International Journal of Information Technology and Knowledge Management
January-June 2009, Volume 2, No. 1, pp. 1-3

A NOVEL METHOD FOR BOUNDARY DETECTION BASED ON EDGE FLOW TECHNIQUE

C. Nagaraju*, L. S. S. Reddy* & D. Ramesh Babu*

A novel boundary detection method based on edge flow is proposed in this paper. This method utilizes a predictive coding model to identify the direction of change in color and texture at each image location at a given scale, and constructs an edge flow vector. By iteratively propagating the edge flow, the boundaries can be detected at image locations which encounter two opposite directions of flow in the stable state. A user defined image scale is the only significant control parameter that is needed by the algorithm.

Keywords: Edge flow, Segmentation, Predictive coding and Scale parameter

1. Introduction

The goal of texture segmentation is to partition an image into homogeneous regions and identify the boundaries which separate regions of different textures. Segmentation is obtained either by considering a gradient in the texture feature space [4, 7, 8], or by unsupervised clustering [1, 3, 5], or by texture classification [9]. Segmentation by labeling often suffers from a poor localization performance because of the conflicting requirements of region labeling and boundary localization in terms of the observation neighborhood [11]. Unsupervised segmentation requires an initial estimate of the number of the regions in the image, which is obtained mostly by setting a threshold in the feature clustering algorithm. However, estimating the number of regions is a difficult problem and the results are usually not reliable.

In this paper edge flow technique is used for Image segmentation. Traditionally edges are located at the local maxima of the gradient in intensity or image feature space. In contrast, in our approach the detection and localization of edges or image boundaries are performed indirectly: first by identifying a flow direction at each pixel location that points to the closest boundary; then followed by the detection of locations that encounter two opposite directions of edge flow. Since any of the images attributes such as color, texture, or their combination can be used to define the edge flow, this scheme provides image information for boundary detection.

The general form of edge flow vector at image location with an orientation is defined as:

$$F(s, \theta) = F[E(s, \theta), P(s, \theta), P(s, \theta + \Pi)] \quad (1)$$

where

- $E(s, \theta)$ is the edge energy at location along the orientation θ.
- $P(s, \theta)$, represents the probability of finding the image boundary if the corresponding flow at location 's' flows in the direction θ.
- $P(s, \theta + \Pi)$ represents the probability of finding the image boundary if the corresponding flow at location s flows backwards, i.e., in the direction $\theta + \Pi$.

1.1 Computing $E(s, \theta)$

Now consider an image at a given scale σ as $I_\sigma(x, y)$, which is obtained by smoothing the original image $I(x, y)$ with a Gaussian kernel $G_\sigma(x, y)$. The scale parameter will control both the edge energy computation and the local flow direction estimation, so that only edges larger than the specified scale are detected. The edge flow energy $E(s, \theta)$ at scale σ is defined to be the magnitude of the gradient of the smoothed image $I_\sigma(x, y)$ along the orientation θ:

$$E(s,\theta) - \left|\frac{\partial}{\partial n} I_\sigma(s,y)\right| - \left|\frac{\partial}{\partial n}\left[I(x,y) * G_s(x,y)\right]\right| - \left|I(x,y) * \frac{\partial}{\partial n} G_\sigma(x,y)\right| \quad (2)$$

where $s = (x, y)$ and n represents the unit vector in the gradient direction. We can rewrite (2) as

$$E(s, \theta) = |I(x, y) * GD_{\sigma, \theta}(x, y)|. \quad (3)$$

This edge energy indicates the strength of the intensity change. Many existing edge detectors actually use similar operations to identify the local maxima of intensity changes as edges. The distinguishing part of the edge flow model is that the edge energy is represented as a flow vector by assigning probabilities to its flow directions. Boundary

* Professor in CSE, K. L. College of Engineering, Vijayawada-522502, *E-mail: cnrcse@yahoo.com, rameshdamarla@gmail.com*

detection itself is formulated as a dynamic process wherein the local edge energies *flow* in the direction of most probable image boundaries closest to the corresponding locations.

