active contour models and balloons for 2-d and 3-d images, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (11) (1993) 131–147.

[10] Y. Wong, P. Yuen, C. Tong, Segmented snake for contour detection, Pattern Recognition 31 (11) (1998) 1669–1679.

[11] J. Ivins, J. Porrill, Active region models for segmenting regions and colors, Image and Vision Computing 13 (5) (1995) 431–438.

[12] R. Samadani, Changes in connectivity in active contour models, in: Proceedings of the Workshop on Vision Motion, 1989, pp. 337–343.

[13] R. Durkovich, K. Kaneda, H. Yamashita, Dynamic contour: A texture approach and contour operations, Visual Computing 11 (1995) 227–289.

[14] K. Ngoi, J. Jia, An active contour model for color region extraction in natural scenes, Image and Vision Computing 17 (3) (1999) 955–966.

[15] W. Choi, K. Lam, W. Siu, An adaptive contour model for highly irregular boundaries, Pattern Recognition 34 (2) (2001) 323–331.

[16] H. Delingnette, J. Montagnat, New algorithm for controlling active contour shape and topology, in: Sixth European Conference on Computer Vision (ECCV), Vol. 2, 2000, pp. 381–395.

[17] L. Ji, H. Yan, Robust topology-adaptive snakes for image segmentation, Image and Vision Computing 20 (2002) 147–164.

[18] T. McInerney, D. Terzopoulos, T-snakes: Topology adaptive snakes, Medical Image Analysis 4 (2) (2000) 73–91.

[19] G. Giraldi, E. Strauss, A. Oliveira, Dual-T-Snakes model for medical imaging segmentation, Pattern Recognition Letters 24 (2003) 993–1003.

[20] M. Rochery, I. H. Jermyn, J. Zerubia, Higher order active contours, International Journal of Computer Vision 69 (1) (2006) 27–42.

[21] V. Caselles, F. Catte, T. Coll, F. Dibos, A geometric model for active contours, Numerische Mathematik 66 (1993) 1–31.

[22] R. Malladi, J. Sethian, B. Vemuri, Shape modeling with front propagation: A level set approach, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) 158–175.

[23] J. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, 1999.

[24] C. Xu, D. Pham, J. Prince, Image segmentation using deformable models, in: Handbook of Medical Imaging Volume 2: Medical Image Processing and Analysis, SPIE Press, 2000, pp. 129–174.

[25] C. Li, J. Liu, M. Fox, Segmentation of external force field for automatic initialization and splitting of snakes (2005).

[26] C. Xu, J. L. Prince, Gradient Vector Flow: A new external force for snakes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1997, pp. 66–71.

[27] C. Xu, J. Prince, Snakes, shapes, and gradient vector flow, IEEE Transactions on Image Processing 7 (3) (1998) 359–369.

[28] H. Knutsson, R. Wilson, G. Granlund, Anisotropic nonstationary image estimation and its applications: Part I — restoration of noisy images, IEEE Transactions on Communications COM-31 (3) (1983) 388–397.

[29] P. Perona, J. Malik, Detecting and localizing edges composed of steps, peaks, and roofs, in: International Conference on Computer Vision, 1990, pp. 52–57.

[30] W. Freeman, E. Adelson, The design and use of steerable filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (9) (1991) 891–906.

[31] C. Steger, An unbiased detector of curvilinear structures, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (2) (1998) 113–125.

[32] S. Konishi, A. Yuille, J. Coughlan, S. Zhu, Statistical edge detection: Learning and evaluating edge cues, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (1) (2003) 57–74.

[33] J. G. Daugman, Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters, Journal of the Optical Society of America A 2 (1985) 1160–1169.

[34] J. P. Jones, L. A. Palmer, An evaluation of the two-dimensional Gabor filter model of receptive fields in cat striate cortex, Journal of Neurophysiology 58 (6) (1987) 1233–1258.

[35] F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, O. Kübler, Simulation of neural contour mechanisms: From simple to end-stopped cells, Vision Research 32 (5) (1992) 963–981.