1.2 Computing $P(s, \theta)$

For each of the edge energy along the orientation at location, we now consider two possible flow directions; the forward (θ) and the backward ($\theta + \Pi$), and estimate the probability of finding the nearest boundary in each of these directions. These probabilities can be obtained by looking into the prediction errors toward the surrounding neighbors in the two directions. Consider the use of image information at location s to predict its neighbor in the direction θ. Ideally they should have similar intensity if they belong to the same object and the prediction error can thus be computed as

$$Error\ (s, \theta) = I_{\sigma}\ (x + d \cos \theta, y + d \sin q) - I_{\sigma}\ (x, y)| = |I\ (x, y) + DOOG_{\sigma,\ \theta}\ (x, y) \quad (4)$$

where is the distance of the prediction, and which should be proportional to the scale at which the image is being analyzed, in the experiment we choose $d = 4\sigma$.

A large prediction error in a certain direction implies a higher probability of finding a boundary in that direction. Therefore, the probabilities of edge flow direction are assigned in proportion to their corresponding prediction errors:

$$P(s, \theta) = \frac{Error(s,\theta)}{Error(s,\theta) + Error(s,\theta+\pi)} \quad (5)$$

The idea of this approach to computing the flow probabilities comes from [2, 10]. It has been suggested that the human visual system uses a predictive coding model to process image information. This model has been successfully used in interpreting many vision phenomena such as the retinal inhibitory interactions [10], and the coding of textured patterns [2].

The first component $E\ (s, \theta)$ of edge flow is used to measure the energy of local image information change such as intensity, color, texture, and phase difference and the remaining two components $P(s, \theta)$ and $P(s, \theta + \Pi)$ are used to represent the probability of flow direction.

The basic steps for detecting image boundaries are summarized as follows:

- At each image location, we first compute its local edge energy and estimate the corresponding flow direction.
- The local edge energy is iteratively propagated to its neighbor if the edge flow of the corresponding neighbor points in a similar direction.
- The edge energy stops propagating to its neighbor if the corresponding neighbor has an opposite direction of edge flow. In this case, these two image locations have both their edge flows pointing at each other indicating the presence of a boundary between the two pixels.
- After the flow propagation reaches a stable state, all the local edge energies will be accumulated at the nearest image boundaries. The boundary energy is then defined as the sum of the flow energies from either side of the boundary.

2. METHODOLOGY

After the edge flow $F(s)$ of an image is computed, boundary detection can be performed by iteratively propagating the edge flow and identifying the locations where two opposite direction of flows encounter each other. At each location, the local edge flow is transmitted to its neighbor in the direction of flow, if the neighbor also has a similar flow direction. The steps are:

(1) Set $n = 0$ and $F_0\ (s) = F(s)$.

(2) Set the initial edge flow $Fn_{+1}(s)$ at time $n + 1$ to zero.

(3) At each image location s, identify the neighbor $s' = (x', y')$ which is in the direction of edge flow $F_n(s)$.

(4) Propagate the edge flow if $Fn(s)$. $F_n(s') > 0$: $F_{n+1}(s') = F_{n+1}(s') + F_n(s)$; otherwise the edge flow stay at its original location, $F_{n+1}(s) = F_{n+1}(s) + F_n(s)$.

(5) If nothing has been changed, stop the iteration. Otherwise, set $n = n+1$ and go to the step 2 and repeat the process.

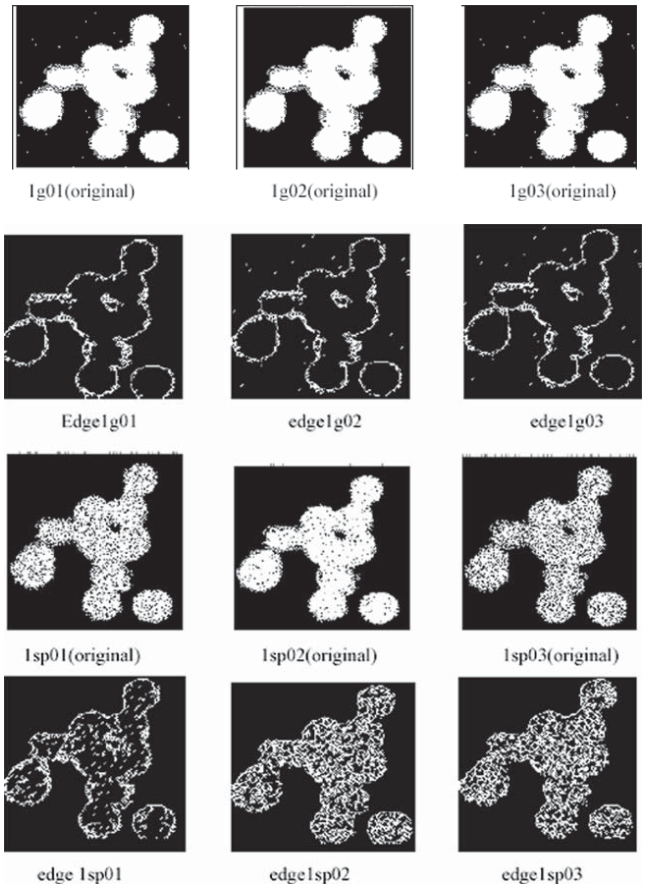
Once the edge flow propagation reaches a stable state, we can detect the image boundaries by identifying the locations which have non-zero edge flows coming from two opposing directions.

3. EXPERIMENTAL RESULTS

The Images contain intensity, texture, and illusory boundaries respectively. This paper constructed the edge flow field for these different image attributes. The final segmentation results after the post-processing are illustrated. As can be seen, these images, which traditionally require different algorithms to segment, can now be processed using the edge flow model. Shown below are a few more segmentation results on some typical images such as blood cells.

4. CONCLUSIONS

In this paper we have presented a novel method for detecting image boundaries and demonstrated its use in segmenting a large variety of natural images. In contrast to the traditional approaches, the edge flow model utilizes a predictive coding



scheme to detect the direction of change in various image attributes and construct an edge flow field. By iteratively propagating the edge flow, the boundaries can be detected at image locations which encounter two opposite directions of flow in the stable state. The only significant control parameter is the image scale, which can be adjusted to the user's requirements. For simplicity, a single scale parameter has been used for the entire image segmentation in our current implementation. This local scale control remains as a future research problem. However, our experiments with some of the textures have given results better than most of the algorithms that we are currently aware of in the segmentation literature.

References

[1] A. C. Bovic, M. Clark, W. S. Geisler, "Multichannel Texture Analysis using Localized Spatial Filters," *IEEE Trans. Pattern Anal. and Machine Intell*., **12**, (January 1990), 55–73.

[2] J. G. Daugman and C. J. Downing, "Demodulation, Predictive Coding, and Spatial Vision," *Journal of the Optical Society of America A*, **12**, (4), (April 1995), 641–660.

[3] D. Dunn, W. E. Higgins, and J. Wakeley, "Texture Segmentation using 2-D Gabor Elementary Functions," *IEEE Trans. Pattern Anal. and Machine Intell*., **16**, (Feb. 1994), 130–149.

[4] I. Fogel and D. Sagi, "Gabor Filters as Texture Discriminator," *Biological Cybernetics*, **61**, (1989), 103–113.

[5] A. K. Jain and F. Farroknia, "Unsupervised Texture Segmentation using Gabor Filters," Pattern Recognition, **24**(12), (1991), 1167–1186.

[6] K. Karu, A. K. Jain, and R. M. Bolle, "Is There Any Texture in the Image?" *Pattern Recognition*, **29**, (9), (1996), 1437–1446.

[7] J. Malik and P. Perona, "Preattentive Texture Discrimination with Early Vision Mechanisms," *J. Opt. Soc. Am. A*, **7**, (May 1990), 923–932.

[8] B. S. Manjunath and R. Chellappa, "A Unified Approach to Boundary Detection," *IEEE Trans. Neural Networks*, **4**, (1), (January 1993), 96–108.

[9] J. Mao and A. K. Jain, "Texture Slassification and Segmentation using Multiresolution Simultaneous Autoregressive Models," *Pattern Recognition*, **25**, (2), (1992), 173–188.

[10] M. V. Srinivasan, S. B. Laughlin, and A. Dubs, "Predictive Coding: A Fresh View of Inhibition in the Retina," *Proc. R. Soc. London Ser. B* **216**, (1982) 427–459.

[11] S. R. Yhann and T. Y. Young, "Boundary Localization in Texture Segmentation," *IEEE Trans. Pattern Anal. and Machine Intell*., **4**, (June 1995), 849–855.